

Dynamic Bandwidth Management for the Internet and its Wireless Extensions

Raymond Rui-Feng Liao

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Graduate School of Arts and Sciences

Columbia University

2003

© 2003

Raymond Rui-Feng Liao

All Rights Reserved

ABSTRACT

Dynamic Bandwidth Management for the Internet and its Wireless Extensions

Raymond Rui-Feng Liao

Over the past decade network bandwidth has become a commodity item putting pressure on Internet Service Providers (ISPs) to differentiate their service offerings to customers in order to maintain market share. However, realizing service differentiation in IP networks is a broad, multi-dimensional and challenging problem. This thesis addresses this problem and proposes new approaches for bandwidth service management for the Internet and its wireless extensions that include: (i) utility-based adaptation mechanisms, which capture applications needs and address the technical challenges of supporting controlled service degradation in differentiated service networks; (ii) dynamic provisioning for core networks, which resolves the technical issues associated with managing complex traffic aggregates and delivering quantitative differentiated services in networks with limited state information and control mechanisms; and (iii) incentive engineering, which effectively deals with the arbitrage problem inherent in differentiated services models. We take a systems approach to these problems and investigate new policy modeling, algorithms, and protocol design techniques. We evaluate our research ideas using a combination of analysis, simulation, and results from an experimental wireless testbed.

This thesis makes a number of contributions. Our study is founded on bandwidth utility functions, which are capable of capturing the intrinsic adaptability of applications to bandwidth changes. First, we propose a unified formulation of bandwidth utility functions for application aggregates including TCP, small audio flows,

and individual video flows. We discuss experiments using the online generation of utility functions from video traces and present a utility prediction algorithm that addresses the time scale mismatch that exists between video content changes and network adaptation time-scales.

Next, we present two groups of utility-based link allocation algorithms that provide a foundation for utility differentiating and utility maximizing bandwidth management. The utility maximizing algorithm leverages the piecewise linear quantization of utility functions and uses the Kuhn-Tucker condition to significantly reduce the algorithm execution time. Our utility differentiating algorithm supports utility fair allocation that allows individual utility functions to have different maximum utility values. We extend these results to the problem of multi-hop utility-based flow control by augmenting the max-min flow control algorithm to support utility functions. We study, propose and evaluate a utility-based max-min fair allocation and renegotiation protocol in the context of an edge-based wireless access network taking into consideration convergence speed, protocol state reduction, and the management of application adaptation states.

Third, we present a dynamic bandwidth provisioning model for quantitative service differentiation in core networks that comprises node and core provisioning algorithms. The node provisioning algorithm prevents transient violations of Service Level Agreements (SLAs) by predicting the onset of service level violations based on a multi-class virtual queue technique, self-adjusting per-class service weights, and packet dropping thresholds at core routers. Persistent service level violations are reported to a dynamic core provisioning algorithm, which dimensions traffic aggregates at the network ingress taking into account fairness issues not only across different traffic aggregates but also within the same aggregate whose packets can take different routes in the core IP network. We solve the problem of rate regulation

for point-to-multipoint flow aggregates with the use of matrix inverse operations. We demonstrate that our model is capable of delivering capacity provisioning in an efficient manner and providing quantitative delay-bounds with differentiated loss across per-aggregate service classes.

Finally, we propose incentive engineering techniques and design two incentive-based allocation service classes that effectively constrain the strategy space of subscribers to a set of cooperative behaviors that include the truthful selection of a service class and truthful declaration of bandwidth demands. Our design minimizes protocol messaging overhead imposed on wireless subscribers while possessing a number of beneficial properties including Nash bargaining fairness for the instantaneous allocation service, and incentive compatibility for mobile users promoting the truthful declaration of their service preferences.

Contents

1	Introduction	1
1.1.	Overview	1
1.2.	Technical Barriers	6
1.2.1	Application-Aware Controlled Service Degradation	6
1.2.2	Quantitative Service Differentiation for Traffic Aggregates	8
1.2.3	Creating Incentives for Service Differentiation	9
1.3.	Thesis Outline	10
1.3.1	Utility Function Formulation	10
1.3.2	Utility-Based Link Allocation Algorithms	11
1.3.3	Utility-Based Adaptation in Wireless Access Networks	12
1.3.4	Quantitative Service Differentiation for Traffic Aggregates in Core Networks	13
1.3.5	Incentive Engineering for Service Differentiation in Wireless Access Networks	14
1.4.	Thesis Contribution	15
2	Utility Function Formulation: A Unifying Abstraction for Net- works, Applications and Content	17
2.1.	Introduction	17
2.1.1	Utility-based Bandwidth Allocation Framework	17

2.1.2	Utility Formulation	21
2.2.	Related Work	23
2.3.	Utility Formulation	24
2.3.1	Utility Metrics	24
2.3.2	Utility Generation for Adaptive Video	27
2.3.3	Utility Formulation for Application Aggregates	32
2.4.	Utility Prediction for Video	37
2.5.	Evaluation of Utility Prediction Algorithm	40
2.5.1	Experiment Setup	40
2.5.2	Algorithm Evaluation	42
2.5.3	Error Analysis	47
2.6.	Summary	50
A.	Optimum Utility Prediction Time-Scale	51
3	Utility-Based Link Allocation Algorithms	55
3.1.	Introduction	55
3.2.	Related Work	57
3.3.	Proportional Utility-Differentiated Allocation	58
3.3.1	Proportional Utility-Fair	58
3.3.2	Proportional Utility-Differentiation	59
3.4.	Utility-Maximizing Allocation	61
3.4.1	Algorithm Formulation	61
3.4.2	Algorithm Evaluation	65
3.4.3	Aggregation State	67
3.4.4	Priority Allocation	67
3.5.	Implementation and Evaluation of Utility-Based Allocation	68
3.5.1	Utility-Based Hierarchical and Hybrid Link Sharing	69

3.5.2	Simulation Results	71
3.6.	Summary	80
4	Utility-Based Adaptation in Wireless Access Networks	81
4.1.	Introduction	81
4.2.	Related Work	84
4.3.	Utility-Based Adaptation Model for Wireless Access Networks	85
4.3.1	Utility-Based Network Control	85
4.3.2	Policy-Based Application Adaptation	88
4.4.	Utility-Based Network Control	90
4.4.1	Definition of Utility-Based Max-min Fairness	90
4.4.2	Distributed Algorithm	91
4.4.3	Resource Probing Protocol	93
4.4.4	Convergence Property	95
4.5.	Policy-Based Application Adaptation	97
4.5.1	Greedy Adaptation Script	98
4.5.2	Discrete Adaptation Script	99
4.5.3	Smooth Adaptation Script	101
4.5.4	Handoff Adaptation Script	101
4.6.	Simulation	103
4.6.1	Simulation Environment	104
4.6.2	Fairness Metric	107
4.6.3	Results	108
4.7.	Summary	113
A.	Pseudo-code for the Utility-weighted Max-min Fair Allocation Algo- rithm	114

5	Quantitative Service Differentiation for Traffic Aggregates in Core Networks	118
5.1.	Introduction	118
5.2.	Related Work	121
5.3.	A Dynamic Bandwidth Provisioning Model for Core Networks	124
5.3.1	Architecture	124
5.3.2	Control Messaging	125
5.3.3	Service Model	126
5.4.	Dynamic Node Provisioning	129
5.4.1	Loss Measurement	130
5.4.2	Delay Constraint	131
5.4.3	Virtual Queue Scaling	132
5.4.4	Control Action	134
5.5.	Dynamic Core Provisioning	138
5.5.1	Core Traffic Load Matrix	138
5.5.2	Edge Rate Reduction Policy	140
5.5.3	Edge Rate Alignment	146
5.6.	Simulation Results	148
5.6.1	Simulation Setup	148
5.6.2	Dynamic Node Provisioning	150
5.6.3	Dynamic Core Provisioning	161
5.7.	Summary	166
6	Incentive Engineering for Service Differentiation in Wireless Access Networks	169
6.1.	Introduction	169
6.2.	Economics Background and Related Work	172

6.3.	Incentive Engineering Model for WLAN Access Networks Overview .	176
6.3.1	Network Model	176
6.3.2	Service Purchasing Power	177
6.3.3	Price-Service Menu	179
6.3.4	IA and SA Algorithms	180
6.4.	Incentive Engineering for IA Class	181
6.4.1	Baseline IA Algorithm	181
6.4.2	Measurement-Based Price Calculation	182
6.4.3	Optimistic Rate Allocation with Incomplete Information . . .	184
6.5.	Incentive Engineering for SA Class	188
6.5.1	Baseline SA Algorithm	188
6.5.2	IA Allocation Pegging	192
6.6.	Mobile Device Strategy	195
6.6.1	Fairness	195
6.6.2	Dominant Mobile Strategy	196
6.7.	Experimental Results	199
6.7.1	Wireless Testbed	199
6.7.2	Parameter Tuning	200
6.7.3	IA and SA Allocation Algorithm	203
6.7.4	Pricing Dynamics	206
6.8.	Summary	208
7	Conclusion	210
8	My Publications as a Ph.D. Candidate	215
8.1.	Patents	215
8.2.	Journal Papers	215

8.3. Journal Papers under Submission	216
8.4. Magazine Papers, Review Articles and Book Chapters	216
8.5. Conference Papers	217
References	219

List of Figures

1-1	QOS Mechanisms over Different Time-Scales	3
2-1	Different Styles of Utility Functions	18
2-2	Utility-Based Bandwidth Allocation Framework	19
2-3	A Video Scaling Profile	31
2-4	Example of Normalized $v(x)$ for TCP	35
2-5	Pseudo-code of Long-range Utility Prediction Algorithm	39
2-6	Example of Instantaneous Utility Functions	42
2-7	Algorithm Sensitivity to T	45
2-8	Examples of Predicted Utility Function Envelopes	46
2-9	Utility Prediction Error (Trace 3, $T=20$)	48
2-10	Time-averaged Over-estimation Error	49
3-1	Example of Utility-maximizing Aggregation	63
3-2	Pseudo-code of Utility-Maximization Algorithm	64
3-3	Performance of Utility-based Utility Maximization Algorithm	66
3-4	Examples of Utility Aggregation under Utility Maximization	67
3-5	Example Structure of $U(x)$ -CBQ Link Sharing Server	70
3-6	$U(x)$ -CBQ Link Sharing Simulation Setup	73
3-7	Aggregated Utility Function in Case 1	74
3-8	Utility Distribution in Case 1	76

3-9	Results for Link Sharing Case 2	78
4-1	Utility-based Adaptation Model for Wireless Access Networks	86
4-2	Simple Adaptation Script Schema	89
4-3	Simple Greedy Adaptation Script	99
4-4	Two Discrete Adaptation Scripts	100
4-5	Smooth Adaptation Script	102
4-6	Handoff Adaptation Script Results	103
4-7	Simulated Mobile Access Network Topology	104
4-8	Utility Functions Used in Simulations	105
4-9	Greedy Adaptation Results: Fairness Index	109
4-10	Greedy Adaptation Results: $\overline{FI}$ v.s. Probing Cycle	110
4-11	Greedy Adaptation Results: Utility Value	110
4-12	Discrete Adaptation Script Results: Utility Value	111
4-13	Comparison of Discrete Adaptation Strategies	112
4-14	Smooth Adaptation Script Results	113
5-1	Dynamic Bandwidth Provisioning Model for Core Networks	125
5-2	Example of κ Values	134
5-3	Node Provisioning Algorithm Pseudo-code	137
5-4	Example of a Network Topology and its Traffic Matrix	139
5-5	Edge Rate Reduction Algorithm Pseudo-code	147
5-6	Edge Rate Alignment Algorithm Pseudo-code	147
5-7	Simulated Network Topology	150
5-8	Node Provisioning Service Differentiation Effect: Throughput	152
5-9	Node Provisioning Service Differentiation Effect: Mean Delay	153
5-10	Node Provisioning Service Differentiation Effect: Loss	153

5-11	Node Provisioning Control Parameters	154
5-12	Node Provisioning Sensitivity to <i>update_interval</i> , AF1 Class with Pareto On-Off Traffic	156
5-13	Node Provisioning Algorithm Performance, AF1 Class with Bursty Traffic Load	158
5-14	Node Provisioning Algorithm Performance, AF1 Class with TCP Ap- plications	160
5-15	Reduction Policy Comparison (Ten Independent Tests)	162
5-16	Core Provisioning Allocation Result, Default Policies	163
5-17	Average Bandwidth Allocation for AF1 Aggregates	165
5-18	Delay for AF1 Aggregates (Averaged over 10s)	166
6-1	Wireless LAN Based Mobile Access Network	177
6-2	Example of Aggregated IA Price Function	184
6-3	Baseline IA Allocation Algorithm	188
6-4	Example of Aggregated IA & SA Price Function	190
6-5	Baseline SA Allocation Algorithm at Access Point	192
6-6	Experimental Wireless Testbed	199
6-7	Linux Traffic Control Response	201
6-8	Parameter Setting	202
6-9	IA Allocation Experiment	204
6-10	IA/SA Allocation Experiments	205
6-11	SA Service: Allocation Stability Ranking	206
6-12	Relation between SA and IA Prices	207
6-13	Effect of T_S on the SA Price	208
6-14	Additional Service Purchasing Power for Allocation Guarantee	209

List of Tables

2.1	Video Traces Used in Experiment	40
2.2	Sensitivity to Initial Value of e (T=30s)	44
4.1	Flow Utility Curve Parameters	105
5.1	Traffic Distribution Matrix	162

Acknowledgements

I would like to thank my advisor Professor Andrew T. Campbell for supporting me throughout my time at Columbia University. His hard working attitude will be my role model, forever. I will always appreciate his cheerful encouragement during my studies. He always inspired me to pursue ideas beyond the traditional boundaries. I am particularly grateful for his generous efforts in providing me with exposures to both industry and academia, and helping me develop much needed soft skills that you need as a mature researcher.

I would also like to thank Professor Aurel A. Lazar for his role in helping spark part of the work presented in thesis, and for serving on my thesis proposal and defense committees. Next, I would like to express my sincere thanks to Professor Mischa Schwartz for many insightful comments on my work and for also serving on my thesis proposal and defense committees. I would also like to thank the other members of my defense committee for kindly taking time out from the busy schedules to serve. Many thanks to Professors Edward G. Coffman, Jr., Jörg Liebeherr, and Jason Nieh.

I am so very grateful to Professor Andy Hopper, and Drs. Martin Brown and Glenford Mapp for my stay working with them during the summer of 1999 at the AT&T Labs in Cambridge, UK. I have many valuable memories of that summer working on Linux network programming and wireless broadband network deployment.

This thesis is dedicated to my dear wife Debbie Yi-Ching Lin for enduring my erratic and hectic work schedule, to my parents on both sides, and to my brother and sister-in-law for their support.

Many members of the COMET and ADVENT groups in the Electrical Engineering Department have contributed to my research and have made my time at

Columbia University a truly memorable one. In particular, I would like to thank Dr. Nemo Semret for working with me during my first three years as a Ph.D. candidate, and for the many stimulating discussions since. Thanks to Dr. Mun-Choon Chan for introducing me to the programmable network (Xbind) project and for many enjoyable times since. I would like to thank Rita H. Wouhaybi for her collaboration on the implementation of the incentive engineering algorithm in the experimental wireless testbed discussed in Chapter 6. Thanks to Dr. Paul Bocheck and Professor Shih-Fu Chang for collaborating with me on video utility function generation and with developing the scene predication algorithm. Thanks also to Dr. Steve Jacobs and Professor Alexander Eleftheriadis for providing me with the source code of their very neat dynamic rate shaping algorithm and explaining how that algorithm works in detail. I would like to thank several project students who have worked with me over the last few years: Suhail Mohiuddin for implementing the utility maximization algorithm discussed in Section 3.4.2; Pradu Bouklee for experimenting with the utility prediction algorithms discussed in Section 2.4.; as well as Stefan Berger, Stephen T. Chou, Kijoon Hong, Bong Jun Ko, Suhail Mohiuddin and Vassilis Stachtos for implementing an early version of the experimental wireless testbed discussed in Section 6.7.1 of this thesis.

That is a long list of people to thank. All contributed in one way or another to this thesis. To them all - I'm eternally thankful.

Chapter 1

Introduction

1.1. Overview

The tremendous growth of Internet users, service providers, and networking infrastructure over the past decade has effectively driven network bandwidth close to a commodity item [81]. As a result, service providers are under mounting pressure to differentiate their service offerings to their customers in order to maintain market share. In order for telecommunications carriers and Internet Service Providers (ISPs) to differentiate themselves from their competitors in this manner there is a need to better understand the challenges in offering service differentiation to customers in order to engineer scalable and efficient bandwidth management mechanisms and policies into the global Internet. These challenges are not solely in the domain of network quality of service (QoS) research and development. Rather, they also impact other related areas such as controlled service degradation and pricing, as well as non-technical concerns such as marketing (e.g., branding and market segmentation). These challenges present a number of technical barriers that complicate the design and wide-scale deployment of service differentiation and bandwidth management techniques. This thesis concerns itself with this problem and investigates, designs and analyzes new bandwidth service models, control algorithms, and

policies for the Internet and its wireless extensions.

The implications of product differentiation using commodity bandwidth services can be better understood by studying other related commodity markets. The electricity distribution network, which evolved over a much longer period into a commodity, shares a number of similarities to emerging data networks. For example, both electricity and bandwidth are perishable goods, and both networks experience significant load differences between peak and off-peak hours. In the electricity market, “controlled service degradation” based on interruptible service incentive contracts have been used as a product differentiator for a number of years now. For example, ConEdison in New York offers four energy management programs to large-use customers with financial incentives for voluntary load reduction¹. This control-based incentive technique has proven very effective in reducing congestion and the spot price of wholesale electricity [86]. Marketing techniques such as branding have also played an important role in differentiating electricity as a commodity item. In Europe, electricity generated from “green energy” (e.g., hydro, solar and wind) enjoys a higher price over electricity generated from fossil and nuclear fuels [97].

Following on from these observations it is likely that future bandwidth service models, mechanisms, and policies capable of providing a foundation for service differentiation will also be driven by a complex mix of technical and non-technical concerns. For example, network QOS mechanisms will need to provide flexible interfaces to support diverse service differentiation needs (e.g., supporting measures such as controlled service degradation). Engineering network QOS, however, will likely not be the sole concern of future bandwidth service management mechanisms. Rather, mechanisms will need to be carefully designed to avoid interfering with

¹http://www.coned.com/sales/business/bus_econ_develop.htm

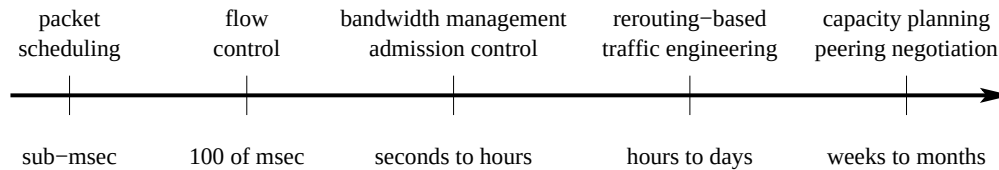


Figure 1-1: QOS Mechanisms over Different Time-Scales

other service differentiating non-technical and business drivers (e.g., pricing). A good example of such interference is congestion pricing, which is technically sound but practically infeasible because it entangles congestion mitigation techniques with largely business concerns such as service charging.

Such “tussles” in mechanism design are described by Clark et. al. in [25], where “different stakeholders that are part of the Internet milieu have interests that may be adverse to each other, and these parties each vie to favor their particular interests”. Two design principles to deal with tussles have been proposed as part of the NewArch Project²: (i) to modularize the design along tussle boundaries, so that one tussle does not spill over and distort unrelated issues; and (ii) to design for choice, to permit the different players to express their preferences. In this thesis, we focus on the tussle space surrounding the realization of bandwidth service management for the Internet and its wireless extensions.

The bandwidth service management mechanisms that we study in this thesis all operate at the medium time-scale control point (i.e., seconds to hours), as illustrated in Figure 1-1. We argue in this thesis that this control point best bridges the gap between packet level operations and the diverse needs of applications. At the packet-level, QOS techniques such as packet scheduling, queue management, traffic shaping, and flow control have been the concern of networking research community for over a decade. These techniques operate on fast time-scales (i.e., sub-second time-scales) as shown in Figure 1-1. Standardization efforts best exemplified by

²<http://www.isi.edu/newarch>

the work of IETF DiffServ Working Group [35] are addressing the inter-operability issues of these mechanisms in support of global differentiated services. Over slower time-scales (i.e., hours to months), traffic engineering techniques including rerouting and capacity planning are used for network management and planning. Over medium time-scales (i.e., seconds to minutes), network resource allocation policies are solely based on admission control techniques. However, because admission control by definition only imposes control at session setup time, while bandwidth service management for ongoing sessions has not been thoroughly investigated in the literature, as a result, service differentiation is not maintained during periods of device failure and severe congestion (e.g., resulted from the flash crowd effect). In this thesis we study models and mechanisms that can be effective at this medium time-scale (as shown Figure 1-1) in support of service differentiation. We conjecture that this point of control and time-scale offers the best Application Programming Interfaces (APIs) for service creation and bandwidth service management. Furthermore, we argue that this control point and time-scale will not impact fast time-scale packet forwarding operations but will provide a suitable management interface to important parameters to influence traffic control (e.g., service weights for a weighted fair queueing scheduler, or dropping threshold for a buffer management algorithm, etc.).

The problem of engineering solutions for service differentiation for the Internet core and edge-based networks is broad, multi-dimensional and challenging. In this thesis, we address three specific challenges within the context of this broader problem. We consider these challenges the more pressing ones that require significant advances in research to better judge the overall feasibility of the general approach we advocate. These challenges follow on from the discussion above on degradation control, dynamic provisioning and creating incentives for differentiated services, and comprise:

- The controlled service degradation problem, which relates to the condition where under a service differentiation regime bandwidth becomes limited and time-varying. We study how applications can respond under such conditions and how networks can engineer solutions that can take account of the application's ability to adapt to such changes. This problem is particularly pressing in the design of edge-based IP networks. We focus our attention on edge-based wireless networks, which need to be responsive to changes in available bandwidth (e.g., due to overloading, persistent congestion, addition of new sessions due to handoff), and applications that are agile to such changes. This problem raises a number of questions. How do we capture the agility of applications? How do we take control and management actions based on this knowledge? What are the best optimization policies for such networks in the face of strongly time-varying bandwidth supply and demand?
- The quantitative traffic aggregates problem, which relates to provisioning quantitative differentiated services when there is insufficient control information and fast time-scale control mechanisms available. This problem is acute in the design of core IP networks. We study scalable bandwidth management techniques for traffic aggregates that are capable of delivering quantitative differentiated services between ingress and egress points. This problem raises a number of questions. The most pressing question being: how do we manage quantitative differentiation for traffic aggregates (usually point to multipoint aggregates) when the time-scale for control is coarse and state information is limited?
- The incentive engineering problem, which concerns the stable operation of differentiated service offerings. We conjecture in this thesis that without a suitable built-in incentive structure the deployment of differentiated services

may result in the “tragedy of the commons” phenomenon [46]. Under such conditions lower priority packets take advantage of service differentiation by transiting their packets using higher priority service classes. We address this arbitrage problem by studying incentive engineering techniques for edge-based wireless networks that create incentives for mobile users to truthfully self-differentiate their service needs based on their application needs. This problem raises a number of important issues. How do we support service differentiation when there is a lack of user cooperation? How do we design market-based mechanisms that promote suitable user strategies and the stable operation of bandwidth management systems? How do we create incentives for users to truthfully self-differentiate based on their application needs? We consider incentive engineering within the context of edge-based wireless access networks, which presents a highly dynamic problem space. We investigate incentive engineering techniques that eliminate arbitrage in service differentiation based networks.

1.2. Technical Barriers

In what follows, we discuss the specific technical barriers to solving the problems discussed above.

1.2.1 Application-Aware Controlled Service Degradation

Controlled service degradation is an important component of service management for networks. It reflects a paradigm shift away from hard QOS guarantees toward soft and adaptive QOS assurances. We study mechanisms and models for controlled service degradation in the context of edge-based wireless networks where delivering hard QOS is not feasible for a number of reasons. Physical layer impairments

(e.g., co-channel interference, hidden terminals, path-loss, fast fading and shadowing) contribute toward time-varying error characteristics and time-varying channel capacity making the delivery of hard QOS guarantees unlikely. In addition, user mobility can trigger rapid degradation in the delivered service quality (e.g., during handoff). These operating conditions result in the delivery of time-varying QOS to mobile applications.

The ability of mobile applications to adapt to such changes while keeping the user's perceptible quality meaningful is challenging. Existing mobile networks (e.g., Mobile IP and 3G cellular systems) lack the architectural flexibility to accommodate application-specific adaptation needs in time-varying environments. Those networks and models that support network QOS mechanisms often rely on end-systems to declare their QOS requirements such as bandwidth, delay, and delay jitter. This approach leads to frequent renegotiation between end-systems and the network during periods of change resulting in poor scalability when the number of flows or traffic aggregates³ grow or when adaptation becomes more frequent (e.g., as in the case of wireless mobile networks). As a result, there is a need to design efficient network adaptation techniques to support controlled service degradation. Unlike end-system oriented approaches, however, network-based adaptation is intrinsically more complex, presenting a number of challenges:

- How do we model an application's adaptive QOS demands, and furthermore, over what time-scale?
- How do we formulate bandwidth allocation algorithms for controlled service degradation that maximize utilization, fairness, and differentiation?

³The terms flow and traffic aggregate are used synonymously in this thesis. Both refer to packets satisfying the same packet header classification rules and sharing the same routing path in the network.

- How do we design these bandwidth allocation algorithms to operate in networks with multiple bottlenecks, and at the same time support efficient state management and application dynamics such as convergence and allocation stability?

1.2.2 Quantitative Service Differentiation for Traffic Aggregates

Detailed control information (e.g., per-flow states) and supporting control mechanisms (e.g., per-flow queueing) are not practical in the design of core networks in order to gain architectural scalability. Consequently, the resulting level of service differentiation between service classes is often *qualitative* in nature. However, network practitioners have to use *quantitative* provisioning rules to automatically engineer a network that experiences persistent congestion or device failure while attempting to maintain service differentiation [98, 84]. This presents a number of challenges for the emerging differentiated services Internet. For example, there is a need to develop solutions that can deliver quantitative differentiated services with suitable network control granularity, and scalable and efficient network state management.

We conjecture that a more dynamic form of provisioning is needed to compensate for the coarser-grained state information and the lack of network controllability, if, service differentiation is to be effectively realized. However, unlike traditional telecommunication networks, where traffic characteristics are well understood and well controlled, and long-term capacity planning can be effectively applied, Internet traffic is more diverse and bursty, often exhibiting long range dependence [107]. As a result, there is a need to design measurement-based dynamic control algorithms that can perform well under diverse traffic conditions. Another important challenge facing bandwidth management is the complexity associated with the rate control of traffic aggregates in core networks which may comprise of flows exiting at different

network egress points. This problem occurs when ingress rate control can only be exerted on a per traffic aggregate basis, (i.e., at the root of a traffic aggregate’s point-to-multipoint distribution tree). Under such conditions, any rate reduction of an aggregate would penalizes traffic flowing along branches of the tree that are not congested.

1.2.3 Creating Incentives for Service Differentiation

Service differentiation has an intrinsic arbitrage problem if left unchecked where packets switch from the lower service class to be sent using a better service class. Consider two service classes, one designed to offer stable allocation for real-time applications such as streaming video, and the other designed for best effort allocation for bursty data applications such as web transactions. Without an incentive structure, the stable allocation service could be “overrun” by data applications. An obvious solution to this arbitrage problem is monetary pricing. For example, congestion pricing [74, 58, 72] is considered superior in theory for distributed implementation of optimal bandwidth allocation, where the price is the Lagrange multiplier of the underlying resource optimization problem. Unfortunately, monetary congestion pricing is not practical because it becomes entangled with the service charge which is largely business driven - creating an unwanted tussle. It also violates users’ preference on pricing simplicity, stability, and predictability [81, 38, 5]. As a result, the prevailing charging model for network service is block-rate charging, which comprises a fixed charge for usage within a block of usage time or bytes delivered, and a higher flat rate for any usage that exceeds the block amount. However, this type of charging model is vulnerable to arbitrage situations too. Therefore, there is a need to develop new incentive mechanisms that on the one hand do not interfere with monetary service charging, and on the other, eliminates potential arbitrage in

differentiated service-based networks.

1.3. Thesis Outline

This thesis proposes new approaches for bandwidth service management for the Internet and its wireless extensions that include (i) utility-based adaptation mechanisms to capture applications needs and to address the technical challenges of controlled service degradation; (ii) dynamic provisioning for core networks that resolves the technical issues associated with managing complex traffic aggregates and supporting quantitative differentiated services; and (iii) incentive engineering to deal with the arbitrage issues that emerge in differentiated services models. We take a systems approach to these problems and investigate new policy modeling, algorithms and protocol design techniques. We evaluate the proposed protocols, mechanisms, and policies using a combination of analysis, simulation, and results from an experimental testbed. The outline of our study is as follows.

1.3.1 Utility Function Formulation

Our investigation of bandwidth service management is founded on bandwidth utility functions, which are capable of capturing the intrinsic adaptability of applications to bandwidth changes. Bandwidth utility functions have been widely discussed in the signal processing community in the form of rate distortion functions [92, 9, 82] for lossy video coding. However, their use in the networking community has remained largely on an abstract level [93, 18] or as a model to formulate the network effect of TCP congestion control algorithms [60, 63]. There is a need for better coordination between the utility generation methods used by the signal processing and networking communities; that is, to unify the difference between utility *measurement* and utility *formulation*. Another challenge related to video utility measurement is the

lack of online utility measurement methods that can operate at network adaptation time-scales. Previous work on video utility measurement is based on off-line procedures [87, 66, 106, 65]. Online utility measurement methods [15], on the other hand, generate utility functions over very short time intervals (e.g., tens of milliseconds for each video frame) due to the scene changes associated with video flows. Network adaptation, however, typically operates over much longer intervals potentially in the order of seconds or even minutes. This is a product of the network signaling system efficiency, the need for stable allocation by the resource management system, and the round trip delay between source encoders and receiver decoders. In Chapter 2, we take a cross-disciplinary approach to utility generation and propose a unified formulation of bandwidth utility functions for application aggregates including TCP and small audio flows, and individual video flows. We discuss experiments using the online generation of utility functions from video traces and present a utility prediction algorithm that addresses the time-scale mismatch between video content changes and network adaptation speed. The self-adaptive algorithm discussed in this chapter predicts a utility function that tightly tracks variations in video content over the network adaptation time-scale while minimizing incidences of prediction error.

1.3.2 Utility-Based Link Allocation Algorithms

There is a lack of utility-based bandwidth management algorithms that are efficient and flexible enough to realize different service differentiation policies. These policies may include equalizing the utility of all applications, or differentiating utility by service class, or maximizing the total utility. Previous work on utility maximization algorithms (e.g., the Q-RAM [85] framework of maximizing total utility) are known to be computationally intensive (i.e., NP-hard), while the work on fair or differentiated allocation [47, 102] has been bandwidth-based rather than utility-based. In

Chapter 3, we present two groups of utility-based link allocation algorithms that provide a foundation for utility differentiating and utility maximizing bandwidth management. The utility maximizing algorithm discussed in this chapter leverages the piecewise linear quantization of utility functions and uses the Kuhn-Tucker [62] condition to significantly reduce the algorithm execution time. Our utility differentiating algorithm supports utility fair allocation that allows individual utility functions to have different maximum utility values. In addition, we present a hierarchical structure that augments the Class Based Queueing (CBQ [40]) algorithm to support a combination of the proposed utility-based bandwidth management policies.

1.3.3 Utility-Based Adaptation in Wireless Access Networks

Distributed bandwidth management schemes are more complex than the case of a single link (as discussed in Chapter 3) because one flow's allocation can be affected by other flows sharing a portion of a multi-hop route. In Chapter 4, we extend our study in two directions: (i) from a single link to multi-hop network with multiple bottlenecks; and (ii) augment utility functions with additional properties such as convergence and allocation stability. *Max-min* fairness [52] is the most widely used fairness criterion found in bandwidth allocation algorithms for networks. Here, the idea is to maximize the allocation of flows with the least allocation; that is, to allow a flow to increase its allocation provided that the increase does not subsequently cause a decrease in the allocation of a flow holding a lower or equal bandwidth allocation [10]. We augment the max-min flow control algorithm with a new utility-based max-min fair allocation scheme, and design a renegotiation protocol to improve the convergence speed and to reduce protocol state management. We investigate the performance of the algorithm as part of a mobile network adaptation architecture comprising a split level adaptation control framework that operates at the network

and application levels. The network level control realizes a distributed implementation of the utility-based max-min fair bandwidth allocation. The application level control is managed by a set of distributed adaptation handlers that operate at mobile devices realizing application-specific adaptation strategies. Our network-based approach to realizing adaptation policies differs from end-system oriented approaches found in the literature [79, 41]. We demonstrate that our utility-based adaptation framework is a more generic than end-system oriented approaches because all applications can benefit. In addition, our approach is more efficient because we reduce the reliance on the real time signaling of application resource requirements by using utility functions to model a range of application requirements in advance.

1.3.4 Quantitative Service Differentiation for Traffic Aggregates in Core Networks

In Chapter 5, we shift our focus from edge-based networks to core networks. In a core network, per-flow states such as utility functions are kept at the edge of the core network. In this case, the challenge becomes gaining effective control with coarse granularity control information resulting from the aggregation of flow states and control mechanisms. Most of current quantitative differentiated service results are based on packet scheduling [95, 96, 27, 71, 48] and admission control techniques [21, 17, 44, 59]. In contrast, we present a dynamic bandwidth provisioning framework for quantitative service differentiation. Our scheme comprises a pair of node and core provisioning algorithms. The node provisioning algorithm prevents transient violations of service level agreements by predicting the onset of service level violations based on a multi-class virtual queue technique [44, 59], self-adjusting per-class service weights and packet dropping thresholds at core routers. Persistent service level violations are reported to a dynamic core provisioning algorithm, which

dimensions traffic aggregates at the network ingress taking into account fairness issues not only across different traffic aggregates but also within the same aggregate whose packets can take different routes in the core IP network. We solve the problem of rate regulation for point-to-multipoint flow aggregates with the use of matrix inverse operations. We demonstrate that our model is capable of delivering capacity provisioning in an efficient manner and providing quantitative delay-bounds with differentiated loss across per-aggregate service classes.

1.3.5 Incentive Engineering for Service Differentiation in Wireless Access Networks

In Chapter 6 we address the incentive compatibility issue to remove arbitrage that are intrinsic to service differentiation. We focus on edge-based wireless extensions to the Internet based on wireless LAN technology. In this chapter, we apply incentive engineering techniques [28] (also known as mechanism design in game theory [103, 43]) without the need of monetary service charge. The scheme supports two incentive-based allocation service classes: instantaneous allocation and stable allocation, which are designed to support data and real-time applications, respectively. The algorithms effectively constrain the strategy space of subscribers to a set of cooperative behaviors that include the truthful selection of service class and truthful declaration of bandwidth demands, which avoid the convergence delay that is common to repeated game strategies [43]. Our design minimizes protocol messaging overhead imposed on wireless subscribers while possessing a number of beneficial properties including Nash bargaining fairness [109, 78] for the instantaneous allocation service, and incentive compatibility for mobile users promoting the truthful declaration of service preferences and bandwidth demands.

1.4. Thesis Contribution

The major contributions of this thesis can be summarized as follows:

- We propose the first unified framework of bandwidth utility functions for both multimedia and data applications. We resolve the time-scale mismatch between the generation of video utility functions and network adaptation by designing a self-adaptive utility prediction algorithm.
- We design the first algorithm that is able to integrate and support in a hierarchical and efficient manner diverse utility-based allocation policies including utility-fair, utility-differentiated and utility-maximizing allocations.
- We present the first design of adaptation control for wireless/mobile networks that supports both network and application level adaptation requirements. The network protocol and mechanisms are built on utility-based max-min fair bandwidth allocation based on an extension of the max-min flow control algorithm. The application level design supports application specific adaptation constraints based on time-scale and bandwidth granularity.
- Our DiffServ provisioning work is the first to demonstrate quantitative service differentiation for traffic aggregates. The state of the art at the time was either quantitative performance guarantee for individual flow (Stoica's CSFQ work [95, 96]) with the packet header carrying the control state or qualitative differentiation for traffic aggregates (Dovrolis' work of proportional DiffServ [27]). We realize this by dynamically changing the service weights of an off-the-shelf per-class weighted scheduler, with an extended virtual queue technique [44, 59] to predict traffic overloads under bursty and self-similar multi-class traffic conditions.

- Our core provisioning work is unique because we are the first to identify a potential problem in edge provisioning of traffic aggregates. We formulate the problem as a flow control problem where some branches of the point-to-multipoint trees are congested but the control points are at the root of these trees. We present optimum solutions for fairness across multiple trees, fairness within a tree, and a combination of both.
- We present the first incentive-compatible service model and mechanisms that supports stabilizing bandwidth allocation needed by multimedia applications. Our design possesses a number of beneficial properties including minimizing the algorithmic and protocol overhead on mobile devices, Nash bargaining fairness, and incentive compatibility for mobile users promoting the truthful selection of service classes and bandwidth declarations. The system is implemented over *wincent* (for wireless LAN incentive engineering testbed), whose source code is available from the Web⁴.

⁴*wincent* testbed open source code can be downloaded from <http://www.comet.columbia.edu/cubanet/wincent/>.

Chapter 2

Utility Function Formulation: A Unifying Abstraction for Networks, Applications and Content

2.1. Introduction

Multimedia systems that deliver content across the Internet and its wireless access networks often encounter congested links shared by other multimedia flows and contending best-effort TCP data traffic. Conventional solutions to this problem redistribute congested bandwidth among contending flows in a manner that is typically application and service unaware. With advances in the packet processing capability of networking equipment (e.g., gateways, switches, routers and base stations), more sophisticated and effective bandwidth allocation algorithms become viable at the network congestion points (e.g., wireless links).

2.1.1 Utility-based Bandwidth Allocation Framework

In the first part of the thesis (viz. Chapters 2-4), we study a unified approach capable of supporting both application-aware and service-differentiated bandwidth allocation for Internet core and edge-based access networks, with particular em-

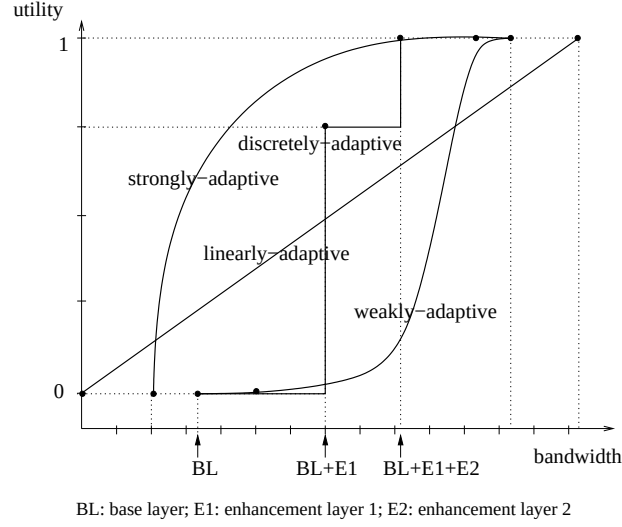


Figure 2-1: Different Styles of Utility Functions

phasis on emerging wireless and mobile edge-based access networks. The approach presented in this chapter is based on bandwidth utility functions, which model application, and therefore indirectly end-user relative preferences in relation to network bandwidth. In our study, we formulate utility functions as a unifying abstraction to solve network allocation, application adaptation and content delivery problems in a synergistic manner. Utility functions can represent a wide variety of application adaptation behavior, as illustrated in Figure 2-1. The quality index of a utility function refers to the level of satisfaction perceived by an adaptive mobile application. Concave utility functions represent strongly adaptive applications that are not sensitive to bandwidth changes when the bandwidth allocation is close to the maximum requirement. TCP represents an example of such an adaptive application. In contrast, convex utility functions represent weakly adaptive applications that are sensitive to bandwidth changes when the bandwidth allocation approaches the maximum requirement. Some video applications exhibit this behavior. Linear utility functions model the case of equal bandwidth adjustment regardless of the original bandwidth. Hence, linear utility functions are well suited to represent data

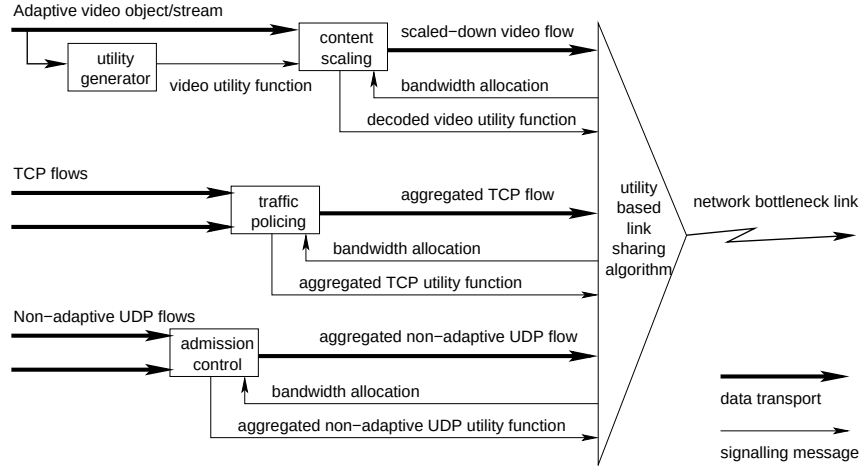


Figure 2-2: Utility-Based Bandwidth Allocation Framework

applications that are insensitive to bandwidth variation over any particular range of bandwidth allocation. Other types of utility functions include discrete curves (e.g., step or staircase shaped curves) that model discretely adaptive applications (e.g., multi-layered MPEG video flows).

Figure 2-2 shows our proposed framework of utility-based bandwidth allocation. The framework comprises utility function generating and utility-based link sharing modules. The utility function generating module is designed to process adaptive video flows, TCP aggregate flows, and non-adaptive UDP aggregate flows¹, respectively. Based on the generated utility functions, the utility-based link sharing module distributes bottleneck bandwidth among contending (aggregated) flows to exploit applications' adaptability to bandwidth changes.

For adaptive video flows, their intrinsic scalability is exploited by the design of two components: a utility generator and a content scaler. The utility generator creates bandwidth utility functions on-demand (e.g., using the method to be

¹These flows can be identified through explicit signalling protocols (e.g., SIP [34] or RTSP [33]) and packet header processing.

discussed in Section 2.3.). The resulting instantaneous utility function is then processed by a utility prediction algorithm (to be discussed in Section 2.4.) to keep the predicted utility functions meaningful over network adaptation time-scales. The content scaler performs content-based media scaling to rate control outgoing multimedia traffic based on the network allocated bandwidth. The content scaler also forwards predicted utility functions to the link sharing module to make resource allocations. The utility generator and content scaler modules can be flexibly deployed based on the type of network. For example, in wireless networks, the content scaler could be placed close to base station where it can quickly react to wireless link dynamics. The utility generator could be located at the video encoder to best reuse coding information. In another example scenario that is capable of supporting a video-on-demand multicasting service, both the utility generator and content scaler could be placed at video gateways to scale up/down video based on the needs of individual receivers.

The equivalent of content scaler for TCP aggregates is the traffic policing module. It maintains a shared buffer where member TCP flows are queued and served. Packet arrivals exceeding the allocated bandwidth are dropped to invoke the TCP flow control mechanism. For non-adaptive UDP flows like voice applications, their individual bandwidth consumption is small but sensitive to bandwidth changes. Therefore, rather than dropping packets of all member flows within an aggregate, bandwidth reduction is done through the admission control module to drop ongoing calls and block new call requests. As a result, the utility function for non-adaptive UDP aggregates will model the impact of call dropping during congestion.

2.1.2 Utility Formulation

The work on formulating bandwidth utility functions can be categorized into two distinct camps, one used by the signal processing community and the other by the networking research community. The work from signal processing largely focuses on utility measurement and utility-based end-to-end rate control; while the work from networking research focuses on modeling TCP applications with utility functions and utility-based bandwidth allocation algorithms. The study presented in this chapter helps bridge the gap between these two schools of thought. One might conjecture that if bandwidth utility functions can be similarly formulated for adaptive multimedia applications, the same distributed control framework could be applied to non-TCP-like adaptive applications to improve the overall system utility which would better support fairness issues such as TCP-friendliness. Unfortunately, unlike TCP where a generic utility function could be synthesized from a limited set of TCP congestion avoidance algorithms, a utility function for compressed video is content and encoder dependent with no generic functional form. In our study, we present a new formulation of utility functions for TCP applications to capture the same effect of rate-distortion that is individually measured for video applications. This makes the utility metrics of both formulated and measured utility functions compatible, and hence leads to a unified adaptation control scheme. The unified utility function formulation that we detail in this chapter closes the control loop between network and application based adaptations. As a result, application-awareness can be modelled with utility functions generated from multimedia content (e.g., video frames), while service-differentiation is achieved by scaling utility functions with respect to network service classes.

This chapter presents the first cross-disciplinary attempt to our knowledge that bridges the gap between image processing and network adaptation in order to bet-

ter support multimedia adaptations. The contribution of this chapter is as follows. We present a systematic formulation of bandwidth utility metrics. We formulate utility functions based on application and service types for aggregates of TCP flows, small multimedia flows, and large video flows. We analyzed the trade-off between utility generation (i.e., the online dynamic generation of utility functions) and network adaptation time-scales. In particular, we identify the time-scale mismatch problem where the utility generation interval (i.e., the time before which we need to generate another utility function) is usually orders of magnitude faster than the bandwidth renegotiation time-scale in networks (which is bounded by the end-to-end delay). In order to extend the utility generation interval to match the network renegotiation time-scale, we proposed a self-adaptive algorithm capable of dynamically adjusting any bandwidth over-estimation in utility measurements over prolonged utility generation intervals. Our results verify the effectiveness of the utility prediction algorithm presented in this chapter, which looks particularly promising for constant rate encoded video because we found that the predicted bandwidth vector envelopes are insensitive to scene changes.

The structure of this chapter is as follows. In Section 2.2. we survey the related work. In Section 2.3. we present the formulation of utility functions for application aggregates including TCP and small audio flows, and individual video flows. This is followed in Section 2.4. by our utility prediction algorithm that addresses the time-scale mismatch problem. We evaluate our utility prediction algorithm using video traces in Section 2.5. Finally, we present some concluding remarks in Section 2.6. In addition, in Appendix A. we calculate the optimal value of the utility generation interval and identify its dependency on both content and network parameters.

2.2. Related Work

The performance of distributed multimedia content (e.g., video streams) can be improved by exploiting the intrinsic scalability of content through rate control techniques coupled with effective media scaling [110] and content-aware periodic bandwidth renegotiation [14]. These techniques are well suited toward transporting and scaling video content in response to time-varying bandwidth availability typically found in the Internet, and more characteristically, in wireless and mobile networks. There has been a significant amount of research on video utility functions in the form of rate-distortion function [92, 9] from the signal processing community. Here the focus of the research has been on video compression and rate-control for the design of encoders. See Ortega and Ramchandran [82] for an extensive survey of this topic. This is in contrast to the networking community, where there has been little work in the area of dynamic generation of utility functions suitable for network resource management. In [87], Reininger constructs utility functions from experimental data based on the 5-level mean-opinion-score (MOS) [51] test for subjective video quality. However, this reported work is based on an off-line process for subjective quality testing. Even though there have been some recent developments in objective measurement techniques that model the human visual system [106, 65], these schemes are computationally intensive and involves decoding and buffering delay. In [15], a machine learning technique is used to estimate a utility function in order to speed up the utility generation process.

The networking research community considers utility functions as an abstraction of a user's preference in the macroscopic analysis of the Internet [93, 18]. For example, Breslau and Shenker [18] use several types of bandwidth utility functions to investigate the merits of bandwidth reservation for adaptive and rigid applications. Recently, there has been a large amount of work on modeling the TCP congestion

control mechanism with bandwidth utility functions. Various forms of utility functions are proposed, (e.g., $\log(r)$ in [60] and $(-1/r)$ in [63], where r denotes the transmission rate). These utility functions, however, are not measured from the effect of bandwidth reduction on TCP applications. Rather, they are specifically formulated to model the property of TCP congestion avoidance algorithm (e.g., proportionally fair [60] with $\log(r)$) on bandwidth allocation for network flows. Clearly, this modeling approach differs significantly from the measurement methodology for video utility functions.

2.3. Utility Formulation

2.3.1 Utility Metrics

We do not use utility functions to model a user's monetary valuation of bandwidth. Rather, bandwidth utility functions solely model applications' relative bandwidth preferences, where absolute utility values bear no meaning to applications. Therefore a normalized utility function can completely capture an application's preferences. We denote $v_i(r)$ the normalized utility function for the i th flow.

Limiting utility formulation to normalized utility functions provides a great deal of flexibility in allowing network algorithms to scale utility values enabling the implementation of differentiated allocations that reflect a user's service plan and traffic class. Network algorithms can use the scaled utility function $u_{i,\max}v_i(r)$ to calculate bandwidth allocation, where $u_{i,\max}$ is dependent on the service plan and traffic class of flow i . As we will show in Chapter 3, under a utility maximization allocation, changing $u_{i,\max}$ is similar to assigning an allocation priority, while under a proportional utility differentiation allocation, utility scaling can realize allocations that can, for example, give one user twice the normalized utility value in comparison to others.

The normalized utility function $v_i(r)$ is an increasing function with bandwidth r in the range of $[r_{i,\min} \ r_{i,\max}]$, where $v_i(r_{i,\min}) = 0$ and $v_i(r_{i,\max}) = 1$. Here $r_{i,\min}$ is called inelastic demand, which represents an application's minimum bandwidth requirement, which must be always guaranteed once accepted by the admission control procedure. The remaining bandwidth demand $(r - r_{i,\min})$, in contrast, is called elastic demand, which represents an application's additional bandwidth requirement, which is adaptive, and therefore, can be adjusted by utility-based allocation algorithms during network congestion. To simplify algorithmic design, we rewrite $v_i(r)$ as $v_i(x + r_{i,\min})$, (i.e., as a function of x in the bandwidth range of $[0 \ r_{i,\max} - r_{i,\min}]$). This bandwidth offset operation represents a separate treatment of inelastic and elastic demands with utility-based allocation algorithms supporting elastic demands and admission control algorithms (not covered in this work) handling the inelastic demands. Note, that the admission control of inelastic traffic is trivial because the inelastic minimum bandwidth demand can be treated the same as constant bit rate traffic.

The scaled utility function used by the network has the form:

$$u_i(x) = u_{i,\max} v_i(x + r_{i,\min}), \quad (2.1)$$

$$\text{where } x \in [0 \ b_{i,\max}] \ b_{i,\max} \triangleq r_{i,\max} - r_{i,\min}.$$

It is challenging to consider maintaining network states capable of storing generic forms of utility functions. A natural solution to this problem is to consider quantization techniques, as a means for state reduction. Therefore, we quantize a utility function into a continuous and monotonically increasing piecewise linear function.

We denote K_i the number of linear segments in a normalized piecewise linear utility function $v_i(x)$. The starting and ending points of the k th segment are

$(r_{i,k-1}, v_{i,k-1})$ and $(r_{i,k}, v_{i,k})$. For convenience, we use a vector representation, that is:

$$\left\langle \begin{pmatrix} v_{i,0} \\ r_{i,0} \end{pmatrix} \cdots \begin{pmatrix} v_{i,K_i} \\ r_{i,K_i} \end{pmatrix} \right\rangle. \quad (2.2)$$

We note that $r_{i,0} \geq r_{i,\min}$; $v_{i,0} \geq 0$ and $r_{i,K_i} \leq r_{i,\max}$; $v_{i,K_i} \leq 1$. These inequalities are intended to allow the quantization procedure to choose the best quantization levels within the range of $r \in [r_{i,\min}, r_{i,\max}]$ and $v \in [0, 1]$.

Similarly, for the corresponding scaled utility function $u_i(x)$, the vector representation of its piecewise linear quantization is:

$$\left\langle \begin{pmatrix} u_{i,0} \\ b_{i,0} \end{pmatrix} \cdots \begin{pmatrix} u_{i,K_i} \\ b_{i,K_i} \end{pmatrix} \right\rangle, \quad (2.3)$$

where

$$b_{i,k} = r_{i,k} - r_{i,0} \quad \text{and} \quad u_{i,k} = u_{i,\max} \frac{v_{i,k} - v_{i,0}}{v_{i,K_i} - v_{i,0}}. \quad (2.4)$$

In particular, $b_{i,0} = 0$; $u_{i,0} = 0$ and $b_{i,K_i} = r_{i,K_i} - r_{i,0}$; $u_{i,K_i} = u_{i,\max}$.

In summary, we formulate two piecewise linear utility function metrics. The normalized utility function $v_i(r)$ is used for formulating and measuring utility functions of different types of applications, as discussed in Section 2.3.2 and 2.3.3. In contrast, the scaled utility function $u_i(x)$ is used for implementing network control policies capable of differentiating bandwidth allocation with respect to service types and traffic classes, as discussed in Chapter 3.

2.3.2 Utility Generation for Adaptive Video

2.3.2.1 Video Utility Measurement

For UDP based adaptive video applications, the utility value represents a valuation of the loss of content information due to lossy rate-controlled encoding. Therefore, the utility metrics under this setting represents the video quality metrics. The theoretical foundation of video quality metrics is based on a distortion-rate function $D(r)$ [9], where r is the scaled-down rate of the information source and $D()$ is the distortion which is measured by the “minimum distance” between the original data and scaled-down data whose rate is less than or equal to r . Given the distortion measure $D(r)$, the normalized utility value $v(r)$ is given as:

$$v(r) = 1 - D(r)/D(0). \quad (2.5)$$

Here $D(0)$, the maximum distortion when the transmission rate is zero, is also the total amount of information in original data source. For better clarity, we omit the subscript i in $v_i(r)$ and $u_i(x)$, etc. in the remaining part of this chapter.

Ideally, the distortion measure should be based on perceptual video quality metrics. However, finding a general enough, not to mention easily computed, measure of perceptual quality has proven to be an elusive goal. Since the generation of a video utility function is content dependent, subjective quality metrics requiring off-line human intervention are not viable. Thus, in practice, utility metrics have to be objective, based on the peak signal-to-noise ratio (PSNR) or correlated in a simple form to subjective tests of the human visual system. It is worth noting that in some cases, such as the current JPEG 2000 standard, encoders designed to minimize the PSNR metrics have given excellent results in perceptual tests [82]. The work reported on in this chapter adopts the PSNR metrics in support of fast processing.

By choosing $err^2(r)/sig^2$ as the distortion measure, the normalized utility metrics becomes:

$$v(r) = 1 - \frac{err^2(r)}{sig^2}, \quad (2.6)$$

$$\text{where } sig^2 = \frac{1}{N} \sum_{j=1}^N x_j^2 \text{ and } err^2(r) = \frac{1}{N} \sum_{j=1}^N (x_j - y_j(r))^2. \quad (2.7)$$

The sig^2 component represents the mean energy each pixel has in the original picture and err^2 the mean square error between the distorted image and the original image. N is the number of pixels in the picture, x_j and y_j are the j th pixel in the original and distorted images, respectively. $err^2(r)$ is non-increasing as r increases in $[r_{\min}, r_{\max}]$, where $err^2(r_{\min}) = sig^2$ and $err^2(r_{\max}) = 0$. Therefore, $v(r)$ is non-decreasing in the range of $[0, 1]$ as r increases in $[r_{\min}, r_{\max}]$.

In general, the calculation of Equation (2.6) depends on the coding scheme. All of the recent coding standards (e.g., JPEG, JPEG 2000, MPEG 1/2/4) take the same approach of transform coding that decomposes the source into frequency components using block transforms such as the discrete cosine transform (DCT) or wavelet filters [82]. Because the DCT transform matrix is unitary, the calculation of sig^2 and err^2 in (2.7) can be accomplished in DCT transform domain so that:

$$sig^2 = \frac{1}{N} \sum_{j=1}^N X_j^2 \text{ and } err^2(r) = \frac{1}{N} \sum_{j=1}^N (X_j - Y_j)^2, \quad (2.8)$$

where X_j and Y_j are the j th DCT coefficients in the original and distorted images, respectively. This property allows for the calculation of utility metrics directly from the encoded bit stream without the need for inverse-DCT transcoding. An additional benefit of the SNR metric is that it is additive, (i.e., err^2 can be calculated in sequence from one macro-block to another).

Coding standards (e.g., MPEG) using prediction or context-based coding are

by nature dependent on consecutive frames. Therefore, the per-frame distortion measure may be inaccurate. The dependency effect can be often ignored to speed up computation with little performance loss, as discussed in [82]. We choose the distortion metrics based on one group of pictures (GOP) to discount the dependency between I, B, and P frames within a GOP. The resulting per-GOP utility function becomes the “instantaneous utility function”.

Measuring utility functions requires sampling the video quality distortion function across its entire rate range $[r_{\min} \ r_{\max}]$. This procedure can be computationally intensive if fine-granularity sampling is used. In the case of instantaneous video utility functions, we choose $K = 4$, the same scale used by the 5-level MOS test [51]. The corresponding 4-segment piecewise-linear utility function is obtained by calculating its break points (i.e., the first-order discontinuity points) from interpolation of the rate-distortion samples. We choose the break points such that their normalized utility values are equally spaced in $[0 \ 1]$ ².

2.3.2.2 Video Scaling Profile

One state management issue associated with video utility functions is that there are a wide variety of media scaling techniques that could be used to scale (i.e., up/down) video content. These media scaling techniques include:

- *spatial resolution scaling*, which changes picture size (e.g., from CIF to QCIF, requiring transcoding);
- *temporal domain scaling*, which drops frames;

²Other choices of discrete utility values can also be used. For example, one can set the minimum utility v_0 to a value based on the relationship between the PSNR and subjective video quality metrics, or choose discrete levels with the least square of quantization error. Because these more complex choices do not affect the subsequent development of algorithms, we do not pursue them further in this work.

- *quality scaling*, which changes quantization levels, chrominance dropping, Discrete Cosine Transform (DCT) and Discrete Wavelet Transform coefficients dropping; and
- *content-based scaling*, which uses MPEG-4 video object prioritization and dropping.

Because a video content scaler may be not co-located with a utility function generating module, it is necessary to ensure that a content scaler will select the same combination of scaling techniques used by a utility generator when reducing the data rate. This is the role of the scaling profile, which keeps the content scaler's operating point on the utility function. In addition, the scaling profile cannot be fixed for all content because it is also dynamically changing with user preferences and content type. For example, in the case of a fast-motion scene, spatial resolution or quality scaling techniques are more suitable than temporal-domain scaling techniques (e.g., dropping frames) because the detail within a picture may be unimportant under fast-motion. However, slow-motion scenes may favor the opposite approach. User preferences may choose, for example, to drop chrominance for wireless devices. Therefore, the scaling profile needs to be managed together with utility functions inside network. In our current experiments, the scaling profile of a single video stream is specified as a sequence of scaling actions. Because most scaling actions generate coarse-grained rate changes that can be estimated by the number of coefficients dropped, the resulting distortion function will have a discrete drop for a scaled-down rate. In contrast, dropping transform coefficients supports fine-granularity rate changes (e.g., the dynamic rate shaping (DRS) [32] method optimally drops luminance data DCT coefficients to minimize the distortion). As a result, the derived utility function takes the shape of a concatenation of curves of either piecewise linear or step shapes. Each first-order discontinuity point (v_k, r_k)

on the utility function is augmented with one set of scaling techniques – S_k that is used to scale down video to rate r_k and achieve utility value v_k . A set S_k comprises scaling technique pointers A_j that are predefined and known to content scalers. In contrast, A_j could use uniform resource locators (URL) pointing to the scaling method implementations, allowing the content scaler to use the URL information to install the scaling method. Therefore, with the addition of a scaling profile, the state associated with a utility function becomes:

$$\left\langle \begin{pmatrix} u_1 \\ b_1 \\ S_1 \end{pmatrix} \cdots \begin{pmatrix} u_K \\ b_K \\ S_K \end{pmatrix} \right\rangle. \quad (2.9)$$

This set of state information can then be delivered to the network content scaler through “out-of-band” signalling messages or an “in-band” packet field such as the Extension Descriptor component of MPEG-4 object descriptor [1] and/or MPEG-7 content descriptor.

Figure 2-3 illustrates an example of a scaling pattern and its relation to a utility function. The scaling actions are A_2 (dropping chrominance); $A_1(30)$ (DRS to

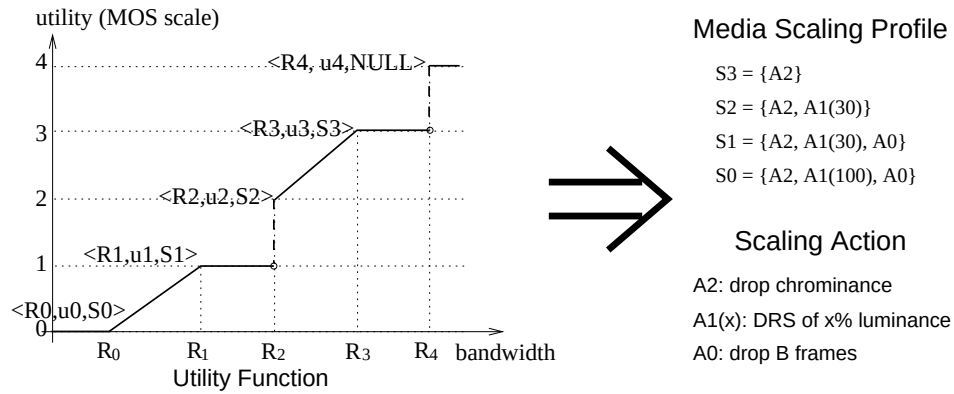


Figure 2-3: A Video Scaling Profile

drop DCT coefficients up to 30%); A_0 (dropping B frames); and $A_1(100)$ (DRS to drop DCT coefficients up to 100%). The utility function on the left comprises a concatenation of four parts, each corresponding to a different scaling patterns. For example, between utility value 4 and 3 the utility function is a step function resulting from scaling profile S_3 , which consists of one scaling action A_2 – dropping chrominance. Between utility values 3 and 2, the utility function has a linear shape approximating the scaling profile S_2 , which comprises one additional scaling action on top of S_3 – $A_1(30)$, resulting in DRS dropping up to 30%.

Remark: We note that directly using step shape utility functions in utility-based allocation leads to a complex combinatorial optimization. Therefore, in the utility-based allocation algorithms discussed in Chapter 3, segments of step shape utility functions are replaced by linear segments. The over-allocation error is corrected by the allocation protocol between the content scaler and link sharing module, (i.e., the content scaler could accept a portion of the assigned bandwidth from the link sharing module). A detailed solution is presented in Chapter 4.

2.3.3 Utility Formulation for Application Aggregates

Unlike adaptive video applications, whose utility functions are content and encoding dependent and have to be generated individually, applications like TCP can be accommodated using a generic utility function formulation. In addition, extending utility formulation from individual applications to flow aggregates is essential for a scalable utility-based solution. With aggregated utility functions, the amount of network state can be significantly reduced because the utility function of each individual flow does not need to be sent to the network.

2.3.3.1 Aggregates of TCP Applications

For TCP-like reliable transport protocols, the effect of bandwidth reduction causes no information loss. Instead, we directly model the distortion as a quadratic function of the reduction in allocated bandwidth at a target link where the utility-based link-sharing algorithm is running. That is,

$$D_{agg_TCP}(r) = (r_{\max} - r)^2, \quad (2.10)$$

where $r_{\max}$ is the maximum throughput of a TCP aggregate when the target link is not a bottleneck for this TCP aggregate (i.e., $r = \infty$). The quadratic $D(r)$ function is a convex and decreasing function to model the bandwidth elasticity of TCP [93], namely its diminishing gain in utility when bandwidth increases. Here by assuming that the allocated bandwidth is fully utilized by the TCP aggregate, we ignore the “global synchronization” among member TCP flows within an aggregate, whose effect on bandwidth utilization leads the saw-tooth shape that is typical for individual TCP flow. However, studies like [83] have shown that the pathological behavior of global synchronization does not materialize in reality due to differences in round-trip delays, host processing time, and/or the presence of Random Early Detection (RED) [39] gateways among member TCP flows.

In general, $r_{\max}$ can be measured when the target link is not a bottleneck for the TCP aggregate, i.e., when the bandwidth allocation r is larger than the measured maximum throughput of the TCP aggregate $\bar{r}_{\max}$ in an observation window. In this case, the TCP aggregate is constrained either by its senders’ slow start capability, or by the bottlenecks in the other part of the flow paths. When the target link is a bottleneck for the TCP aggregate, $r_{\max}$ is set to the link capacity. Therefore, we

have:

$$r_{\max} = \min\{\beta \bar{r}_{\max}, C\}, \quad (2.11)$$

where $\beta \geq 1$ is a multiplicative parameter to account for the under-estimation of $r_{\max}$ by the $\bar{r}_{\max}$ measurement due to limited observation window.

With Equations (2.5) and (2.10), we have the normalized utility function for TCP aggregate as:

$$v_{agg_TCP}(r) = 1 - \frac{(r_{\max} - r)^2}{(r_{\max} - 0)^2} = 1 - \left(1 - \frac{r}{r_{\max}}\right)^2. \quad (2.12)$$

This is a strictly concave function with the minimum utility at $v(0) = 0$ and the maximum utility at $v(r_{\max}) = 1$. Its marginal utility function

$$v'(r) = \frac{2}{r_{\max}} \left(1 - \frac{r}{r_{\max}}\right) \quad (2.13)$$

reaches maximum at $v'(0) = 2/r_{\max}$ and minimum at $v(r_{\max}) = 0$.

In the case of the quantized piecewise linear function, from Equation 2.12, with an equal quantization step for relative bandwidth, we have the 4-segment vector representation of the normalized TCP aggregated utility function as:

$$\left\langle \begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0.4375 \\ 0.25r_{\max} \end{pmatrix} \begin{pmatrix} 0.75 \\ 0.5r_{\max} \end{pmatrix} \begin{pmatrix} 0.9375 \\ 0.75r_{\max} \end{pmatrix} \begin{pmatrix} 1 \\ r_{\max} \end{pmatrix} \right\rangle \quad (2.14)$$

Figure 2-4 illustrates an example of normalized bandwidth utility function and its corresponding piecewise linear approximation.

Remark: With this aggregated TCP utility function formulation, under the (normalized) utility fair policy to be presented in Section 3.3.1, bandwidth allocation to each member aggregate is proportional to its $r_{i,\max}$, (i.e., $r_i = r_{agg} r_{i,\max} / \sum r_{i,\max}$,

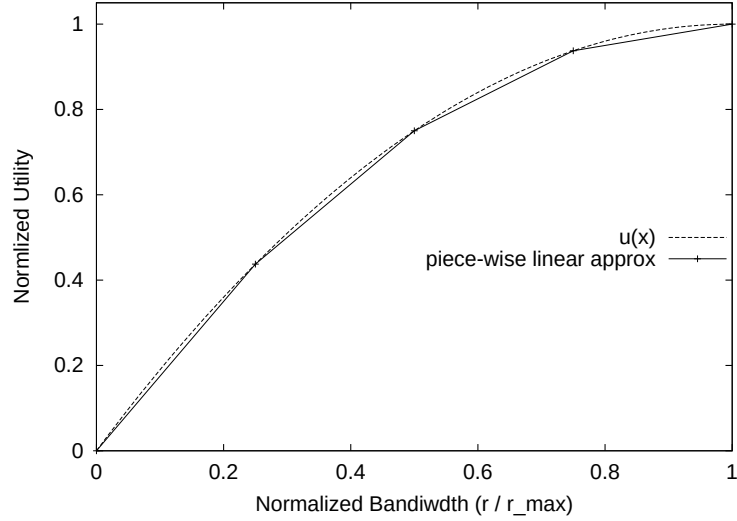


Figure 2-4: Example of Normalized $v(x)$ for TCP

where r_{agg} is the total allocation). Under the utility maximizing policy to be presented in Section 3.4., bandwidth allocation to each aggregate needs to have the same marginal benefit (i.e., $v'_i(r_i) = v'_j(r_j)$), which leads to $r_i = r_{i,\max}(1 - r_{i,\max}(\sum r_{i,\max} - r_{agg}) / \sum r_{i,\max}^2)$.

2.3.3.2 Aggregates of Small Non-Adaptive UDP Applications

For individual non-adaptive applications, the bandwidth utility function will have a convex functional form. The following proposition supports the generalization for convex utility aggregation.

Proposition 1 *For flows with convex utility functions, utility-maximizing allocation is equivalent to sequential allocation, that is, the allocation will satisfy one flow to its maximum utility before assigning available bandwidth to another flow.*

Proof: The proof is based on the well known convex analysis result for convex functions. In this case, the maximization solution lies at the extreme points of a convex

hull [89]. Since the extreme points of a convex hull of pure convex functions are at the combinations of zero or maximum allocation to each individual convex function, therefore a sequential allocation through admission control will maximize the total system utility. $\square$

We note that this is the same argument used by Breslau and Shenker in [18] for supporting bandwidth reservation for rigid applications but not for elastic applications like TCP. When a flow aggregate contains a large number of these non-adaptive applications with convex bandwidth utility functions, under utility maximization, from Proposition 1, the aggregated bandwidth utility function is a cascade of individual convex utility functions. The normalized aggregated bandwidth utility function can be approximated as a linear function between the two points $(0,0)$, and $(nr_{\max}, 1)$, where n is the number of flows, and $r_{\max}$ is the maximum required bandwidth of an individual application. In other words, we have

$$D_{agg-rigid}(r) \approx nr_{\max} - r, \forall r \in [0 \ nr_{\max}], \text{ and} \quad (2.15)$$

$$v_{agg-rigid}(r) = 1 - \frac{nr_{\max} - r}{nr_{\max} - 0} = \frac{r}{nr_{\max}}, \forall x \in [0 \ nr_{\max}]. \quad (2.16)$$

In summary, we generalize the formulation/generation of bandwidth utility functions based on the following application categories:

- for UDP-based audio/video applications with large bandwidth needs in comparison to the link capacity, utility functions are measured based on the distortion-rate function, as defined in Section 2.3.2;
- for TCP-based application aggregates, utility functions are formulated based on Equation (2.14); and finally,

- for UDP-based audio/video application aggregates, where each application consumes a small amount of bandwidth in comparison to the link capacity, utility functions are formulated based on Equation (2.16).

2.4. Utility Prediction for Video

Utility functions generated once do not remain valid over the lifetime of a video stream. Rather, utility functions are time varying due to their sensitivity to changes in video content. Network adaptation, however, operates over much longer time-scale (e.g., seconds or minutes) because network adaptation speed is limited by factors such as the delay of signaling messages and the convergence time of distributed resource allocation protocols. Therefore, reducing the network adaptation time-scale implies an overhaul of the whole network infrastructure. In order to reconcile this time-scale mismatch, a more viable approach would be to extend the utility function generation interval close to the network adaptation time-scale.

In what follows, we introduce an adaptive utility prediction algorithm. The algorithm is designed for the piecewise linear utility functions quantized by a set of discrete utility levels (i.e., $v_k, 0 \leq k \leq K$ as described in Section 2.3.2.1). Since $\{v_k\}$, the set of discrete utility levels, is predefined and fixed for all adaptive video flows in a network, a piecewise linear utility function can then be uniquely identified by its set of discrete bandwidth values (i.e., r_k) that correspond to the discrete utility levels. We denote the bandwidth vector of a quantized utility function at time t as $\mathbf{r}(t) \triangleq \langle r_0(t), \dots, r_k(t), \dots, r_K(t) \rangle$. We will use the term “predicted bandwidth vector” and “predicted utility function” synonymously in the remaining part of this chapter.

The goal of utility prediction is to find a predicted bandwidth vector $\mathbf{r}^{pred}(t)$ at

time $t = t_n$, such that

$$\mathbf{r}^{pred}(t_n) \geq \sup\{\mathbf{r}^{inst}(t) \mid \forall t \in [t_n, t_n + T)\}, \quad (2.17)$$

where $\mathbf{r}^{inst}(t)$ denotes the bandwidth vector of instantaneous utility functions, and the supreme of the set of $\mathbf{r}^{inst}(t)$ is the *bandwidth vector envelope*. The interval T is defined as the *utility generation interval* and should ideally be of the order of the network adaptation time-scale.

The prediction procedure predicts the bandwidth vector envelope of a future instantaneous utility function. By accurately predicting the envelope, all newly generated instantaneous utility functions below the bandwidth vector envelope can be omitted from the process of bandwidth renegotiation. This prediction process increases the utility generation interval over which utility functions are valid. The limitation of this approach is the potential over-allocation of bandwidth, or in some cases, the under-estimation of resources. We denote the event of under-estimation as a “prediction violation”. More specifically, a prediction violation occurs when there exists an index k and time $t_v \in [t_n, t_n + T)$, such that $r_k^{pred}(t_n) < r_k^{inst}(t_v)$, (i.e., a predicted bandwidth vector is less than the bandwidth vector of an instantaneous utility function during a utility generation interval).

Next, we describe the operation of the prediction algorithm. The prediction algorithm operates in the normal or exception modes. In the normal mode, the algorithm generates one predicted bandwidth vector every utility generation interval T . During this interval, the algorithm uses $\mathbf{r}^{inst}(t)$, the bandwidth vectors of instantaneous utility functions, to update its internal measurement $\mathbf{r}^{max}$, where $\mathbf{r}^{max} = \max\{\mathbf{r}^{max}, \mathbf{r}^{inst}(t)\}$. When the utility generation interval T expires, the algorithm decreases the value of the expanding factor e by 1% until e reaches 1, and then generates a predicted bandwidth vector as $\mathbf{r}^{pred} = e \cdot \mathbf{r}^{max}$ for the next interval

T .

When violations occur the utility predictor transitions its state machine to the exception mode. In the exception handling mode, the algorithm updates the “expanding factor” e with respect to the bandwidth level $r_k^{\max}$ of the instantaneous utility function that caused the violation. The value of e is increased by:

$$e = e + (\max_k \{r_k^{\max} / r_k^{\text{pred}}\} - 1). \quad (2.18)$$

The prediction algorithm then generates a new bandwidth vector $\mathbf{r}^{\text{pred}} = e \cdot \mathbf{r}^{\max}$ before returning to the normal mode. The pseudo-code for the prediction algorithm is given in Figure 2-5.

```

T:      utility generation interval;
e:      the expanding factor;
 $\mathbf{r}^{\text{pred}}$ : predicated bandwidth vector;
 $\mathbf{r}^{\max}$ :  maximum of the bandwidth vectors of instantaneous utility
        functions  $\mathbf{r}^{\text{inst}}$  in  $T$  period;

On arrival of  $\mathbf{r}^{\text{inst}}(t_0)$  the first instantaneous utility function, init
     $e = \text{initial\_}e$  ;
     $\mathbf{r}^{\max} = \mathbf{r}^{\text{inst}}(t_0)$  ;
     $\mathbf{r}^{\text{pred}} = e \cdot \mathbf{r}^{\max}$  ;

On arrival of  $\mathbf{r}^{\text{inst}}(t_n)$ 
     $\mathbf{r}^{\max} = \max\{\mathbf{r}^{\max}, \mathbf{r}^{\text{inst}}(t_n)\}$ ;
     $x = \max_k \{r_k^{\max} / r_k^{\text{pred}}\}$ ;
    if ( $T$  timeout OR  $x > 1$ ) {           // need to update  $\mathbf{r}^{\text{pred}}$ 
        if ( $x > 1$ ) {                       // under-estimated  $\mathbf{r}^{\text{pred}}$  by  $(x - 1)$ 
             $e += (x - 1)$ ;
        }
        else {                             // update, decrease  $e$ 
             $e = \max\{1, 0.99e\}$ ;
        }
         $\mathbf{r}^{\text{pred}} = e \cdot \mathbf{r}^{\max}$ ; send  $\mathbf{r}^{\text{pred}}$  to network;
         $\mathbf{r}^{\max} = \mathbf{0}$  ;
    }

```

Figure 2-5: Pseudo-code of Long-range Utility Prediction Algorithm

It should be noted that the proposed prediction algorithm is designed specifically to be self-adaptive to various video content without any external control parameters. In the algorithm shown in Figure 2-5, only the initial value of e has to be externally set. In Section 2.5.2, we demonstrate that the algorithm is insensitive to the initial setting of e , and its value can be effectively set as

$$initial_e = \frac{r_{\max} + r_{\min}}{2r_{\min}}. \quad (2.19)$$

Since $(r_{\max} + r_{\min})/2$ is the mid point between $r_{\max}$ and $r_{\min}$, this formula gives a good initial value to the expanding factor by setting it to the middle of the video's scalable range $[r_{\min} \ r_{\max}]$.

Remark: We should note that, in addition to the periodic generation of utility functions, an on-demand asynchronous generation of instantaneous utility functions is also needed in some cases. For example, network algorithms may need to force a refresh of the current utility function, or the content server may re-initialize the operation of the utility predictor when there is substantial change in content (e.g., after detecting a scene change).

2.5. Evaluation of Utility Prediction Algorithm

2.5.1 Experiment Setup

Table 2.1: Video Traces Used in Experiment

Trace	Format	Encoding Rate	GOP Size	Length (minute)	Content Type	Scene Change
1	MPEG-1	constant	15	1	TV interview	slow
2	MPEG-1	constant	15	3.5	“True Lies”	fast
3	MPEG-1	constant	15	3.8	animation	very fast
4	MPEG-2	variable	12	9.3	“Forrest Gump”	medium

For experimentation we used three MPEG-1 video traces and one MPEG-2 video

trace representing a wide range of content types. The MPEG-1 traces (viz. Traces 1, 2 and 3) are encoded as constant rate (i.e., the peak rate of each GOP is fixed, but the actual bit rate still varies as frame size varies) and include various degrees of scene changes. The MPEG-2 trace (viz. Trace 4) represents a long video clip generated using the Columbia University MPEG-2 software encoder. The highly variable bit rate of Trace 4 is used to verify the capability of the utility prediction algorithm in selecting the appropriate range of scaling rates, (i.e., extracting the maximum and minimum scalable rates $r_{\max}$ and $r_{\min}$ either from frame header when the encoding rate is constant, or otherwise by measurement). It should be noted that even though we experiment with the MPEG-1/2 traces, the findings on utility prediction are general enough to be applied to other types of video streams because of the similarity in bit-stream syntax and compression techniques used by these coding techniques.

In comparison to continuous-rate scaling, discrete-rate scaling is relatively simple to model, (e.g., by measuring the rate of the dropped components and the drop in the utility value based on the scaling profile). In the following experiments, we are primarily concerned with how to predict utility functions over long utility generation interval, and without loss of generality, we focus on one continuous-rate scaling method as a means of implementing DCT coefficient dropping. We selected the dynamic rate shaping method [32] because DRS optimizes dropping to best approximate the ideal rate-distortion function.

In our experiments, we use 20 rate-scaling samples evenly spaced in the range $(r_{\min}, r_{\max})$ to formulate one normalized utility function. The resulting 19-segment piecewise linear function is further quantized into a 4-segment piecewise linear function, as discussed in Section 2.3.2.1. A per-GOP instantaneous utility function is formulated by taking the per-frame scaling rate samples and accumulating their

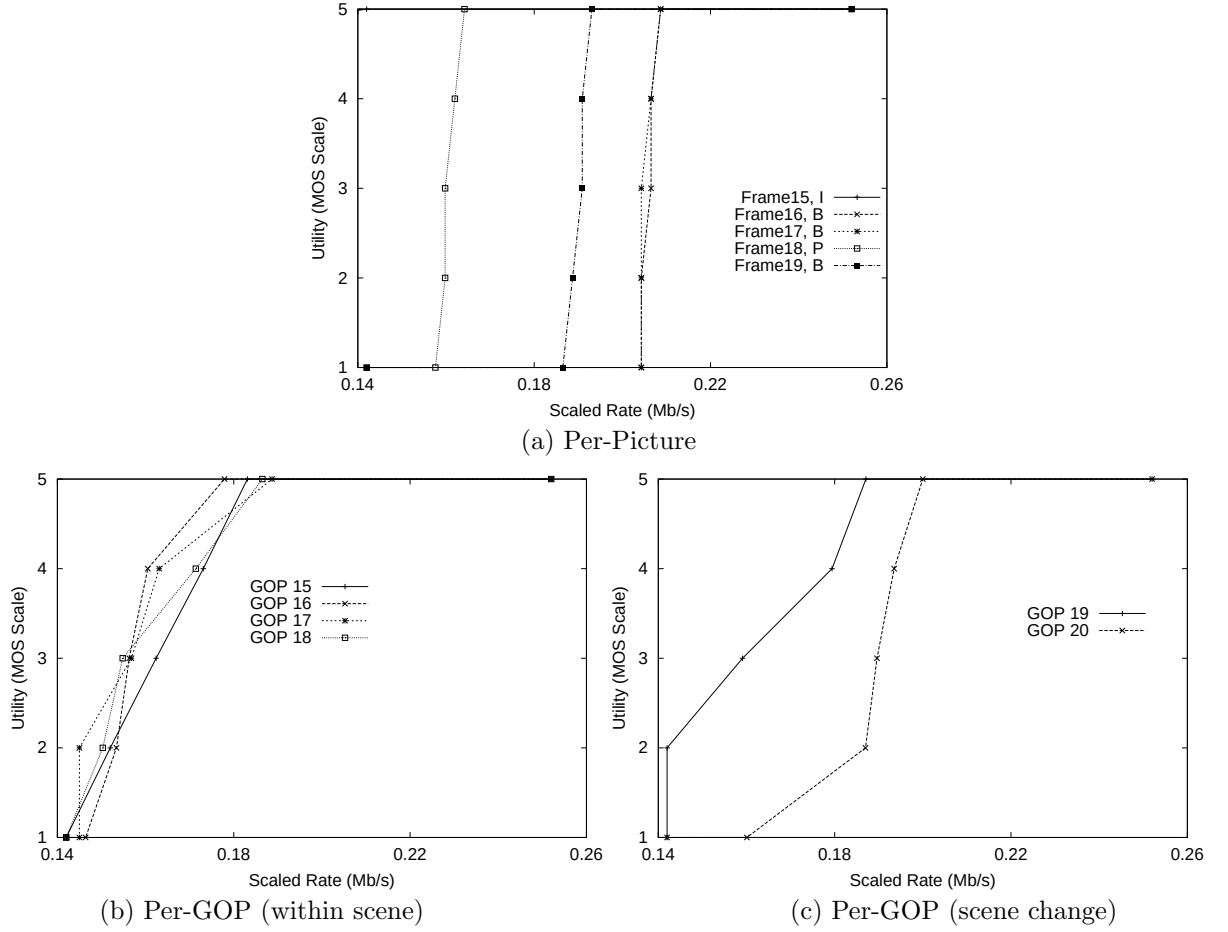


Figure 2-6: Example of Instantaneous Utility Functions

corresponding distortion measure over one entire GOP.

2.5.2 Algorithm Evaluation

The first experiment is designed to investigate the similarity of utility functions generated for every MPEG picture using Trace 1, which is constant-bit-rate encoded with $r_{\max} = 0.25\text{Mb/s}$ and $r_{\min} = 0.18\text{Mb/s}$. As illustrated in Figure 2-6(a), the resulting utility function varies significantly between pictures. Note that picture 15 and 18 are I and P pictures, respectively, and that pictures 16, 17 and 19 are B pictures.

We investigate per-GOP generated utility functions and observe that by averaging out fast inter-picture variation across a small number of pictures we can reduce the dependency over picture types. Smoothing over multiple pictures (e.g., a GOP) requires buffering utility functions for every picture. Because these functions are constructed from a number of scaling rate samples, in our case 20 samples, the required memory for such an operation is small.

Figures 2-6(b) and 2-6(c) illustrate a number of examples of the per-GOP (15 pictures) generated utility functions. To verify the effect of scene-based utility function generation, we further grouped the resulting curves by the similarity of their shapes. From a total of 1809 pictures from Trace 1, we generate 120 per-GOP instantaneous utility functions. These were then grouped into 22 different scene groups. The largest scene group contains 20 GOPs, while the smallest group contains only one GOP. In Figure 2-6(b), GOP 15 to 18 are associated with the same scene group with the corresponding utility functions being well matched. However, GOP 19 and 20, as illustrated in Figure 2-6(c), have quite different utility function shapes. Note that scene changes occur within these two GOPs.

This experiment demonstrates that the input signal to the utility prediction algorithm (i.e., the generated instantaneous utility functions) are very sensitive to picture types and scene changes. Even with scene detection and predication, the resulting utility functions may not be applicable over long time-scales. In fact, when considering GOPs 19 and 20, the generated utility function is only good for 0.5s (i.e., one GOP interval). In Trace 1, even the largest scene (with 20 GOPs) can only provide a valid and accurate utility function for a duration of approximately 10s. We observe that this may be a useful time-scale for the network adaptation time-scale to operate over. However, the prediction over even longer periods would be a more desirable goal.

Table 2.2: Sensitivity to Initial Value of e (T=30s)

initial_e	Number of Violations after the first 30s			
	Trace 1	Trace 2	Trace 3	Trace 4
1.1	0	1	0	3
1.2	0	1	0	3
1.3	0	0	0	3
1.4	0	0	0	3
1.5	0	0	0	3
auto e	0	0	0	3

The prediction algorithm is designed to operate over per-GOP generated instantaneous utility functions. We first demonstrate its performance sensitivity to *initial_e*, the initial value of the expanding factor e . The performance metrics used here is the number of prediction violations. In Table 2.2, we list the number of violations for all four traces under different *initial_e* after the first 30s. As *initial_e* changes from 1.1 to 1.5, the number of violations for all four traces do not change significantly. This behavior validates the effective operation of the adaptive mechanism to adjust e , which should not be sensitive to *initial_e* over the long run. Furthermore, we observe that the results are the same when compared to the last row of Table 2.2, which is generated under the “automatic setting” of *initial_e* using Equation (2.19). This result justifies the decision to internally set the value of *initial_e* making the prediction algorithm self-adaptive. All experiments use the self-adaptive prediction algorithm with no external control parameters.

A more surprising observation is that the algorithm is not very sensitive to the utility generation interval T . Intuitively, one would anticipate that the number of violations would possibly increase as T increases, (i.e. as the required utility generation interval becomes larger). Figure 2-7 shows the number of violations that occurred against T for the range 10-100s. As shown in the figure, the number of violations is almost constant as T varies between 30s and 100s. This result indicates that the algorithm will perform equally well for large and small values of T . The

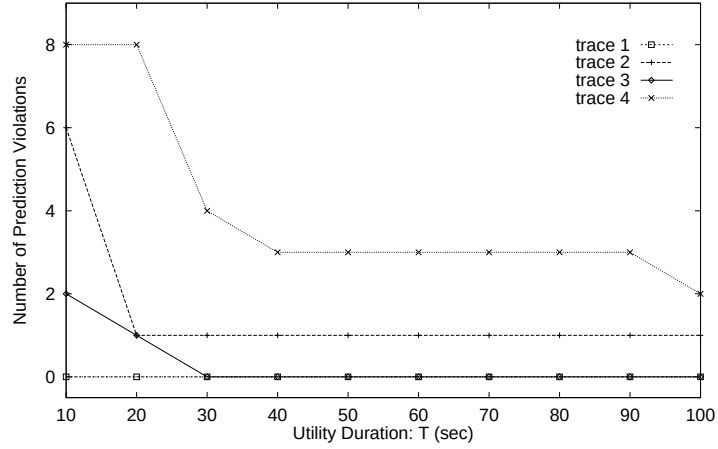


Figure 2-7: Algorithm Sensitivity to T

reason for this result is two-fold. First, the violations occur in bursts, caused by a sequence of fast scene changes in the video content. This phenomenon typically holds true in all four traces. Second, when the utility generation interval is large, the likelihood of reducing the expanding factor e becomes smaller because e can only be reduced at the end of the utility generation interval. However, it is not recommended to disable the reduction mechanism on e because the mechanism reduces the extent of over-estimation, as discussed in Section 2.5.3.

The shape of a predicted utility function is a good indicator of the effectiveness of the overall utility-based bandwidth renegotiation approach. Ideally the predicted utility function should span the whole range of scalable rates. A utility function with the shape of a step function is of little use for adaptation since it is equivalent to peak rate allocation.

Figures 2-8(a) and 2-8(b) illustrate predicted utility function for $T = 30s$. In the case of Trace 1, we can predict one utility function every 30s (i.e., over 60 GOPs) with no violations. The resulting utility functions are illustrated in Figure 2-8(a). Only two utility curves needed to be generated in this experiment. Each curve is

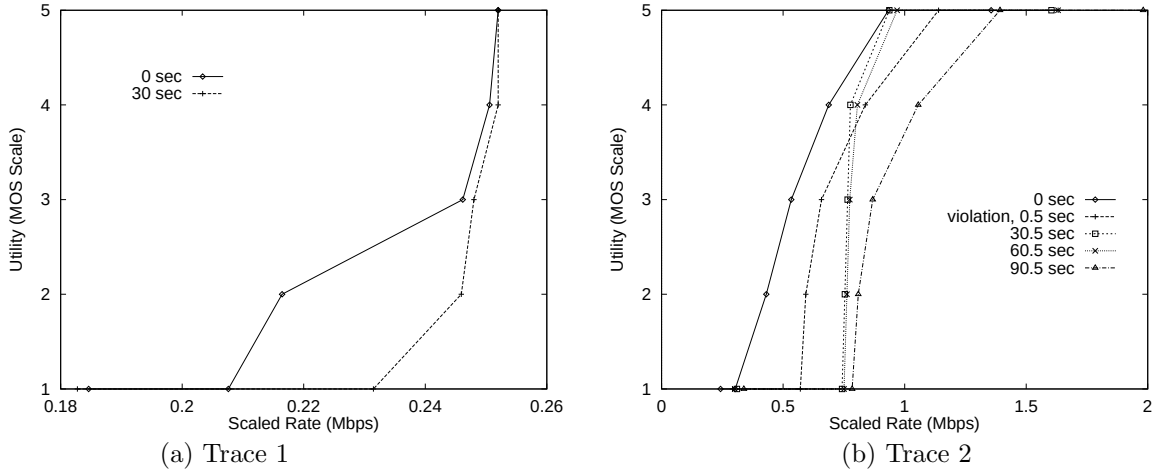


Figure 2-8: Examples of Predicted Utility Function Envelopes

derived from the measurement of 60 per-GOP instantaneous utility functions over the past 30s. Because no violations are observed for the complete video clip, each utility function can be used for network adaptation for the next 30s without any need to renegotiate a new utility function with network bandwidth allocation.

Figure 2-8(b) shows the predicted utility function for Trace 2, which contains more frequent scene changes than Trace 1. We observe that half of the utility functions do not have step-shapes at the peak rate. The two step-shape utility functions ($t = 30.5s$ and $60.5s$, respectively) in 2-8(b) are the result of a single violation (at time equals $0.5s$) due to bandwidth under-estimation by the utility prediction process at time $0s$. This is caused by the fact that there is no historical data available for the very first utility prediction. When increments are added to e , the subsequent utility functions are generated with a near step-shape, due to a large value of e . The prediction algorithm corrects this operation by reducing e . Observe that at $t = 90.5s$, the predicted function becomes a non-step shape once again.

2.5.3 Error Analysis

To quantitatively analyze the measurement error introduced by the prediction algorithm, we define two error metrics that measure the maximum distance between a predicted bandwidth vector $\mathbf{r}^{\text{pred}}(t_k)$ and the bandwidth vector of an instantaneous utility function $\mathbf{r}^{\text{inst}}(t)$, where $t_k \leq t < t_k + T$:

- the over-estimation error err^+ , which tracks the maximum amount of bandwidth over-estimation between a predicted bandwidth vector and the bandwidth vector of an instantaneous function. The error of $\mathbf{r}^{\text{pred}}(t_k)$ over-estimating $\mathbf{r}^{\text{inst}}(t)$ is given by

$$err^+ = \frac{\max_i r_i^{\text{pred}} - r_i^{\text{inst}}, 0}{r_{\max}}; \quad (2.20)$$

- the under-estimation error err^- , which captures the maximum amount of bandwidth under-estimation. The error of $\mathbf{r}^{\text{pred}}(t_k)$ under-estimating $\mathbf{r}^{\text{inst}}(t)$ is given by

$$err^- = \frac{\min_i \{r_i^{\text{pred}} - r_i^{\text{inst}}, 0\}}{r_{\max}}. \quad (2.21)$$

Both metrics are defined as the percentage of over or under allocation relative to the maximum scalable rate $r_{\max}$. Figure 2-9(a) shows the measured errors after applying the prediction algorithm to Trace 2. The top curve in the figure represents the over-estimation error, which frequently oscillates within the 10% to 50% range. The under-estimation error mostly remains zero, and generates a negative spike of less than -3% when violations are observed.

To determine the optimal performance, we ran the prediction algorithm in an off-line mode. The off-line predictor stores all the instantaneous utility functions generated during a target utility generation interval to accurately compute the bandwidth vector envelope for a particular utility generation interval. The resulting system becomes non-causal because it uses *a posteriori* data to predict bandwidth

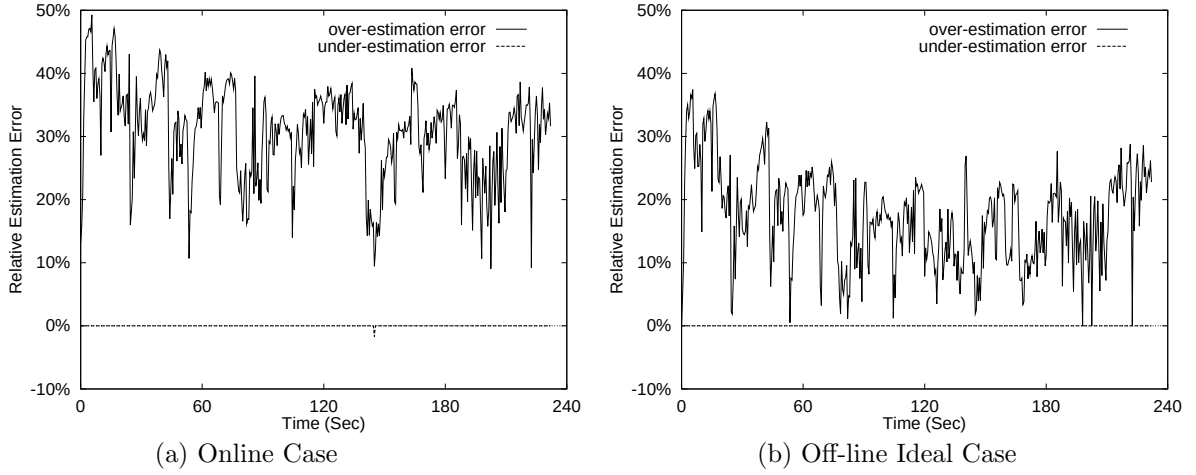


Figure 2-9: Utility Prediction Error (Trace 3, $T=20$)

vectors. Figure 2-9(b) shows the smallest estimation error that can be achieved by the off-line algorithm. The under-estimation error always remains zero. The over-estimation error has a similar shape to that shown in Figure 2-9(b) with the exception that it is shifted to a range between 0% and 35% (i.e., the over-estimation error is smaller by an amount of 15%). In this case, the over-estimation error is mainly a result of scene changes, and not caused by the utility generation procedure.

The extra degree of over-estimation, (e.g., 15% additional over-estimation in Figure 2-9(a)) is necessary for a causal system and represents a trade-off between the tightness of utility prediction and the number of violations observed in a utility generation interval.

In Figure 2-10, the time-averaged err^+ is shown with respect to T for all four traces. The extent of over-estimation is positively related to the rate dynamics of video content, which is represented by the err^+ in the off-line ideal estimation. From Figure 2-10(a), (b) and (c), we can observe that the degree of over-estimation remains flat as T increases. This is a result of the prediction algorithm operating over video with a constant maximum encoding rate, as is the case with traces 1, 2, and 3.

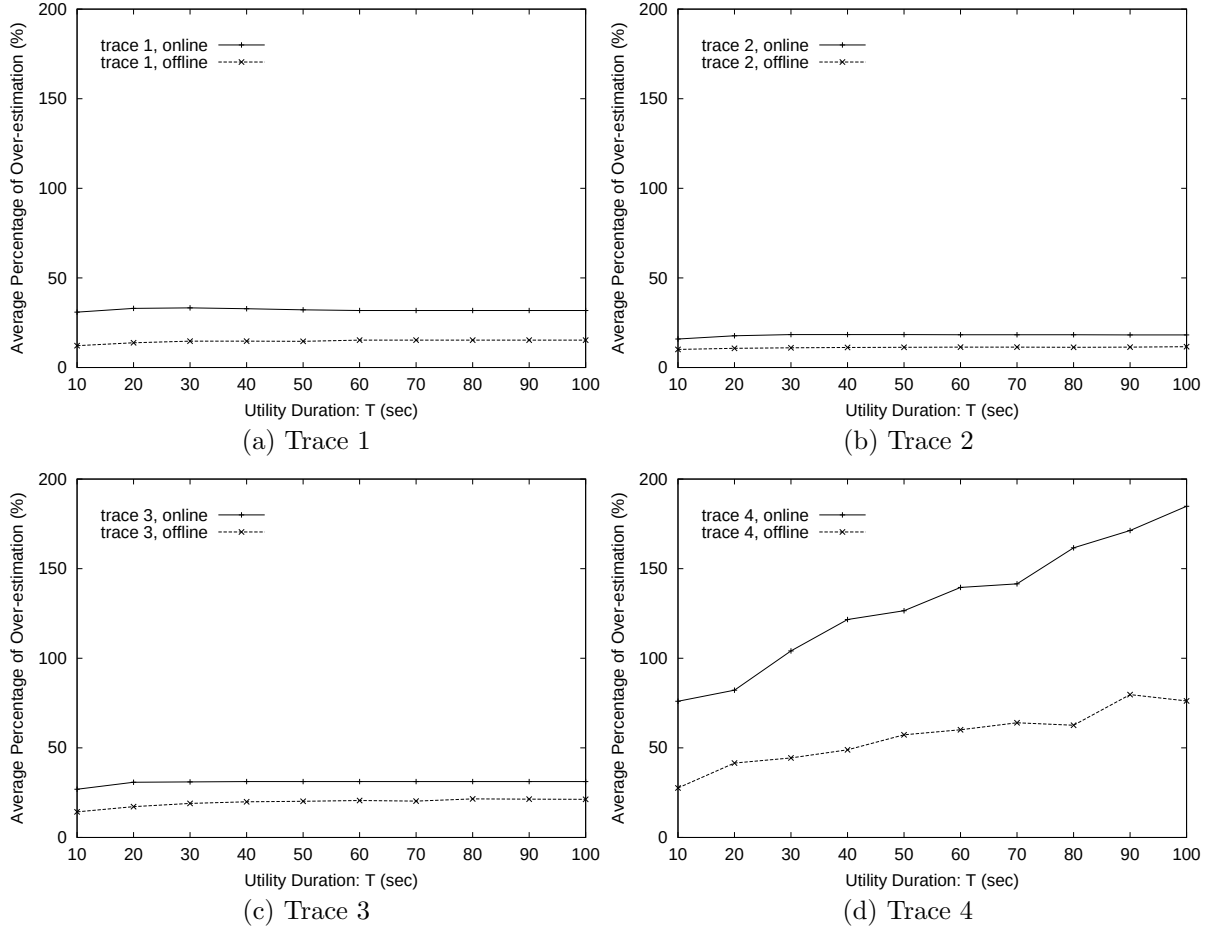


Figure 2-10: Time-averaged Over-estimation Error

Here, the constant maximum encoding rate is used by the prediction algorithm as an upper bound for over-estimation. As the video encoding rate becomes variable, as is the case with Trace 4, the over-estimation error starts to increase following the increase in T for both off-line and online prediction cases, as shown in Figure 2-10(d). Results from this experiment look very promising because utility functions for constant rate encoded video is not sensitive to scene changes, and hence, can be applied for longer utility generation intervals. The analysis on the optimal value of T is presented in Appendix A.

In all cases, the extent of over-estimation by the online prediction algorithm is about twice that of the off-line algorithm. This indicates that the prediction algorithm tracks the intrinsic dynamics of the content of the four videos stream under study while maintaining a small number of violations as T increases. These results demonstrate that the proposed prediction algorithm is applicable to a wide range of utility generation interval and to different types of video content.

2.6. Summary

The work presented in this chapter takes a cross-disciplinary approach to delivering multimedia services in networks by attempting to bridge the gap between image processing and network adaptation. The contribution of this chapter is as follows. We started by formalizing bandwidth utility metrics and formulating utility functions based on application and service types. After deriving utility functions for aggregates of TCP flows and small multimedia flows, we focused on the generation of video utility functions, which is challenging because of video content-dependent scalability. We analyzed the trade-off between utility generation and network adaptation time-scales. In order to extend the utility generation interval to match the network renegotiation time-scale, we proposed a self-adaptive algorithm capable of dynamically adjusting any bandwidth over-estimation in utility measurements over prolonged utility generation intervals. Our results verify the effectiveness of the proposed utility prediction algorithm, which operates in a self-adaptive manner without any external control parameters. The results look particularly promising for constant rate encoded video because we found that the predicted bandwidth vector envelopes are insensitive to scene changes.

In the next chapter, we will study utility-based link sharing algorithms and introduce a set of foundation algorithms that address fairness, service-differentiation,

and utility-maximization policies.

A. Optimum Utility Prediction Time-Scale

The prediction algorithm comprises two conflicting operations (i.e., incrementing and decrementing the expanding factor). These two mechanisms represent the trade-off between a large utility generation interval vs. an accurate measure on video scalability. One may view the variation in video content scalability composed of a spectrum of dynamics. The prediction algorithm filters out the intrinsic slow time-varying dynamics that can be utilized efficiently by the network over time-scales suited to its dynamics (e.g., signaling). Even though a non-causal system may operate off-line and calculate the *exact* bandwidth vector envelope for a video on demand system, this operation can not be applied to network adaptation because the desired utility generation interval may not be known in advance of the off-line operation. This is because the network adaptation time-scale is unknown before transmission and may vary during the lifetime of a video transport session due to variation in round trip delay and capacity of the signaling/control system that performs bandwidth renegotiation.

In what follows, we use mean-value analysis on the optimal utility generation interval to better analyze the trade-off between the reduction on system cost of periodic utility updates and the benefit of bandwidth savings resulting from a more updated utility function.

Based on the results presented in Section 2.5.2, prediction errors occur in bursts, hence we denote τ to be the interval between two consecutive bursts of prediction violations, and $\tau \gg T_{GOP}$, where T_{GOP} is the interval of a GOP. We observe that the experimental results discussed in Section 2.5.2 are insensitive to the utility generation interval T . Therefore, we ignore the effect due to the prediction algorithm,

and assume that τ is not a function of T but only related to the *major* scene changes in a video stream.

Since T_{GOP} represents the minimum utility generation interval, we have $T \geq T_{GOP}$. In addition, we may assume $T \leq \tau$ because the additional utility functions transmitted due to prediction violations will always reduce the utility generation interval T to a value less than τ .

We denote the signaling system capacity as C_{sig} (message/s). Because the rate of generating the periodic utility update messages is $1/T$, the total signaling workload W_{sig} introduced by one flow over τ interval is

$$W_{sig}(T) = \tau / (T \cdot C_{sig}). \quad (2.22)$$

For the data transport system, we assume its capacity is C_{data} (bit/s). Next, we calculate W_{data} , the amount of excessive allocation in the τ interval. The total amount of bits allocated during the τ interval is $\int_{t_0}^{t_0+\tau} e(t) \cdot \gamma(t) dt$, where $e(t)$ and $\gamma(t)$ denote the time varying expanding factor and resulting bandwidth allocation, respectively.

Assume that during the τ interval the allocated bandwidth remains constant and the utility value corresponding to the allocated bandwidth is ν . When the utility generation interval $T = \tau$, we have

$$\int_{t_0}^{t_0+\tau} e(t) \cdot \gamma(t) dt \cong \tau \cdot \gamma^{\max}(\nu), \quad (2.23)$$

$$\text{where } \gamma^{\max}(\nu) = \max_{1 \leq k \leq n} \{\gamma_k(\nu)\}, \quad n = \left\lfloor \frac{\tau}{T_{GOP}} \right\rfloor \quad \text{and} \quad \gamma_k(\nu) = v_k^{-1}(\nu).$$

Here $v_k^{-1}(\cdot)$ denotes the inverse function of each instantaneous utility function generated during the τ interval. We take $\gamma^{\max}(\nu) = \max_{1 \leq k \leq n} \{\gamma_k(\nu)\}$, the bandwidth

envelope corresponding to the utility value ν because only one predicted bandwidth vector is delivered (at time t_0) in this interval.

When $T = T_{GOP}$, namely every per-GOP instantaneous utility function is used as a predicted utility function, we have

$$\int_{t_0}^{t_0+\tau} e(t) \cdot \gamma(t) dt = \sum_{k=1}^n \gamma_k(\nu) \cdot T_{GOP} + \gamma_{n+1}(\nu) \cdot \frac{T_{GOP}}{2} \cong \tau \cdot \gamma^{mean}(\nu), \quad (2.24)$$

$$\text{where } \gamma^{mean}(\nu) = \frac{\sum_{k=1}^n \gamma_k(\nu)}{n} \text{ and } \gamma_k(\nu) = v_k^{-1}(\nu).$$

The term $T_{GOP}/2$ is the average length of the residual interval $(\tau - n \cdot T_{GOP})$. Here we assume that e is close to 1 because of the frequent (every T_{GOP}) utility prediction without any violations within an interval τ .

Now with the two known points of $W_{data}(T)$ (i.e., $W_{data}(T_{GOP}) = 0$ and $W_{data}(\tau)$ is equal to the difference of (2.23) and (2.24) normalized by C_{data}), we can derive a linear approximation of $W_{data}(T)$ as:

$$W_{data}(T) = \tau \cdot \frac{(\gamma^{\max}(\nu) - \gamma^{mean}(\nu))}{C_{data}} \frac{(T - T_{GOP})}{(\tau - T_{GOP})}. \quad (2.25)$$

The optimal utility generation interval T is obtained by solving the optimization problem: $\arg \min W_{sig}(T) + W_{data}(T)$ with constraint $T_{GOP} \leq T \leq \tau$. The solution without constraint is

$$T^* = \sqrt{\frac{C_{data}}{C_{sig}} \frac{(\tau - T_{GOP})}{(\gamma^{\max}(\nu) - \gamma^{mean}(\nu))}}. \quad (2.26)$$

When $T^* \in [T_{GOP}, \tau]$, we have the optimal solution $T_{opt} = T^*$. When $T^* < T_{GOP}$, the objective function $W_{sig}(T) + W_{data}(T)$ is strictly increasing, hence $T_{opt} = T_{GOP}$. Similarly, when $T^* > \tau$, the objective function is strictly decreasing and $T_{opt} = \tau$.

Therefore, the optimal solution is:

$$T_{opt} = \max \{ T_{GOP} , \min \{ \tau , T^* \} \} . \quad (2.27)$$

We observe that $1/T^*$, the ideal frequency of utility generation is proportional to the square root of three values: C_{sig}/C_{data} , $(\gamma^{\max}(\nu) - \gamma^{\text{mean}}(\nu))$, and roughly $1/\tau$ (when $\tau \gg T_{GOP}$). Here C_{sig}/C_{data} represents the signaling system capacity normalized by the transport system capacity. $(\gamma^{\max}(\nu) - \gamma^{\text{mean}}(\nu))$ measures the rate variation between scene changes, while $1/\tau$ models the frequency of major scene changes. In other words, the ideal frequency of utility generation is proportionally increasing to the square-roots of the normalized signaling system capacity, rate variation within a scene, and frequency of scene changes, respectively. An important observation is that when $(\gamma^{\max}(\nu) - \gamma^{\text{mean}}(\nu))$ is small, (i.e., the rate variation is constrained by encoding techniques such as a constant-bit-rate encoder), T^* can be quite large, then the utility prediction algorithm becomes most effective, as we have seen from experimental results in Section 2.5.3.

Chapter 3

Utility-Based Link Allocation Algorithms

3.1. Introduction

Based on the utility-based network control framework (presented in Figure 2-2 of Section 2.1.1) we now study the design of efficient link sharing algorithms based on this formulation. An important advantage of utility-based bandwidth allocation algorithms is that their response to network congestion is faster than conventional resource renegotiation protocols. Conventional reservation messages only carry a single bandwidth value [31, 7] and therefore, extensive renegotiation is usually required prior to any bandwidth allocation adjustments. In contrast, bandwidth utility functions capture the entire bandwidth range within which applications can successfully operate. Therefore, utility-based allocation algorithms have sufficient information to react to network congestion without waiting for user interaction or application approval. This advantage becomes even more apparent and beneficial in edge-based wireless networks where network conditions can vary frequently, hence the network cannot afford to wait, nor are mobile devices capable of renegotiating over such potentially fast time-scales, nor capable of dealing with the complexity of such a renegotiation process.

Utility-based bandwidth allocation can leverage the potential synergy of using

one set of algorithms and control parameters while collectively gaining the benefit of both application-aware and service-differentiated allocation. However, this synergy can only be built on the foundation of *compatible and flexible* utility-based algorithms. Currently there is a lack of utility-based foundation algorithms that are efficient and at the same time flexible enough to realize various bandwidth allocation policies. These policies may include equalizing the utility of all applications, or differentiating utility by service class, or maximizing the total utility. A hierarchical integration of these policies is often required to reflect the structure of service plans or organization hierarchy [40] (e.g., a service subscriber could be an organization that has a company-level allocation policy with different policies at different department levels). Utility-based algorithms need to be capable of supporting such service plans and organization hierarchies found in practical network settings.

The contribution of this chapter is as follows. We address the challenge discussed above by proposing two classes of utility-based allocation algorithms that include utility-based differentiated allocation (including its special case of utility-fair allocation) and utility-maximizing allocation. These algorithms form the foundation of our utility-based bandwidth management framework. These algorithms are implemented over a link sharing scheme that uses aggregated bandwidth utility functions to adjust the link-sharing ratio of each flow aggregate for a class based queueing (CBQ [40]) scheduler. We extend the CBQ scheduler to support bandwidth allocation policies in a hierarchical form based on customer service types and traffic classes.

The structure of the chapter is as follows. In Section 3.2. we discuss the related work. This is followed in Section 3.3. and Section 3.4. by detailed descriptions of a proportional utility-differentiated allocation algorithm and a utility-maximizing allocation algorithm, respectively. We evaluate these link sharing algorithms and

augment a CBQ scheduler to support utility-based hierarchical link sharing in Section 3.5. Finally, we present some concluding remarks in Section 3.6.

3.2. Related Work

There is a lack of research associated with the implementation of utility-based allocation. Several isolated algorithms such as utility maximization (i.e., the Q-RAM [85] framework of maximizing total utility) and our utility-based fair allocation [12] have been proposed. However, these algorithms are not compatible with each other (e.g., their formulation has to be modified to support hybrid or hierarchical policies). In addition, the utility maximization algorithm is also known to be computationally intensive (i.e., NP-hard). In this chapter, we present two groups of utility-based foundation algorithms that can efficiently realize utility-maximizing and utility-based differentiated allocation. Our utility maximization algorithm leverages the piecewise linear quantization of utility functions and uses the Kuhn-Tucker [62] condition to significantly reduce algorithm execution time. Our utility-based differentiation algorithm supports utility-based fair allocation and allows individual utility function to have different maximum utility values.

There has been no investigation on how to integrate different allocation policies (e.g., fairness vs. utility-maximizing) together in the same link-sharing scheduler. In this chapter, we present a first attempt to do this and augmenting the CBQ [40] link sharing algorithm to support hierarchical utility-based allocation algorithms, which can be configured to support utility-fair, utility-differentiated or utility-maximizing allocation policies across customer types and service classes.

3.3. Proportional Utility-Differentiated Allocation

We first present the simpler version of proportional utility-fair allocation, and then extend the algorithm to encompass the more general utility-differentiated allocation.

3.3.1 Proportional Utility-Fair

Proportional utility-fair gives the same proportion of the maximum utility value to each flow. When all flows have the same maximum utility value $u_{\max,}$, the algorithm becomes the utility fair allocation described in [12, 20]. In addition, we generalize the formulation by removing the constraint given in [12] that all individual utility functions need to have the same discrete utility levels, (i.e., $u_{i,k} = u_{j,k}$).

For piecewise linear function $u_i(x)$, we denote the set of normalized discrete utility levels as $\mathcal{A}_i = \left\{ \frac{u_{i,k}}{u_i^{\max}} \mid k \in \{1, \dots, K_i\} \right\}$. For the aggregated utility function $u_{agg}(x)$, the set of normalized discrete utility levels is the union of each individual set $\mathcal{A} = \bigcup_i \mathcal{A}_i$. We rename the members of $\mathcal{A}$ sorted in ascending order as $v_{agg,k}$.

Proposition 2 *Under the proportional utility-fair policy, the normalized aggregated utility function becomes:*

$$v_{agg}(x) = \frac{v_{agg,k+1} - v_{agg,k}}{b_{agg,k+1} - b_{agg,k}}(x - b_{agg,k}) + v_{agg,k}, \quad \forall x \in [b_{agg,k}, b_{agg,k+1}), \quad (3.1)$$

where $b_{agg,k} = \sum_i u_i^{-1}(v_{agg,k} u_i^{\max})$, and the aggregated utility function is:

$$u_{agg}(x) = v_{agg}(x) \sum_i u_i^{\max}. \quad (3.2)$$

Given a link capacity C , the resulting allocation x_i and utility value u_i to each flow is:

$$u_i = v_{agg}(C) u_i^{\max}, \quad \text{and} \quad x_i = u_i^{-1}(u_i). \quad (3.3)$$

Proof: It is straightforward that the allocation given by Equation (3.3) satisfies proportional utility fairness definition because $\forall i, j, u_i/u_i^{\max} = u_j/u_j^{\max} = v_{agg}(C)$. What needs to be verified is that the allocation $\{x_i\}$ is Pareto efficient, (i.e., no allocation can increase without reducing another one), that is $\sum_i x_i = C$ if the total bandwidth demand $\sum_i b_{i,K_i} > C$. Since $v_{agg}(C)$ is also the normalized utility value of u_i , and $v_{agg}(C) \in [v_{agg,k}, v_{agg,k+1})$, in this range the inverse function $u_i^{-1}(u_i)$ is linear with the form:

$$\begin{aligned} x_i = u_i^{-1}(u_i) &= \frac{b_{i,k+1} - b_{i,k}}{u_{i,k+1} - u_{i,k}}(u_i - u_{i,k}) + b_{i,k} \\ &= \frac{b_{i,k+1} - b_{i,k}}{v_{agg,k+1} - v_{agg,k}}(v_{agg}(C) - v_{agg,k}) + b_{i,k}. \end{aligned}$$

where $u_{i,k} \triangleq v_{agg,k} u_i^{\max}$, and $b_{i,k+1} \triangleq u_i^{-1}(v_{agg,k} u_i^{\max})$. Taking the sum, we have

$$\begin{aligned} \sum_i x_i &= \frac{v_{agg}(C) - v_{agg,k}}{v_{agg,k+1} - v_{agg,k}} \left(\sum_i b_{i,k+1} - \sum_i b_{i,k} \right) + \sum_i b_{i,k} \\ &= \frac{v_{agg}(C) - v_k}{v_{agg,k+1} - v_{agg,k}} (b_{agg,k+1} - b_{agg,k}) + b_{agg,k} \\ &= C. \end{aligned}$$

The last step is derived by replacing $u_{agg}(C)$ with (3.1). □

Remark: When all the $u_i(x)$ are the same, the outcome of the algorithm is equal allocation, which is the same result as the max-min allocation for a single link.

3.3.2 Proportional Utility-Differentiation

The proportional utility-fair allocation algorithm is limited to provide only fair allocation. We extend the algorithm to support differentiated allocation, (i.e., a controlled “unfair” allocation in terms of utility values for multiple service classes). Let $\beta_i > 0$ denote the utility differentiation parameter for flow i . The differentiated

utility fair allocation ensures that for i, j whose utility values are less than their maximum, $\frac{u_i/u_i^{\max}}{u_j/u_j^{\max}} = \frac{\beta_i}{\beta_j}$.

We extend the definition of $\mathcal{A}_i$, the set of normalized discrete utility levels defined in Section 3.3.1, by incorporating β_i , that is, $\mathcal{A}_i = \left\{ \frac{u_{i,k}}{\beta_i u_i^{\max}} \mid k \in \{1, \dots, K_i\} \right\}$. The normalized utility value $v_{agg,k}$ represents the ascendingly sorted member of $\mathcal{A} = \bigcup_i \mathcal{A}_i$. For the set of n flows, we denote $\mathcal{N}_k = \{i \mid v_{agg,k} < 1/\beta_i, i = 1, \dots, n\}$, and $\bar{\mathcal{N}}_k = \{i \mid v_{agg,k} \geq 1/\beta_i, i = 1, \dots, n\}$.

Proposition 3 *Under the proportional utility-differentiated policy, the normalized aggregated utility function becomes:*

$$v_{agg}(x) = \frac{v_{agg,k+1} - v_{agg,k}}{b_{agg,k+1} - b_{agg,k}}(x - b_{agg,k}) + v_{agg,k}, \quad \forall x \in [b_{agg,k} \ b_{agg,k+1}), \quad (3.4)$$

where $b_{agg,k} = \sum_{\forall i \in \mathcal{N}_k} u_i^{-1}(v_{agg,k} \beta_i u_i^{\max}) + \sum_{\forall i \in \bar{\mathcal{N}}_k} b_{i,K_i}$, and the aggregated utility function is:

$$u_{agg}(x) = v_{agg}(x) \sum_{\forall i \in \mathcal{N}_k} \beta_i u_i^{\max} + \sum_{\forall i \in \bar{\mathcal{N}}_k} u_i^{\max}. \quad (3.5)$$

Given a link capacity C , the resulting allocation x_i and utility value u_i to each flow is:

$$u_i = u_i^{\max} \min\{v_{agg}(C) \beta_i, 1\}, \text{ and } x_i = u_i^{-1}(u_i). \quad (3.6)$$

Proof: The proof is the same as for Proposition 2, by replacing $u_{i,k} = v_{agg,k} u_i^{\max}$ with $u_{i,k} = \beta_i v_{agg,k} u_i^{\max}$. $\square$

Remark: It is clear from (3.6) that the proportional utility-differentiated allocation gives flows in $\mathcal{N}_k$ utility-differentiated allocation. For all flows in $\bar{\mathcal{N}}_k$, the utility-differentiated allocation will exceed each flow's maximum bandwidth requirement, therefore the allocation will remain at the maximum.

Remark: The proportional utility-fair allocation is a special case of the proportional utility-differentiated allocation where all the β_i are the same. In addition, the additional complexity of the utility-differentiated allocation lies in maintaining the sorted set $\mathcal{A}$ and $\overline{\mathcal{N}}_k$.

3.4. Utility-Maximizing Allocation

3.4.1 Algorithm Formulation

A utility-maximizing (also known as welfare-maximizing) allocation distributes the link capacity C into per-flow allocations $\mathbf{x} = (x_1, \dots, x_n)$ to maximize $\sum_{i=1}^n u_i(x_i)$ under constraint $\sum_{i=1}^n x_i \leq C$ and $0 \leq x_i \leq b_i^{\max}$. Here we only consider the scaled piecewise linear utility functions after utility scaling (see Section 2.3.1). We focus on the scenario where $\sum_{i=1}^n b_i^{\max} > C$ and rewrite the constraint as $\sum_{i=1}^n x_i = C$ because $u_i(x_i)$ is monotonically increasing in x_i . The case where $\sum_{i=1}^n b_i^{\max} = C$ is trivially solved by giving $u_i(x_i) = b_i^{\max}$.

This maximization problem with target functions that are not always concave is an NP-hard [66] problem. In the case of convex utility functions, the optimal solution lies at the extreme points of the convex hull with the optimal solution only being found by enumerating through all the possible extreme points. In the Q-RAM project [66], various approximation algorithms have been investigated to reduce the complexity of aggregating utility functions. In what follows, we present an enhancement to brute-force utility-maximizing algorithms by exploiting the structure of piecewise linear utility functions and reducing the algorithm's searching space.

One direct result from the Kuhn-Tucker [62] necessary condition for maximization is that:

Proposition 4 *At the maximum-utility allocation $(x_1^*, \dots, x_n^*)$, the allocation to i*

belongs to one of the two sets: either $i \in \mathcal{D} \triangleq \{j \mid u'_j(x_j^*-) \neq u'_j(x_j^*+)\}$, namely x_i^* is at a first-order discontinuity point of $u_i(x)$; or otherwise, $\forall i, j \in \overline{\mathcal{D}}$, $u_i(x_i^*)$ and $u_j(x_j^*)$ have the same slope: $u'_i(x_i^*) = u'_j(x_j^*)$. The slope has to meet the condition that

$$u'_j(x_j^*-) \geq u'_i(x_i^*) \geq u'_j(x_j^*+), \forall i \in \overline{\mathcal{D}} \text{ and } j \in \mathcal{D} \quad (3.7)$$

The intuition behind the formulation is that for $i, j \in \overline{\mathcal{D}}$, $u_i(x_i^*)$ and $u_j(x_j^*)$ must have the same slope. Otherwise, let's assume that $u'_i(x_i^*) < u'_j(x_j^*)$. Then, the total utility can be increased by moving bandwidth allocation from x_i^* to x_j^* . By the same argument, the slope of $u_i(x_i^*)$, $i \in \overline{\mathcal{D}}$ has to be no greater than the slope of $u_j(x_j^*-)$, but no smaller than that of $u_j(x_j^*+)$, for $j \in \mathcal{D}$.

Remark: Graphically, the utility-maximizing aggregated utility function represents the upper envelope of the “shifted” line segments of each individual piecewise linear utility function. Each of these line segments is shifted by every combination of first-order discontinuity break points in all other utility functions. Figure 3-1 shows the two steps of operations to aggregate two utility functions. The first step is to shift each of the two functions respectively by the break points of the other utility function. The second step is to plot the two set of shifted functions in one figure and then find the upper envelop of all the shifted function segments.

In general, it is computationally intensive to find the upper envelope over such a large set of shifted line segments. Figure 3-2 lists the pseudo code of an algorithm that calculates the aggregated utility function from two piecewise linear utility functions $u_i(x)$ and $u_j(x)$. The lines that start with “KT” are the additions due to Inequality (3.7), which is essential to improve the efficiency of the utility maximization algorithm.

Following the vector representation from (2.3) of piecewise linear utility functions, we denote the l th and m th segments of $u_i(x)$ and $u_j(x)$ as $u_{i,l}(x)$ and $u_{j,m}(x)$,

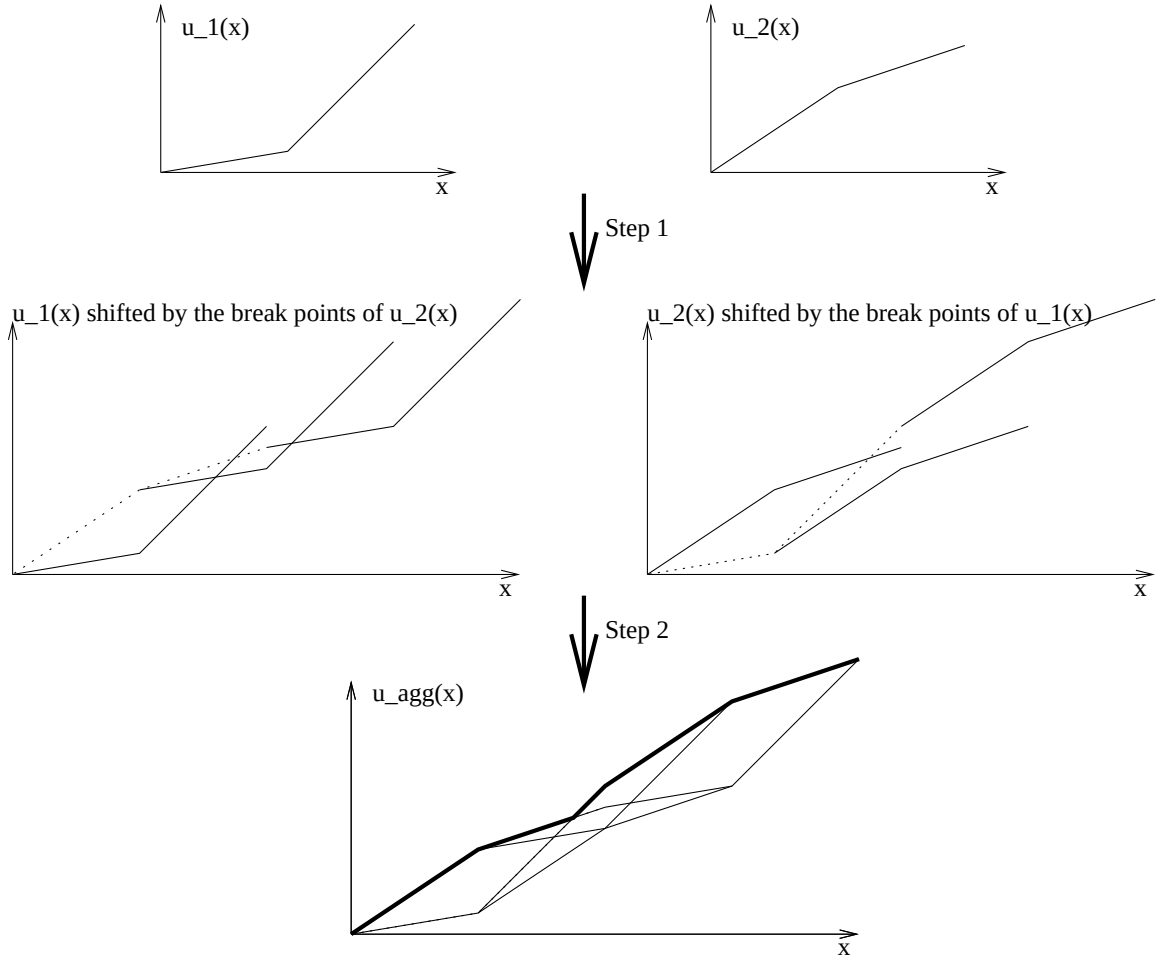


Figure 3-1: Example of Utility-maximizing Aggregation

respectively, where $l = 1, \dots, K_i$ and $m = 1, \dots, K_j$. The aggregated utility function is the upper envelope of the set of line segments of $u_i(x)$ and $u_j(x)$ that are shifted by each other's line segments. These shifted segments can be represented by $u_{i,l+k_j}(x) \triangleq u_{i,l}(x - b_{j,k_j}) + u_{j,k_j}$ and $u_{j,m+k_i}(x) \triangleq u_{j,m}(x - b_{i,k_i}) + u_{i,k_i}$ with $k_i = 1, \dots, K_i$ and $k_j = 1, \dots, K_j$, respectively.

From Inequality (3.7), we can remove at least one of $u_{i,l+m}(x)$ and $u_{j,m+l}(x)$ from the set because they cannot both satisfy the inequality. In addition, when $u_i(x)$ is strictly concave or convex, we have more reductions in line segments.

```

 $u_{i,l}(x)$ :      the  $l$ th linear segment of  $u_i(x)$ ;
 $u_{j,m}(x)$ :      the  $m$ th linear segment of  $u_j(x)$ ;
 $u_{i,l+k_j}(x)$ :   the  $l$ th linear segment of  $u_i(x)$  shifted by  $u_{j,k_j}(x)$ ;
 $u_{j,m+k_i}(x)$ :   the  $m$ th linear segment of  $u_j(x)$  shifted by  $u_{i,k_i}(x)$ ;

Utility_Maximizing_Aggregation( $u_i(x)$  ,  $u_j(x)$ )
{
    initialize set  $\mathcal{S} = \{u_i(x) , u_j(x)\}$ ;
    for all  $k_j \in \{1, \dots, K_j\}$ 
        for all  $l \in \{1, \dots, K_i\}$ 
KT      if  $(u'_{i,l}(x) > u'_{j,k_j}(x) \text{ OR } (\exists u_{j,k_j+1}(x) \text{ AND } u'_{i,l}(x) < u'_{j,k_j+1}(x)))$ 
        // Proposition 4
KT      continue; // skip line segment  $u_{i,l+k_j}(x)$ 
        add line segment  $u_{i,l+k_j}(x)$  to  $\mathcal{S}$ ;
    for all  $k_i \in \{1, \dots, K_i\}$ 
        for all  $m \in \{1, \dots, K_j\}$ 
KT      if  $(u'_{j,m}(x) > u'_{i,k_i}(x) \text{ OR } (\exists u_{i,k_i+1}(x) \text{ AND } u'_{j,m}(x) < u'_{i,k_i+1}(x)))$ 
        // Proposition 4
KT      continue; // skip line segment  $u_{j,m+k_i}(x)$ 
        add line segment  $u_{j,m+k_i}(x)$  to  $\mathcal{S}$ ;
    sort line segments in  $\mathcal{S}$  in ascending order of their
        starting point bandwidth value;
     $u_{agg}(x) = \text{upper\_envelope}(\mathcal{S})$ ; // find envelope of line segments in  $\mathcal{S}$ 
    return  $u_{agg}(x)$  ;
}

```

Figure 3-2: Pseudo-code of Utility-Maximization Algorithm

Proposition 5 *For concave piecewise linear utility functions, the utility-maximizing aggregated function comprises a set of every line segment from every individual utility function. These line segments are placed from left ($x = 0, u(x) = 0$) to right in descending order of their slopes, (i.e., the aggregated utility function is also concave).*

Proof: This proposition results directly from the fact that a concave piecewise linear utility function has its line segments in descending order of their slopes. With the constraint of Proposition 4, the property becomes the only viable outcome. $\square$

Proposition 6 *For two convex piecewise linear utility functions, the utility-maximizing*

aggregated function represents the upper envelope of two functions that are constructed by shifting one convex utility function to the end point of the other.

Proof: For two convex piecewise linear utility functions $u_i(x)$ and $u_j(x)$, all the shifted segments $u_{i,l+k_j}(x)$ and $u_{j,m+k_i}(x)$ will be removed except $k_j = 0, K_j$ or $k_i = 0, K_i$. These remaining segments correspond exactly to two shifted curves of $u_i(x)$ onto $u_j(x)$, and $u_j(x)$ onto $u_i(x)$. $\square$

Remark: Unlike the case in Proposition 5, the resulting aggregated function from Proposition 6 may not be purely convex. Therefore, Proposition 6 cannot be directly applied to aggregating more than two convex utility functions, but only to the convex portions of the functions. It is clear that the complexity of the utility maximization algorithm arises from handling the extreme points of convex utility functions. Next, we will use simulation to evaluate the performance gain by using Inequality (3.7) in the utility-maximization algorithm.

3.4.2 Algorithm Evaluation

We implement the exact utility maximization algorithm with and without applying the Kuhn-Tucker condition, (i.e., the pseudo-code shown in Figure 3-2 with and without the “KT” lines, respectively). We measure the CPU time and the number of intermediary line segments generated for the utility maximization algorithm to aggregate a number of utility functions. The performance metrics are the percentage of reduction in CPU time, and in number of intermediary line segments generated, which indicates the amount of saved operations.

In each simulation run, we use ten randomly generate piecewise linear utility functions, where the number of line segments is randomly chosen from the set $\{2, 3, 4, 5, 6\}$ with equal probability of 0.25. These utility functions are either purely

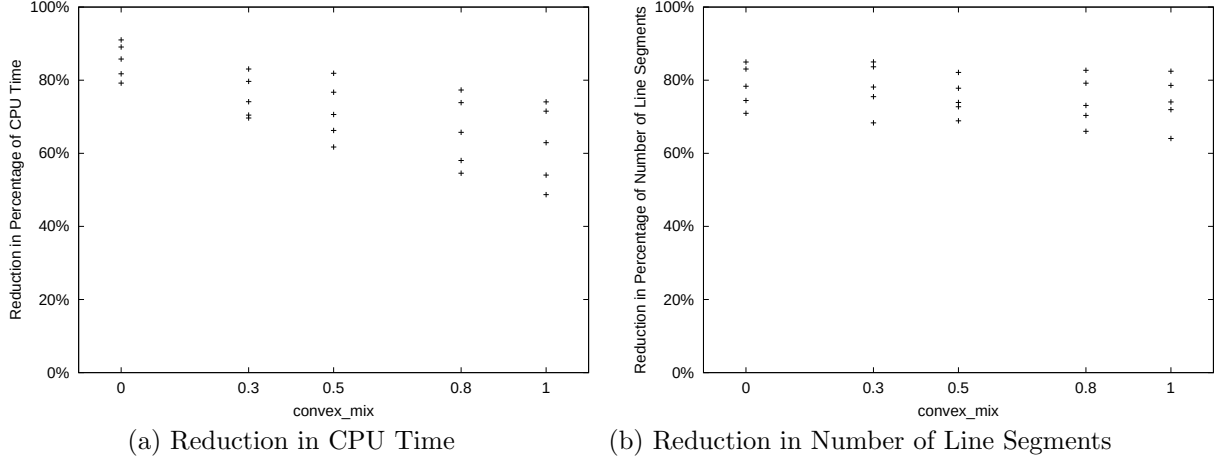


Figure 3-3: Performance of Utility-based Utility Maximization Algorithm

concave or convex¹. We use a simulation parameter *convex_mix* to control the percentage of these ten utility functions that are convex, that is, *convex_mix* = 0 represents the case when all of the ten utility functions are concave; and *convex_mix* = 0.8 represents the case with eight convex and two concave utility functions.

Figure 3-3 (a) and (b) show the performance gain in CPU time and number of line segments, respectively. Each sample point in the two plots represents one simulation run. We observe that the most gain is achieved when all of the individual utility functions are concave, and the least gain occurs when all of the individual utility functions are convex, as governed by Properties 5 and 6. In all the cases, the performance savings are significant. For example, with *convex_mix* = 1, all of the five runs have a gain of at least 48.69% in CPU time and 64.05% in the number of line segments.

¹It should be noted that since utility functions with mixed concavity will be formed in the intermediate aggregation stages when one concave function is aggregated with another convex function, our use of purely concave or convex functions does not limit the evaluation scope.

3.4.3 Aggregation State

The state management of aggregated utility function under utility maximization requires special consideration. An aggregated utility function under proportional utility-fair or utility-differentiated allocation contains the state for all individual utility functions. When a utility function is removed from the aggregated utility function, the reverse operation of Equations (3.2) and (3.5) does not involve other individual utility functions. However, this is not the case for the utility maximization algorithm. Figure 3-4 illustrates an example comprising a convex function $u_1(x)$ and a concave function $u_2(x)$. The aggregated function under utility maximization only contains information of the concave function $u_2(x)$. When $u_2(x)$ is removed from the aggregated utility function, there is insufficient information to reconstruct $u_1(x)$. In this sense the utility function state is not scalable under utility-maximizing allocation. Because of this and complexity concerns, utility-maximizing allocation should not be used for large numbers of flows with convex utility functions.

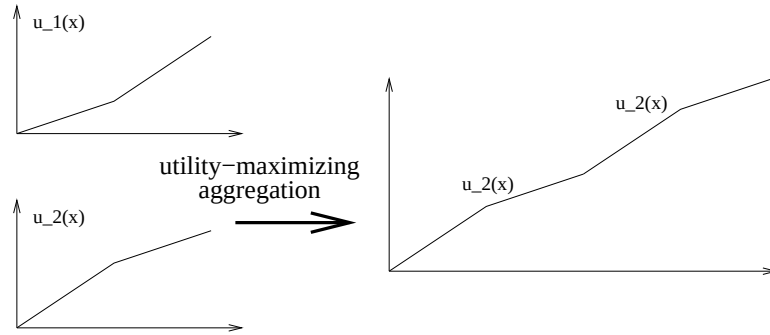


Figure 3-4: Examples of Utility Aggregation under Utility Maximization

3.4.4 Priority Allocation

A priority allocation policy allocates bandwidth to higher priority flows up to their maximum bandwidth requirements before allocating bandwidth to any flows with

lower priority. The aggregated utility function of priority allocation has a straightforward construction, as follows.

Proposition 7 *The aggregated utility function for the priority allocation policy is represented by the cascaded utility functions of each individual utility function sorted in descending order based on the flow priority. Here we assume that a higher priority has a higher priority value, otherwise the order is ascending.*

The utility-maximizing allocation can also realize the priority allocation policy by scaling $u_{i,\max}$ to $\alpha_i u_{i,\max}$. The reason for this is that utility scaling changes the slope of all the line segments of a piecewise linear utility function by the same factor α_i . Therefore, starting from low to high priority flows, one can choose α_i to increase $u_{i,\max}$ such that all the utility line slopes of a higher priority flow are higher than all the utility line slopes of all lower priority flows.

In the next two sections, we will present the implementation and evaluation of the proposed utility prediction and utility-based allocation algorithms, respectively.

3.5. Implementation and Evaluation of Utility-Based Allocation

In the previous sections 3.3. and 3.4., we introduced a number of foundation utility-based allocation algorithms (viz. proportional utility-differentiation and the special case of proportional utility-fairness, utility-maximization, and priority allocation) that represent another important part of our framework, serving as a set of building blocks for implementing utility-based link sharing schedulers. However, a number of technical barriers exist to the deployment of utility-based allocation in existing networks.

First, utility-based allocation algorithms need to be sufficiently flexible enough to accommodate complex combinations of allocation policies. This is driven by the needs of network service providers to enrich service offerings leading to the demand for more complex service structures comprising multiple service classes with different allocation policies. In addition, deployment within an enterprise network also calls for the implementation of allocation policies that reflect the administrative structures of an organization. To address these needs, utility-based allocation algorithms need to be capable of being deployed over hierarchical link-sharing schedulers. In what follows, we discuss the implementation and evaluation of a utility-based hierarchical and hybrid link-sharing scheduler that can flexibly implement a wide variety of policies using the proposed utility-based allocation algorithms.

3.5.1 Utility-Based Hierarchical and Hybrid Link Sharing

A number of link sharing algorithms have been proposed in the literature, among them, the class based queueing (CBQ [40]) algorithm represents a simple and widely cited approach. A CBQ server comprises a link sharing regulator and a general scheduler. The link sharing regulator determines whether to regulate a CBQ class based on the CBQ formal sharing rule, which guarantees that a CBQ class receives its share of bandwidth based on the “link sharing weight”. Any unused bandwidth is distributed among unregulated CBQ classes based on a general scheduler. In CBQ, regulated packets are shaped (i.e., buffered) and unregulated packets are served by a general scheduler. The capability of decoupling the link sharing regulator from the general scheduler is an important feature of CBQ because it allows bandwidth allocation without the concern of priority classes. For example, one does not need to allocate additional bandwidth to a high priority CBQ class (e.g., real-time video) as long as the general scheduler supports priority scheduling.

Since all of current link sharing algorithms are designed with the assumption that the link sharing weights are dynamically configured by external methods, any utility-based allocation algorithms can easily interact with CBQ and other link sharing schedulers by controlling the link sharing weights. Figure 3-5 illustrates the structure of our utility augmented CBQ (denoted as U(x)-CBQ) link sharing server. The tree topology reflects the link sharing hierarchy, (e.g., the bandwidth allocated to one class is the sum of bandwidth allocated to all of its child classes in the hierarchy). A utility-based allocation module is associated with every CBQ class at different hierarchy levels that are higher than the leaf classes.

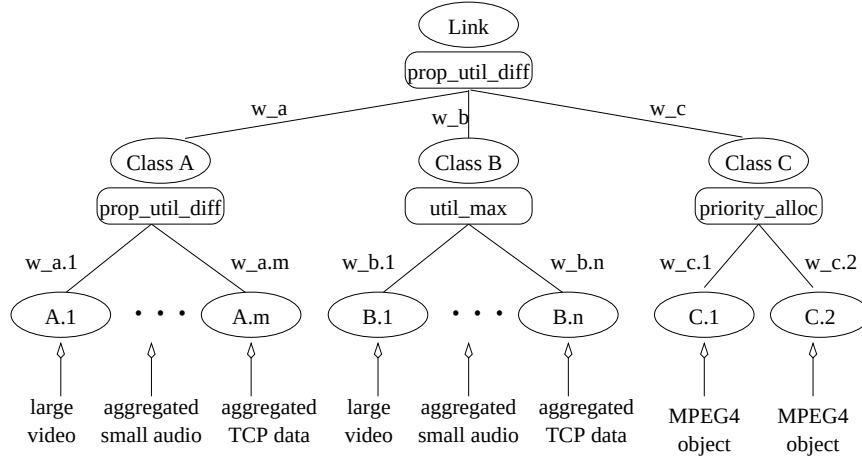


Figure 3-5: Example Structure of U(x)-CBQ Link Sharing Server

U(x)-CBQ supports a hybrid of utility-based allocation algorithms, that is, the allocation modules shown in Figure 3-5 could implement different algorithms. Figure 3-5 demonstrates one useful example of hybrid utility-based allocation. The hybrid allocation policy is motivated by the different behavior of adaptive and non-adaptive applications, as well as a diverse range of service types. In this example, the root CBQ class uses the proportional utility-differentiated allocation with differentiation parameter β_i chosen to reflect the service plan and monthly charge of

agencies (i.e., subscribers) A, B and C (e.g., DiffServ [35] gold, silver and bronze service classes).

Class A classifies flows into three application types (discussed in Section 2.3.), namely, TCP aggregates, aggregates of a large number of small-sized non-adaptive applications, and individual large-sized adaptive video applications. The TCP aggregates could be further classified into two aggregates for intra- and inter-core networks, respectively. We use the proportional utility-differentiation policy and give video/audio applications a large β . TCP aggregates receive a smaller bandwidth allocation but are not starved of bandwidth. In addition, the property of CBQ will also make sure that any unused video/audio bandwidth within Class B can be used by TCP aggregates. Class B uses utility-maximizing allocation. It benefits video/audio applications because a small change in available bandwidth will only cause one flow's allocation to be affected, thus limits the impact of rate oscillation impacting multimedia applications. However, this allocation policy has the down-side of potentially starving some applications. Class C uses the priority allocation algorithm to support intrinsic priority semantics among MPEG-4 elementary streams [1] corresponding to video objects with different priority, (e.g., low priority video object will be dropped first before a high priority object is scaled down).

3.5.2 Simulation Results

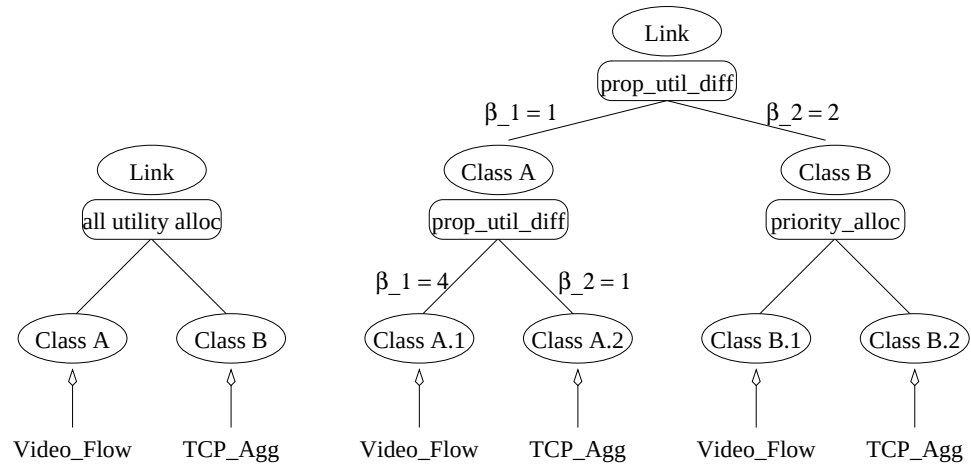
We evaluate the utility-based allocation algorithms using the ns simulator with built-in CBQ and DiffServ modules [99]. We use the CBQ formal link-sharing rules, as described in [40]. We choose the weighted round robin (WRR) algorithm as the scheduler for CBQ because the service weight of each class provides a clean interface for utility-based allocation algorithms to program CBQ. In our simulation, we choose a small buffer size (i.e., 1-2 packets) for every leaf class to ensure the

correct operation of the CBQ WRR scheduler. The choice of a small buffer effectively turns the shaping function of CBQ into policing function. For leaf CBQ classes, we use high priority for video/audio applications and low priority for TCP data aggregates. Unless otherwise stated, we use the default values found in the standard ns release for simulation parameters.

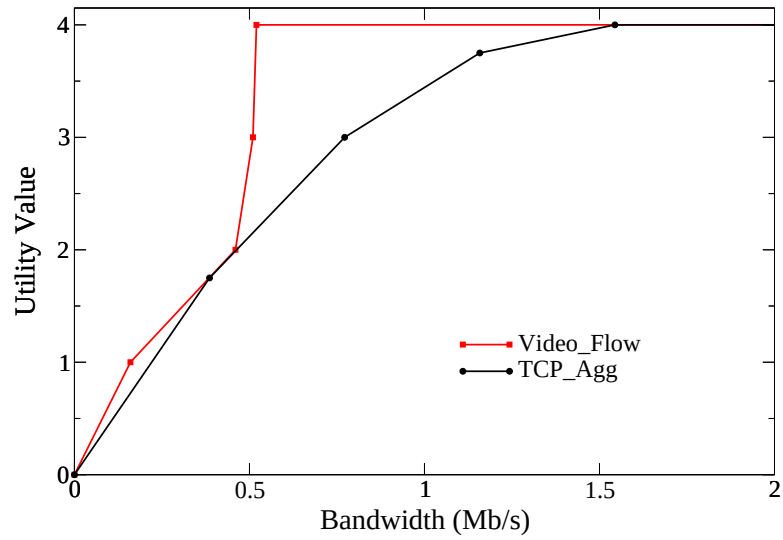
The goal of our simulation is to evaluate the effect of different utility-based policies in handling a combination of adaptive and non-adaptive applications. Because the packet level performance of CBQ has been widely studied in the literature, our simulation will focus on adjusting the sharing weights under time-varying link capacity.

The simulated link sharing structure is shown in Figure 3-6(a). In case 1, we use one level of link sharing among two classes to study the effect of various foundation utility-based allocation algorithms. In case 2, we introduce two levels of link sharing hierarchy among four classes, and investigate the hierarchical utility-based allocations.

The simulation uses two types of traffic sources as CBQ leaf classes. The Agg-TCP source models a TCP aggregate, and the Video-Flow source represents a large-sized video flow. We use the formula from Equation 2.14 to set the utility function for the Agg-TCP, where $r_{\max}$ is chosen as the T1 link speed of 1.544 Mb/s, and the minimum bandwidth requirement is zero ($r_{\min} = 0$). Consequently, the scaled utility function for TCP aggregates is set by $b_{\max} = r_{\max} - r_{\min} = 1.544$ Mb/s. In addition, we choose $u_{\max} = 4$. The utility function for the Video-Flow is measured from the Trace 1 MPEG1 video discussed in Section 2.5.2. To better illustrate the effect of different utility-based allocation algorithms, we scale up the bandwidth demand of the Video-Flow by ten times so that Video-Flow has the same order of bandwidth range as the Agg-TCP class. Both utility functions are shown

**Case 1****Case 2**

(a) Link Sharing Structure



(b) Individual Utility Function

Figure 3-6: U(x)-CBQ Link Sharing Simulation Setup

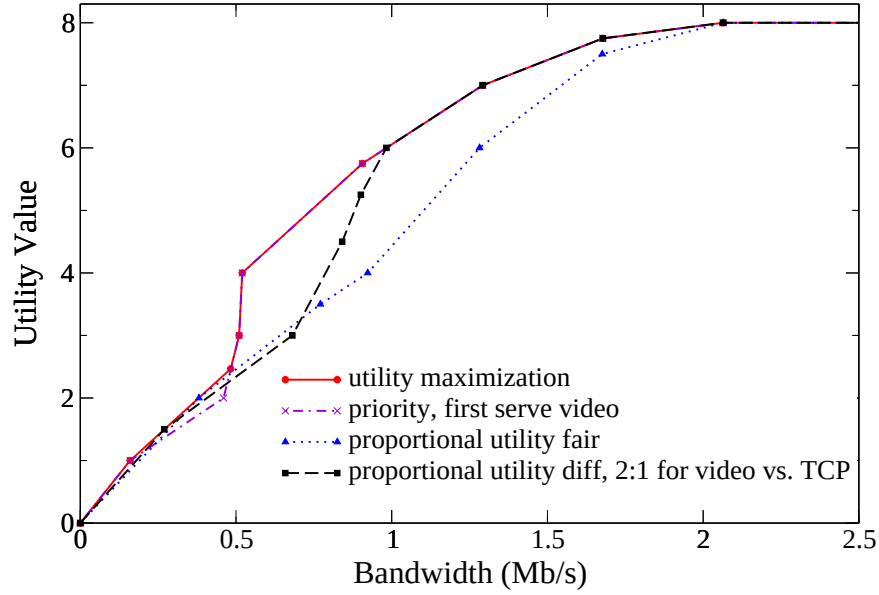


Figure 3-7: Aggregated Utility Function in Case 1

in Figure 3-6(b). In addition, we note that the utility functions shown in Figure 3-6(b) start from zero bandwidth because they represent the scaled utility functions defined by Equation (2.1).

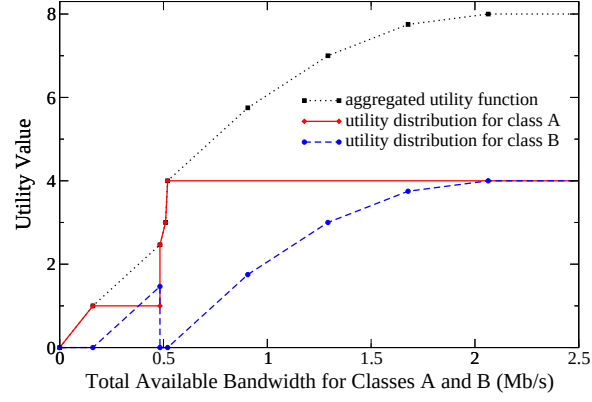
In the first simulation (i.e., case 1), we apply four different utility-based allocation algorithms to CBQ leaf classes A and B, where Class A has the Video_Flow utility function, and Class B has the Agg_TCP utility function. The four algorithms are utility-maximization, priority allocation for Class A, proportional utility fair, and proportional utility differentiation, where Class A receives twice the utility of Class B (i.e., $\beta_A : \beta_B = 2 : 1$).

Figure 3-7 and 3-8 show two different properties of utility-based algorithms, namely, utility maximization and fairness, respectively. In Figure 3-7, the aggregated utility functions for all four algorithms are shown. We observe that the aggregated utility function for utility-maximization represents the envelope of all other aggregated utility functions. The aggregated utility function for priority allocation

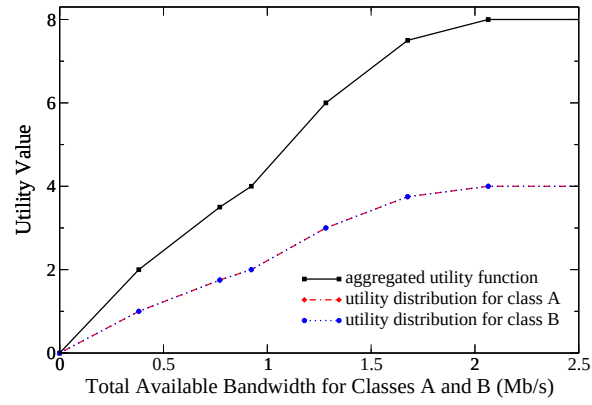
is a direct joint of individual utility functions of classes A and B, with Class A at the bottom because of its higher allocation priority. The aggregated utility function of proportional utility fair allocation lags behind in terms of maximizing total utility values.

In Figure 3-8, the utility values of allocated bandwidths for Classes A and B are plotted against the total available bandwidth. The intention is to illustrate the disparity in allocated utility between the two classes under different algorithms. Clearly, for fairness, the proportional utility fair allocation is preferable because the utility values for classes A and B are the same as shown in Figure 3-8(b). The proportional utility differentiated allocation gives exactly twice the amount of utility received by Class B to Class A (as shown in Figure 3-8(c)) until Class A reaches its maximum utility value of 4 at 0.983 Mb/s. For bandwidth larger than 0.983 Mb/s, Class B will get all the remaining utility until it reaches its maximum utility value. When we inspect the utility distribution of classes A and B under utility maximization (Figure 3-8(a)), the result indicates unfairness. Furthermore, the utility (and also bandwidth) allocation of one class is not strictly increasing as the total available bandwidth increases. This is demonstrated in the case of Class B by the sharp drop in utility to zero when total available bandwidth is between 0.483 and 0.52 Mb/s. Such a behavior could be fine for adaptive data applications such as TCP, but it would be undesirable for multimedia applications that prefer stable bandwidth allocation.

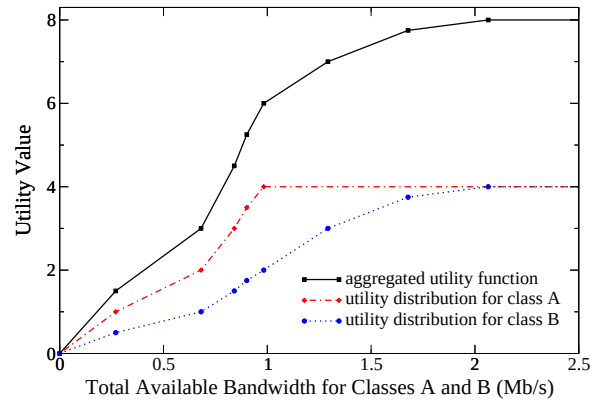
In the second simulation (i.e., case 2), we simulated the behavior of $U(x)$ -CBQ with a two-layer link sharing hierarchy, as shown in Figure 3-6(a). The leaf classes comprise two pairs of Video_Flow and Agg_TCP sources, representing the same as defined in the case 1. The algorithm used for classes A.1 and A.2 is proportional utility differentiation where Class A.1 video could get four times the utility of Class



(a) Utility Maximization



(b) Utility Fair



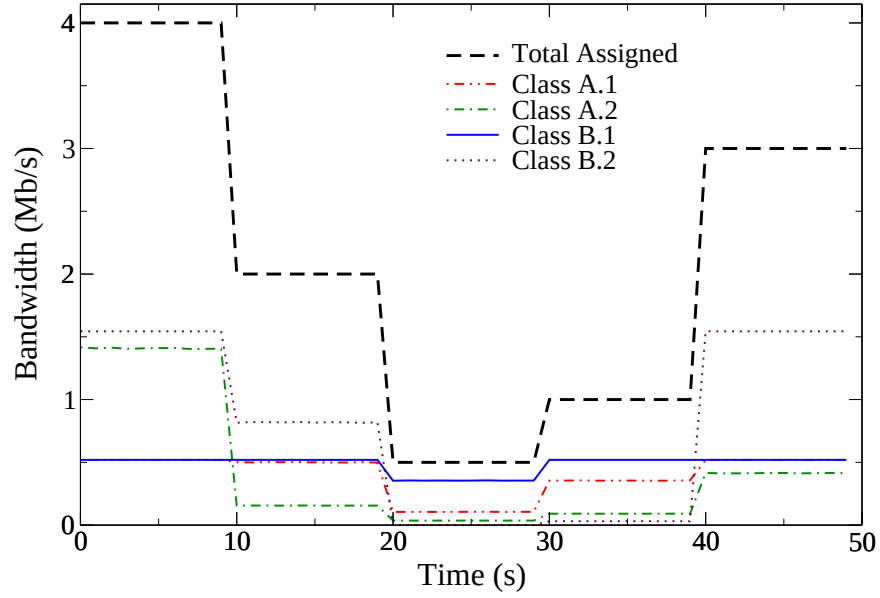
(c) Utility Differentiation

Figure 3-8: Utility Distribution in Case 1

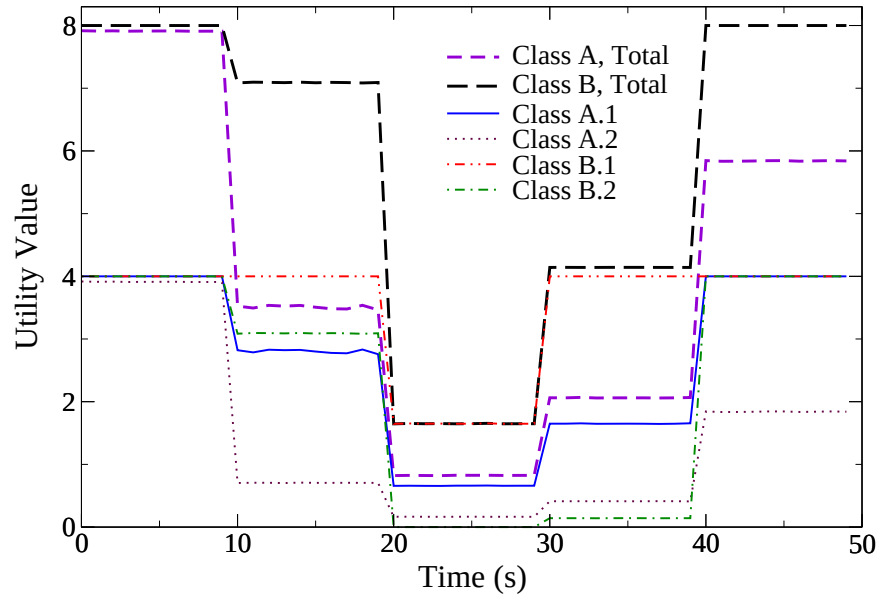
A.2 TCP flow. The purpose of this experiment is to give the video a more stable allocation. For the same reason, priority allocation is applied to Class B.1 and B.2 where priority is given to Class B.1 video. The top level allocation algorithm adopts proportional utility differentiation where $\beta_A : \beta_B = 1 : 2$. In this case, we simulate a “gold” service user (CBQ Class B) who receives twice the utility as a “silver” service user (CBQ Class A).

Following the same approach discussed in [40], we concentrate on the study of link sharing policies free from extraneous factors. A single constant-bit-rate source for each leaf class is used where each leaf has a peak rate higher than the link capacity. The packet size is set to 1000 bytes for TCP aggregates and 500 bytes for video flows. Under simulation, the link capacity changes every ten seconds. This sequence of changes drive the dynamic link sharing algorithms to adjust the link sharing weights for individual classes. The measured throughput and the corresponding utility value for each leaf class are shown in Figure 3-9(a) and (b), respectively.

Figure 3-9(a) shows the bandwidth allocation traces. The total available link bandwidth changes following the sequence of 4, 2, 0.5, 1, and 3 Mb/s every ten seconds. We observe the success of proportional utility differentiated and priority allocations in stabilizing the bandwidth allocation to video Class B.1 and A.1, respectively. Based on the link-sharing hierarchy settings, Class B.1 receives the best service because of the preferential treatment of its parent Class B, and the absolute priority over its peer Class B.2. The throughput of Class B.1 remains constant except between 20-30s when the total link capacity drops to 0.5 Mb/s. We also observe that the two B leaf classes receive more bandwidth than their Class A counterparts, respectively, which reflects the differentiated allocation between classes A and B. This differentiation behavior is better illustrated in Figure 3-9(b), which shows the utility allocation trace. In this case, we observe that the total utility received by



(a) Bandwidth Allocation Trace



(b) Utility Allocation Trace

Figure 3-9: Results for Link Sharing Case 2

Class B are exactly twice the total utility received by Class A between 10-40s. During the first and last 10s of the simulation trace, Class B reaches its maximum utility value of 8, and Class A receives the remaining utility allocation, which is more than half of the utility value of 8. We can also observe this differentiation effect between classed A.1 and A.2, where Class A.1 receives four times the utility of Class A.2. In Figure 3-9, we also observe that classes can be starved of resources due to priority allocation. During the period 20-30s, Class B.2 receives zero bandwidth and utility because of the strict priority given to Class B.1.

Through simulation, we verified the effectiveness of the $U(x)$ -CBQ algorithm. We demonstrated that bandwidth utility functions provide a rich programming tool to implement complex link sharing policies including application-aware and service-differentiated allocation rules. In general, two groups of utility-based allocation algorithms are used: utility-maximization and its variant of priority allocation as one group, and proportional utility-differentiation and a special case of proportional utility-fairness as the other group. Clearly, there is no argument that one policy is better than the other. The utility-maximization policy has a clear economic meaning of maximizing the total system welfare creating incentives for applications to cooperate. When bandwidth changes occur, utility-maximization allocation usually only adjusts the allocation of one flow, rather than all the flows, which is the case with proportional utility-fair allocation. As discussed, however, utility-maximizing and priority allocations could starve some flows. In addition, as discussed in Section 3.4., utility-maximizing allocation can suffer from complexity and scalability problems. Therefore, we recommend that the utility-maximizing allocation is more applicable to the lowest link sharing hierarchy for individual data flows where allocation starvation and state aggregation are not large concerns. In contrast, the proportional utility-differentiated allocation is simple and designed to support ser-

vice differentiation. Therefore, we recommend that utility-differentiated allocation should be applied to aggregated classes at the upper link sharing hierarchy of a single bottleneck link in addition to across networks. Network service providers are likely to adopt solutions that are based on hierarchical and hybrid structures (e.g., the structure shown in Figure 3-5). Such approaches should be capable of exploiting the benefits of different algorithms while limiting any adverse effects.

3.6. Summary

The contribution of this chapter is as follows. Our approach to network adaptation considered a set of foundation utility-based bandwidth allocation algorithms that realized fairness, service-differentiation and utility-maximization policies. These algorithms were efficiently designed with negligible overhead over the session arrival/departure time-scale. We applied these algorithms to augment the widely used CBQ [40] link sharing scheduler supporting hierarchical and hybrid allocation policies. Our simulation used measured video utility functions and formulated TCP utility functions to demonstrate application-aware and service-differentiated link sharing policies. Such a utility-based adaptation model would allow future content providers to offer equalized quality, differentiated allocation or priority allocation to networked-users. In the next chapter we extend the work presented in this chapter and study how to support mobile users in edge-based wireless access networks.

Chapter 4

Utility-Based Adaptation in Wireless Access Networks

4.1. Introduction

A key goal of next-generation wireless systems is to enable mobile users to access and distribute audio, video and data anytime anywhere. However, the support of multimedia services over wireless networks presents a number of technical challenges including physical layer impairments, mobility and bandwidth mismatch between wireline and wireless networks. These characteristics result in the delivery of time-varying QOS to mobile applications. In many cases, mobile applications are not designed to operate successfully under such conditions. In this chapter, we argue that future mobile systems should be capable of capturing and supporting application-specific adaptation characteristics in a flexible fashion. Many existing mobile network systems (e.g., Mobile IP and 3G cellular systems), however, lack the architectural flexibility to accommodate application-specific adaptation needs in time-varying mobile environments. In particular, most network resource allocation mechanisms rely on end-systems to declare QOS requirements such as bandwidth, delay and delay jitter. This approach can lead to frequent renegotiation between

end-systems and the network during periods of adaptation resulting in poor scalability when the number of flows or traffic aggregates grows or adaptation becomes frequent as in the case of wireless and mobile environments.

Unlike end-system oriented approaches, network-based adaptation faces a number of additional challenges. First, network-based adaptation is intrinsically more complex than end-system oriented approaches. In order to maintain a balance between architectural flexibility and scalability, we propose a split-level approach that support application-independent and application-specific adaptation needs. We argue that support for common adaptation demands should be managed by network mechanisms in order to optimize efficiency, while support for application-specific adaptation needs should be handled by a flexible platform at the network edges. Second, network control needs to be extended in an efficient manner to support common adaptation requirements, which can be characterized as having two dimensions; that is, the bandwidth granularity and the time-scale over which adaptation occurs. Network resource allocation schemes, however, are more complex than the case of a single link because one flow's allocation can be affected by other flows sharing a portion of a multi-hop route. *Max-min* fairness [52] is the most widely used fairness criterion found in bandwidth allocation algorithms for networks. Here, the idea is to maximize the allocation of flows with the least allocation; that is, to allow a flow to increase its allocation provided that the increase does not subsequently cause a decrease in allocation of a flow holding a lower or equal bandwidth allocation [10]. A new challenge is to extend max-min fairness to support adaptation in an efficient and scalable manner.

The contribution of this chapter is as follows. We present the design and evaluation of a utility-based adaptation framework for wireless packet access networks comprising *bandwidth utility functions* and *adaptation scripts*. Bandwidth utility

functions capture the adaptive nature of mobile applications in terms of the range of bandwidth over which applications prefer to operate. Adaptation scripts complement bandwidth utility functions by capturing application specific “adaptation time-scales” and “bandwidth granularities”. The utility-based adaptation framework supports both generic network adaptation control and flexible application-specific adaptation. In this sense, we use bandwidth utility functions to formulate a generic model for network adaptation, and deploy adaptation scripts to satisfy individual application needs. Our framework is split into two levels that support network-level utility-based allocation and application-level policy-based adaptation, respectively. At the network-level, we present an efficient extension of max-min fair allocation to support *utility-based max-min fairness*. A distributed algorithm periodically probes the wireless packet network on behalf of mobile devices maintaining their bandwidth allocations. Application level adaptation control employs adaptation handlers at mobile devices that are capable of programming a wide variety of flow adaptation behavior using adaptation scripts.

The structure of the chapter is as follows. In Section 4.2. we discuss related work in the area of adaptive resource control for wireless networks. Following this, in Section 4.3., we describe a utility-based adaptation framework for wireless packet access networks. In Section 4.4., we present a detailed description of our *utility-based network control* algorithm that realizes utility-based max-min fairness. Following this, in Section 4.5., we discuss our *policy-based application adaptation* scheme that works in unison with a network control algorithm to support a set of application-specific adaptation policies. In Section 4.6., we present our simulation results. We show that our framework is capable of supporting a wide range of adaptation needs under various network conditions. We conclude the chapter in Section 4.7. with some final remarks and present the pseudo-code for the utility-fair max-min algorithm in

the Appendix.

4.2. Related Work

Recently, there have been a number of architectural proposals for adaptive services in mobile networks [56, 67, 77]. In [73], Lu and Bharghavan present a set of admission control and reservation mechanisms that extend generic resource renegotiation to wireless and mobile environments. In [67], utility functions are proposed for network resource management. However, there is no discussion concerning specific mechanisms. Our approach to realizing adaptation policies differs from end-system oriented approaches (e.g., the Odyssey Project [79] and the adaptation proxy work discussed in [41]). In order to maintain a balance between architectural flexibility and scalability, we propose a split-level approach that supports application-independent and application-specific adaptation needs. We argue that support for common adaptation demands should be managed by network mechanisms in order to optimize efficiency, while support for application-specific adaptation needs should be handled by a flexible platform at the network edges. For example, a “smooth” adaptation policy can have the same effect as the end-system playout-control mechanism discussed in [88]. However, with the utility-based framework, we can deliver rate-smoothing features as a generic network service which benefits a wide variety of applications including TCP.

There have been several proposals by the ATM Forum to extend the notion of max-min fairness (e.g., the case of non-zero minimum cell rate and allocation proportional to weights [47, 102]). In this chapter, we formalize the utility-based max-min fair allocation and investigate system issues associated with protocol design and adaptation policy, including algorithm and protocol scalability to support aggregated flow states. A similar work [20] was published after our publication of

utility-based max-min fair algorithm [69].

In [68], feedback control theoretic mechanisms (e.g., control based on the Proportional-Integral-Differential of the feedback signal) are used to direct QOS adaptation in networks. We observe that while control theory based approaches are better than heuristic and measurement based approaches, it is hard to apply the theory directly to traffic aggregates without obtaining explicit resource requirements. In our framework, we attempt to reduce the reliance on real-time signaling of application resource requirements by using utility functions to model a range of requirements in advance. In addition, we develop a platform comprising a resource probing protocol and adaptation handlers that supports flexible adaptation policies in a scalable manner. By pushing application-specific adaptation policy into a set of edge-based programmable adaptation handlers we can relieve the network control system of the burden of supporting individual adaptation profiles in the network.

4.3. Utility-Based Adaptation Model for Wireless Access Networks

Figure 4-1 illustrates our utility-based adaptation model for wireless access networks. At the top of the figure, an adaptive service applications programming interface (API) allows end users and service providers to program the underlying control mechanisms. These mechanisms include network-level and application-level adaptation control, which we refer to as utility-based network control and policy-based application adaptation, respectively.

4.3.1 Utility-Based Network Control

We assume a two-tier network model, where the global Internet provides inter-connectivity to a set of wireless packet access networks through gateways, as shown

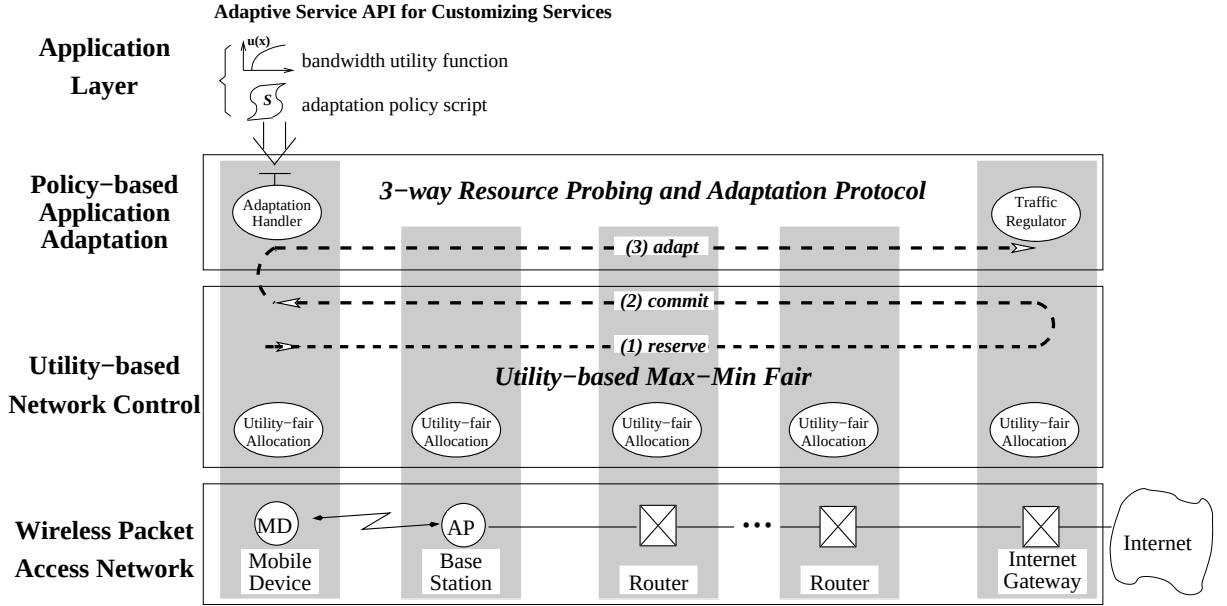


Figure 4-1: Utility-based Adaptation Model for Wireless Access Networks

in Figure 4-1. An example of the wireless packet access network is a Cellular IP [101] network. The Cellular IP access network realizes micro-mobility in support of fast handoff and paging, comprising a set of base stations, routers and gateways. In contrast, Mobile IP enables support for macro-mobility between gateway nodes. Cellular IP is based on per-host routing where the routing state is stored at base stations and gateways. Routing state is maintained by data and paging-update packets flowing between a mobile device and its designated gateway. We augment per-mobile state to include bandwidth allocation and adaptation control information for per-mobile traffic aggregates. A per-mobile traffic aggregate is a state variable that represents all uplink and downlink traffic between a mobile device and its corresponding Internet gateway.

A periodic bandwidth reservation and adaptation protocol is used to allocate and maintain traffic aggregate reservations in a Cellular IP access network. The protocol operates in three phases, as shown in Figure 4-1. The first two phases called *reserve (1)* and *commit (2)* are part of the network control scheme that per-

forms utility-based max-min fair bandwidth allocation (discussed in Section 4.4.2) in a distributed manner. The last phase called *adapt (3)* is associated with the application adaptation control scheme.

The reserve and commit messages operate in a similar manner to the RSVP [31] protocol. However, only unicast traffic aggregates are supported. Bandwidth reservation messages (*reserve*) are periodically sent from a mobile device toward the gateway for both uplink and downlink traffic aggregates. This probing mechanism periodically refreshes “soft-state” bandwidth reservations that are associated with a traffic aggregate path between the mobile device and gateway. The state is “soft” because it is removed after a time-out interval if the state is not reset (i.e., refreshed). Reserve messages interact with a set of utility-fair allocation mechanisms en-route between the mobile device and gateway, as illustrated in Figure 4-1. This probe drives the utility-based max-min fair allocation based on the aggregate bandwidth needs of all flows in a traffic aggregate. A gateway responds to a *reserve* message by sending a *commit* message back to the appropriate mobile device. This action commits resources allocated by the *reserve* message to a traffic aggregate over the next probing interval.

Utility-fair allocation operates locally at each node (e.g., base station, router and gateway) and provides explicit support for common bandwidth adaptation needs at the network level. In addition, the reserve/commit probe efficiently implements the utility-based max-min fair allocation in a distributed manner. This reservation/adaptation mechanism operates over a slow time-scale in the order of seconds, or during handoff or flow renegotiation. Fast time-scale control is needed over the wireless hop, as discussed in our previous work [12].

4.3.2 Policy-Based Application Adaptation

As shown in Figure 4-1, we design an additional control layer on top of the utility-based network control. Both bandwidth utility functions and adaptation scripts represent the applications programming interface (API) at this layer for customizable adaptive service model. Software agents called “adaptation handlers” execute at mobile devices and implement application-specific adaptation scripts that are capable of operating on a per-application or per-service class basis.

The addition of adaptation scripts is fundamental to the usability of the architecture. For adaptive mobile applications perceptible quality is strongly related to how and when they respond to bandwidth availability. While a utility function abstracts an application’s resource needs, an adaptation script is central to capturing the application-specific responses to resource availability in terms of adaptation time-scale and bandwidth granularity; that is, what time-scale and/or events should trigger an increase in bandwidth allocation and by how much.

In essence, a utility function and adaptation script capture an application’s blueprint for adaptation. Collectively, these form the semantics of application-specific adaptive services in wireless networks. Using adaptation scripts, our utility-based framework presents a comprehensive programmable environment for customizing adaptive services. We have designed four types of adaptation scripts (viz. greedy, discrete, smooth, and handoff adaptations) for experimentation that covers a wide set of application adaptation needs. Additional policies can be implemented by programming new scripts, as discussed in Section 4.5..

Application adaptation is performed as part of the last phase of the resource probing and adaptation protocol. After receiving a *commit* message, an adaptation handler located at a receiving mobile device executes its adaptation script, which could be in the form of a default script provided by the system, or a script defined

by a user or application service provider. The schema of the adaptation script is shown below:

-
- 1) retrieve the allocated bandwidth value
in the *commit* message from the network;
 - 2) make adaptation decisions;
 - 3) calculate the consumed bandwidth to be
no greater than the allocated one;
 - 4) send an *adapt* message with the value of
consumed bandwidth toward the gateway.
-

Figure 4-2: Simple Adaptation Script Schema

Based on the *commit* message, the *adapt* message confirms the final committed bandwidth (i.e., the consumed bandwidth) after taking the application's adaptation policy into account. As outlined in Figure 4-2, the consumed bandwidth can be less than or equal to the allocated bandwidth presented at the receiver in the *commit* message. The *adapt* message also notifies a traffic regulator located at the gateway if there is any change to the packet policing/shaping functions, and/or media scaling and packet filtering functions that may operate on the downlink traffic.

It is noted that two steps are sufficient for the reservation/adaptation mechanism for either uplink or downlink traffic aggregates respectively. In this case, the commit and adapt steps can be merged into one. However, the 3-step mechanism is necessary to use the same messaging protocol to manage both uplink and downlink simultaneously.

Our approach keeps the interior of the wireless packet access network simple. With this reservation/adaptation mechanism, any traffic overload observed inside the wireless access network will be controlled at the network edges (i.e., at mobile devices and gateways). By locating adaptation handlers at the edge, we relieve internal routers of supporting adaptation functions.

4.4. Utility-Based Network Control

Based on service-specific utility functions multiple flows can be represented by an aggregated utility function in the wireless packet access network achieving utility-based fairness. In what follows, we extend the per-hop utility-based bandwidth allocation rule presented in Section 3.3.1 to cover a max-min fairness criterion across multiple hops between mobile devices and border gateways in wireless access networks. This algorithm is driven by the resource probing protocol introduced in Section 4.3. and detailed in this section.

4.4.1 Definition of Utility-Based Max-min Fairness

First, we define the feasibility constraint which specifies that any allocation must not allocate more bandwidth than a link's total capacity $\mathcal{B}^l$. The formal definition is:

Definition 1 *A bandwidth allocation vector $\beta = \langle \beta_1, \dots, \beta_n \rangle$ is **feasible** if for each flow $i \in \mathcal{N}_n$, $\beta_i \geq b_{i,1}$ and for each link l , $\sum_{i \text{ passing link } l} \beta_i \leq \mathcal{B}^l$.*

Definition 2 *An allocation vector β is **utility-based max-min fair** if it is feasible and for each flow $i \in \mathcal{N}_n$, its allocation β_i cannot be increased while maintaining feasibility without decreasing some flow j 's allocation β_j , where $u_j(\beta_j) \leq u_i(\beta_i)$.*

Definition 3 *A link l is a **utility-based bottleneck link** with respect to a given feasible allocation vector β for a flow i crossing l if l is saturated, i.e., $\sum_{i \text{ passing link } l} \beta_i = \mathcal{B}^l$, and $u_i(\beta_i) \geq u_j(\beta_j)$ for all the flows j crossing l .*

These definitions are similar to the max-min fairness definition in [10]. The main change is to substitute the comparison in bandwidth values (e.g., $\beta_i \geq \beta_j$) to a comparison in corresponding utility values (e.g., $u_i(\beta_i) \geq u_j(\beta_j)$). The utility-based max-min fairness also has the same properties as conventional max-min fairness.

Proposition 8 *A feasible allocation vector β is utility-based max-min fair if and only if each flow has a utility-based bottleneck link with respect to β .*

The proof follows the same procedure as shown in [10] except the change from bandwidth to utility value. This property implies that under utility-based max-min fair allocation, each flow has one bottleneck link. Therefore, we will mark a flow *bottlenecked* at its only bottleneck link and *satisfied* at all the other links in the path.

Proposition 9 *There exists a unique allocation vector that satisfy utility-based max-min fair rate allocation.*

The proof of Proposition 9 is to first construct one allocation vector satisfying utility-based max-min fairness. One may use the centralized allocation algorithm in [10] by substituting bandwidth with the utility value. Following this, one can construct a proof by contradiction by showing that any other allocation vector satisfying utility-based max-min fairness will lead to a violation of Definition 2.

Because a utility function has the notion of the minimum sustained rate and peak rate for a flow, the utility-based max-min fairness intrinsically captures the constraints on minimum and maximum rate. With the additional properties such as simplicity and consistency under flow aggregation, this extended max-min fairness criterion can be practically implemented using distributed algorithms.

4.4.2 Distributed Algorithm

With the advent of available bit rate (ABR) flow control found in ATM networks, distributed algorithms that emulate a centralized max-min fair allocation algorithm have been proposed [22, 55]. In [22], Charny proposes a distributed and asynchronous algorithm for max-min fair allocation. At each bandwidth allocation iteration, a two-step algorithm is used to partition flows into *bottlenecked* and *satisfied*

sets, denoted by $\mathcal{U}^l$ and $\mathcal{L}^l$, respectively. The fair allocation for bottlenecked flows at link l is calculated as

$$x_i = \frac{\mathcal{B}^l - \mathcal{B}_{\mathcal{L}}^l}{\sum_{j \in \mathcal{U}^l} 1} = \frac{\mathcal{B}^l - \mathcal{B}_{\mathcal{L}}^l}{|\mathcal{U}^l|} \quad (4.1)$$

where $\mathcal{B}_{\mathcal{L}}^l \triangleq \sum_{i \in \mathcal{L}^l} x_i$ is the total allocated bandwidth to $\mathcal{L}^l$, the set of satisfied flows.

We introduce a simple extension that replaces Equation (4.1) with the following derived from (3.3)

$$x_i = u_i^{-1}(u_i^{\max} \cdot v_{agg, \mathcal{U}}^l(\mathcal{B}^l - \mathcal{B}_{\mathcal{L}}^l)) \triangleq \mathcal{F}(\mathbf{u}_i, \mathbf{u}_{\mathcal{U}}^l, \mathcal{B}^l - \mathcal{B}_{\mathcal{L}}^l) \quad (4.2)$$

where $v_{agg, \mathcal{U}}^l()$ is the normalized aggregated utility function as defined in Equation (3.1). The composite function from the inverse utility function $u_i^{-1}()$ and the aggregated utility function $v_{agg, \mathcal{U}}^l()$ captures the notion of weighted fairness.

In Equation (4.2), we use a new notation $\mathcal{F}(\mathbf{u}_i, \mathbf{u}_{\mathcal{U}}^l)$ for the allocation function to improve clarity, with $\mathbf{u}_i$ denotes the piecewise linear utility function $u_i(x)$, and $\mathbf{u}_{\mathcal{U}}^l \triangleq u_i^{\max} \cdot v_{agg, \mathcal{U}}^l(x)$ denotes the aggregated utility function of $\mathcal{U}^l$ – the set of bottlenecked flows. In addition, we will use the operator $\oplus$ to denote the operation of utility-fair aggregation.

In [55], Kalampoukas simplifies Charny's iterative marking procedure (which has complexity of $O(n)$ for each iteration where n is the number of flows) to an $O(1)$ algorithm for each iteration. Instead of trying to partition all the flows at once, the algorithm only updates the bottlenecked/satisfied status of the flow currently in process. Our implementation is based on this efficient algorithmic approach.

Let us consider flow i at link l . If the flow is currently marked as satisfied, to update its allocation, we reset its marking to bottlenecked and aggregate its utility function $\mathbf{u}_i$ into $\mathbf{u}_{\mathcal{U}}^l$, i.e., $\mathbf{u}_{\mathcal{U}}^l = \mathbf{u}_{\mathcal{U}}^l \oplus \mathbf{u}_i$. Subsequently, an abstract view of the

bandwidth assignment rule is given as follows:

$$x_i = \max \left\{ \mathcal{F}(\mathbf{u}_i, \mathbf{u}_{\mathcal{U}}^l, \mathcal{B}^l - \mathcal{B}_{\mathcal{L}}^l), \mathcal{F}(\mathbf{u}_i, \mathbf{u}_{\mathcal{U}}^l \oplus \mathbf{u}_k, \mathcal{B}^l - \mathcal{B}_{\mathcal{L}}^l + x_k) \right\}, \quad (4.3)$$

where $k = \arg \max_{j \in \mathcal{L}^l} \{u_j(x_j)\}$.

Here k is the index of a flow that has the maximum utility value inside the satisfied set $\mathcal{L}^l$. It is possible that during a transient phase a flow in the satisfied set could have a utility value greater than the utility of a bottlenecked flow. Therefore, the purpose of adding flow k into the bandwidth allocation pool is to reduce the allocation oscillation experienced during the transient phase [55]. However, adding flow k , the satisfied flow with the maximum utility value, into the allocation pool could generate a larger allocation for a bottlenecked flow in this case. In some cases, this allocation may violate the feasibility constraint. We will solve this problem in the next section.

4.4.3 Resource Probing Protocol

In what follows, we outline a resource probing protocol which constitutes the first two phases (i.e., the reserve and commit phases) of the three-way protocol described in Section 4.3.. The probing protocol operates in the wireless access network and periodically and asynchronously probes the wireless access network on a per traffic aggregate basis. In this instance, the bandwidth requirement of an individual application is derived from its corresponding aggregated utility function based on Equation (4.2).

The protocol operates on a slow time-scale in the order of seconds, and drives bandwidth renegotiation in the access network. In contrast, the bandwidth renegotiation time-scale for ATM ABR flow control algorithms is in the millisecond range. The main component affecting the convergence time is the probing interval, which,

in our case, is several orders of magnitude greater than the round-trip delay¹.

To reduce the convergence time of the probing scheme by half, we exploit the backward signaling message (i.e., *commit* message) to commit the reservation made by the forward *reserve* message. Each *reserve* probe message contains four parameters: (i) the mobile traffic aggregate identifier; (ii) the ideal bandwidth request ρ_i^{ideal} ; (iii) the actual bandwidth request ρ_i^{actual} ; and (iv) the utility function vector $\mathbf{u}_i$ if it has been changed since the last resource probe. The corresponding *commit* message contains four parameters as well: (i) the mobile traffic aggregate identifier; (ii) the ideal bandwidth allocation r_i^{ideal} ; (iii) the actual bandwidth allocation r_i^{actual} ; and (iv) the updated utility function vector $\mathbf{u}_i$ for the whole mobile traffic aggregate to be confirmed along the path.

To speed up the convergence time while maintaining the feasibility constraint (Definition 1), two parallel sets of reservation state $(\rho_i^{ideal}, r_{i,l}^{ideal})$ and $(\rho_i^{actual}, r_{i,l}^{actual})$ are maintained along the route, where ρ denotes the requested bandwidth value in the *reserve* messages, and r denotes the confirmed bandwidth value in the *commit* messages. $(\rho_i^{actual}, r_{i,l}^{actual})$ tracks the actual allocation under the feasibility constraint, and $(\rho_i^{ideal}, r_{i,l}^{ideal})$ tracks the ideal allocation without the feasibility constraint² and ensures convergence to utility-based max-min allocation.

For a mobile traffic aggregate i , the states stored at each link l include $\mathbf{u}_i, r_{i,l}^{ideal}, r_{i,l}^{actual}$, and a flag $S_{i,l} \in \{\text{bottlenecked}, \text{satisfied}\}$. The $S_{i,l}$ flag implicitly partitions the mobile traffic aggregates into two sets $\mathcal{U}^l$ and $\mathcal{L}^l$. The aggregated states stored at each link l comprise the total available bandwidth $\mathcal{B}^l$, the in-use bandwidth for the set $\mathcal{L}^l$: $\mathcal{B}_{\mathcal{L}}^l$, and the unused bandwidth $\mathcal{B}_{free}^l$. The algorithm also maintains state for an

¹The round-trip delay is the time between sending a *reserve* message and receiving the corresponding *commit* message.

²The feasibility constraint could be violated during transient phases as flows asynchronously update their bandwidth allocations. Tracking ideal allocation allows each flow to converge faster toward the ideal allocation value.

aggregated utility function $\mathbf{u}_{\mathcal{U}}^l$.

The allocation rule follows from Equation (4.3). To reflect the two allocation algorithms operating in parallel, the ideal and actual allocations of a traffic aggregate are calculated based on the following:

$$\begin{aligned} \text{ideal : } r_{i,l}^{ideal} &= \min\{x_{alloc}^l, \rho_i^{ideal}\} \\ \text{actual: } r_{i,l}^{actual} &= \min\{x_{alloc}^l, \rho_i^{actual}, \mathcal{B}_{free}^l\} \end{aligned} \quad (4.4)$$

After the allocation steps given by (4.3) and (4.4) are complete, the S_i^l of the traffic aggregate is changed if necessary. In this case, the state variables are adjusted when the *commit* message arrives to relax any over-allocated bandwidth. Because we follow the approach of [55] not to adjust the states of other traffic aggregates, the algorithm complexity is in the order of $O(K)$, where the utility vector size K is the number of critical utility levels.

In what follows, we present the convergence property of the algorithm and simulation results to verify this property.

4.4.4 Convergence Property

The analysis of our algorithm's convergence property follows closely from [23]. It is shown in [23] that a distributed algorithm needs at least M iterations to stabilize toward max-min allocation in a descending order starting from the most congested bottleneck link, where M is the number of distinct bottleneck levels in the network. Each iteration of our algorithm only requires one probing round without the feasibility constraint because we utilize the *commit* message and the explicit rate information carried in the probing protocol as described in the preceding section. This is not the case with the ABR rate allocation algorithm, which requires three messages to stabilize a change in bandwidth allocation and one more message to

notify the next level bottleneck links along the route.

Denote T a probing interval and RTT as the longest round-trip delay for the signaling message (i.e., reserve/commit) in the access network. Then the amount of time required is $T + RTT$. When we consider the feasibility constraint, one more round of probing is required to allow the actual allocation to reach the ideal allocation. This adds a factor T to the time required.

Proposition 10 *Utility-based max-min fair allocation algorithm converges in $RTT/2 + (2T + RTT)M$, and in $RTT/2 + (T + RTT)M$ without the overload reduction constraint.*

The $RTT/2$ factor is attributed to the case where change is caused by a newly arriving flow or a flow changing its utility function. In this case, it takes $RTT/2$ for the intermediate routers along the path to be updated accordingly.

In practice, the convergence upper-bound can be improved by network engineering. One approach is to use a centralized implementation of the bandwidth allocation algorithm. This effectively removes the M factor from consideration. Another approach taken in wireless networks is to over-provision the wireline part of the cellular network such that only the base stations can be possible bottlenecks. In both schemes the allocation algorithm located at internal access network nodes can be disabled. However, the algorithm is still needed at border gateways to support edge traffic control. It is clear that using a small probing interval T can significantly reduce the convergence time. However, reducing the probing interval will increase the signaling traffic load on the system. A compromise scheme could use variable probing intervals on different portions of the network (e.g., a smaller value across the wireless hop and larger one in the wireline access network) with the result of improving bandwidth usage.

4.5. Policy-Based Application Adaptation

Adaptation policies capture different application behavior in a flexible and customizable manner. A TCP application, for example, may want to instantly take advantage of any resource availability. On the other hand, mobile multimedia applications may prefer to follow trends in bandwidth availability to avoid frequent oscillation in utility level rather than respond to instantaneous changes in available bandwidth. Typically, following trends in this fashion leads to more stability in the user’s perceived quality. In contrast, an adaptation script that responds to instantaneous changes can lead to fast time-scale oscillations (“flip-flopping”), which may be perceived as undesirable by many users. We have designed a number of adaptation scripts representing some common adaptation policies that mobile application can select from; these include:

- *greedy adaptation*, which allows applications to instantly move up along their utility functions when bandwidth becomes available to satisfy any point on their utility curves;
- *discrete adaptation*, which allows applications to move up along step or staircase shaped utility functions, rounding off the assigned bandwidth to the lower discrete bandwidth level;
- *smooth adaptation*, which allows applications to move up along their utility functions only after a suitable damping period has passed; and
- *handoff adaptation*, which allows applications to moves up along their utility functions only after a handoff event has occurred.

While these canned policies have been pre-defined, our approach is open to supporting new policies (e.g., a hybrid of smooth and handoff policies). To facilitate

the introduction of customizable adaptive mobile services, we allow programming and dynamic loading of application-specific adaptation scripts. An abstract view of the adaptation scripts is the **commit**($j, \rho_j^{ideal}, \rho_j^{actual}$) function, where ρ_j^{ideal} and ρ_j^{actual} are the assigned ideal and actual allocations for application j , respectively. Their values are derived from the assigned ideal (ρ^{ideal}) and actual (ρ^{actual}) allocation for traffic aggregate in the *commit* message using Equation (4.2). Based on the adaptation script, the **commit**($j, \rho_j^{ideal}, \rho_j^{actual}$) function determines the consumed bandwidth $\sum_{j \in \text{traffic aggregate}} \rho_j^{ideal}$ and $\sum_{j \in \text{traffic aggregate}} \rho_j^{actual}$ to be returned in the *adapt* message.

Mobile devices are free to change their adaptation policies at any time because the adaptation handlers are implemented locally at mobile devices. This allows users to dynamically respond to the needs of particular applications or dynamic QOS conditions experienced in the wireless access networks. For example, the default policy for a particular flow could be set to a greedy script. After a period of time the mobile device may assert a new script for flows based on the measured conditions. Instantiating a new adaptation script may be driven by a particular stability test related to mobility movement, or the observed performance of an application (e.g., changing to smooth adaptation in cells where the observed QOS oscillates frequently). In this respect, adaptation handlers represent programmable objects that can be tailored to meet application specific service needs under time-varying channel conditions.

4.5.1 Greedy Adaptation Script

The greedy adaptation script is the default adaptation script used for all new mobile traffic aggregates that do not instantiate adaptation handlers. Flows are greedy in the sense that they will accept whatever bandwidth is offered by the network at any instance. A greedy adaptation handler's **commit**($j, \rho_j^{ideal}, \rho_j^{actual}$) function simply

accepts the ρ_j^{ideal} and ρ_j^{actual} values derived from a *commit* message granted by the network and returns them in an *adapt* message.

```

commit( $j, \rho_j^{ideal}, \rho_j^{actual}$ ) {
    adapt( $\rho_j^{ideal}, \rho_j^{actual}$ );
}

```

Figure 4-3: Simple Greedy Adaptation Script

The choice of probing interval needs to balance the trade-off between a desired fairness behavior and the increase in signaling traffic that a short probing interval would bring. In Section 4.6., we study the effect of the probing interval on allocation accuracy.

4.5.2 Discrete Adaptation Script

Discretely adaptive applications require discrete increments of allocated bandwidth to support multi-layered data transport, (e.g., the transport used in [6], where the base and enhancement layers of MPEG2 flows receive different treatments at network congestion points). The goal of such a discrete adaptation script (as shown in Figure 4-4 Strategy A) is to enforce a complete increment/decrement on allocated bandwidth and avoid any partial changes. In our framework, discrete adaptation is specified by a discrete type of utility function (e.g., the staircase shape shown in Figure 2-1). A discrete shape utility function, however, only resides in adaptation handlers at mobile devices. The network bandwidth allocation algorithm and probing protocol only deal with strictly-increasing piecewise linear functions.

Unfortunately, Strategy A in Figure 4-4 can cause allocation disparity, as we will show in Section 4.6.. This disparity is general to adaptation policies with discrete bandwidth granularity. When the residual bandwidth is insufficient to accommodate all the flows (e.g., because of their bandwidth granularity) some flows will receive

```

// Strategy A
commit( $j, \rho_j^{ideal}, \rho_j^{actual}$ ) {
    // locate discrete utility value
     $m = \lfloor u_j(\rho_j^{ideal}) \rfloor, n = \lfloor u_j(\rho_j^{actual}) \rfloor$ ;
    // assign discrete bandwidth
     $\rho_j^{ideal} = b_{j,m}, \rho_j^{actual} = b_{j,n}$ ;
    adapt( $\rho_j^{ideal}, \rho_j^{actual}$ );
}

// Strategy B
commit( $j, \rho_j^{ideal}, \rho_j^{actual}$ ) {
    if(( $\rho_j^{actual} == \rho_j^{ideal}$ ) or ( $\rho_j^{actual} \geq prev\_ \rho_j^{actual}$ )) {
        // for  $\rho_j^{ideal}$ : discrete script
         $m = \lfloor u_j(\rho_j^{ideal}) \rfloor$ ;
         $\rho_j^{ideal} = b_{j,m}$ ;
    }
     $prev\_ \rho_j^{actual} = \rho_j^{actual}$ ;
     $n = \lfloor u_j(\rho_j^{actual}) \rfloor$ ;
     $\rho_j^{actual} = b_{j,n}$ ;
    adapt( $\rho_j^{ideal}, \rho_j^{actual}$ );
}

```

Figure 4-4: Two Discrete Adaptation Scripts

better treatment than others.

To resolve this fairness issue, the adaptation script should give a flow an opportunity to increase its bandwidth allocation regardless of its adaptation style. Under this approach the allocation disparity can be rotated among flows. Strategy B in Figure 4-4 presents one simple example. The algorithm switches to a greedy adaptation script when a reduction on the assigned bandwidth is detected. By doing so, a flow can register itself as a “bottlenecked” flow within the network and force any unclaimed portion of bandwidth to be released by other flows. This provides the flow with a fair chance to share the extra bandwidth resources. If a flow strictly follows a discrete adaptation script it usually is tagged as satisfied within the network, and hence loses its chance of sharing additional bandwidth. Strategy B detects the reduction of the assigned bandwidth by comparing the actual assigned bandwidth

with the ideal assigned bandwidth from the previous allocation.

4.5.3 Smooth Adaptation Script

Smooth adaptation suits the needs of the adaptive multimedia applications (e.g., *vic* and *vat*) that can continuously adapt their rate (e.g., by adjusting receiver-end playout buffers). Such applications require a script that supports a smooth change of rate in the delivered service so that, for example, the playout buffer does not underflow or overflow often. In essence, these applications prefer to follow trends in bandwidth availability as opposed to reacting to instantaneous changes that might be short lived. We describe this form of adaptation as “smooth” because the adaptation handler implements a low pass filter based on network assigned bandwidth. This type of application is supported within our framework by limiting the bandwidth increment and enforcing the minimum time interval between two consecutive bandwidth increments.

In what follows, we present one implementation of such an adaptation script. There are three control parameters in this script: δ denotes the maximum bandwidth (or utility value) increment that the application can tolerate; τ is the minimum interval between two consecutive allocation increments; and κ , is a filter factor. The pseudo-code for the smooth adaptation strategy is shown in Figure 4-5.

4.5.4 Handoff Adaptation Script

The final canned adaptation script deals with different adaptation strategies that can be adopted during handoff. Mobile applications may only want to deal with adaptation on the time-scale of handoff thereby limiting fluctuation in the observed service quality. Typically, these applications require uniform service while resident in a cell and prefer to deal with adaptation issues only during handoff. A number of

```

commit( $j, \rho_j^{ideal}, \rho_j^{actual}$ ) {
     $\rho_j^{ideal} = \text{calc\_bw}(\rho_j^{ideal}, prev\_x_j^{ideal});$ 
     $\rho_j^{actual} = \text{calc\_bw}(\rho_j^{actual}, prev\_x_j^{actual});$ 
    if ((no increment) or
        ( $time\_since\_last\_increment \geq \tau$ )) {
         $prev\_x_j^{ideal} = \rho_j^{ideal}, \quad prev\_x_j^{actual} = \rho_j^{actual};$ 
        reset timer;
    } else // do not change
         $\rho_j^{ideal} = prev\_x_j^{ideal}, \quad \rho_j^{actual} = prev\_x_j^{actual};$ 
    adapt( $\rho_j^{ideal}, \rho_j^{actual}$ );
}

calc_bw( $\rho, prev\_x$ ) {
    if ( $prev\_x == 0$ ) { // first time
        return  $0.5\rho$ ;
    } else if ( $\kappa\rho \geq prev\_x - \delta$ ) {
        // limit increment by  $\delta$  and  $\kappa$ 
        return  $\min\{\kappa\rho, prev\_x + \delta\};$ 
    } else {
        return  $\min\{\rho, prev\_x - \delta\};$ 
    }
}

```

Figure 4-5: Smooth Adaptation Script

adaptation scenarios are possible during handoff depending on the conditions found in a new cell and the bandwidth requirements of the mobile traffic aggregate being handed off. Figure 4-6 shows a number of possible outcomes for handoff adaptation illustrated in the experimental trace obtained from Mobware [6], a programmable mobile networking platform that operates over an experimental indoor pico-cellular testbed.

In this experiment, four mobile devices (M1, M2, M3 and M4) are handing off in sequential order (H1, H2, H3 and H4) from base station AP1 to AP2. Mobile device M1 enters the new cell at H1 and scales up its utility to take advantage of available resources. The M1 adaptation script only adapts after handoff. At point H2 in the trace the mobile device M2 hands off to the base station AP2 and is forced to scale down to its base layer. Mobile device M3 has an adaptation script

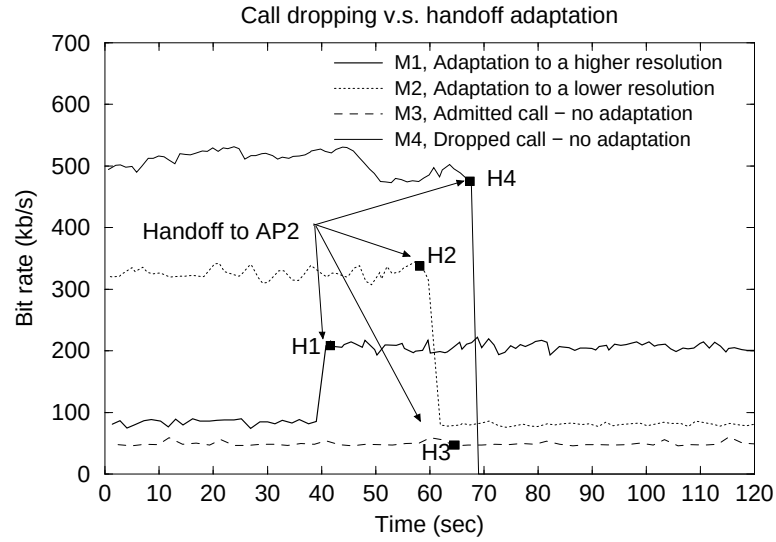


Figure 4-6: Handoff Adaptation Script Results

that never adapts. At H3 the mobile device M3 hands off to AP2 and maintains its current utility. In the final part of the experiment, M4 hands off to AP2 at point H4 in the trace. At this instance, insufficient resources are available to support the base layers of M1, M2, M3 and M4 forcing the base station to block handoff.

4.6. Simulation

We have presented four common adaptation scripts in Section 4.5.. However, a wide range of policies can be programmed using our approach. In what follows, we use simulation to investigate the adaptation performance of flows measured at the base stations AP1 and AP2 in the simulation topology illustrated in Figure 4-7. We are interested in assessing the adaptation performance of flows that are forced to adapt to the observed conditions based on their instantiated adaptation scripts.

4.6.1 Simulation Environment

In Figure 4-7, the simulated wireless packet access network comprises two base stations (AP1 and AP2), two border gateways (GW1 and GW2) and two intermediate routers (SW1 and SW2). In the simulation, we only observe the performance of downlink flows which is sufficient to illustrate the effect of the adaptation script on mobile applications.

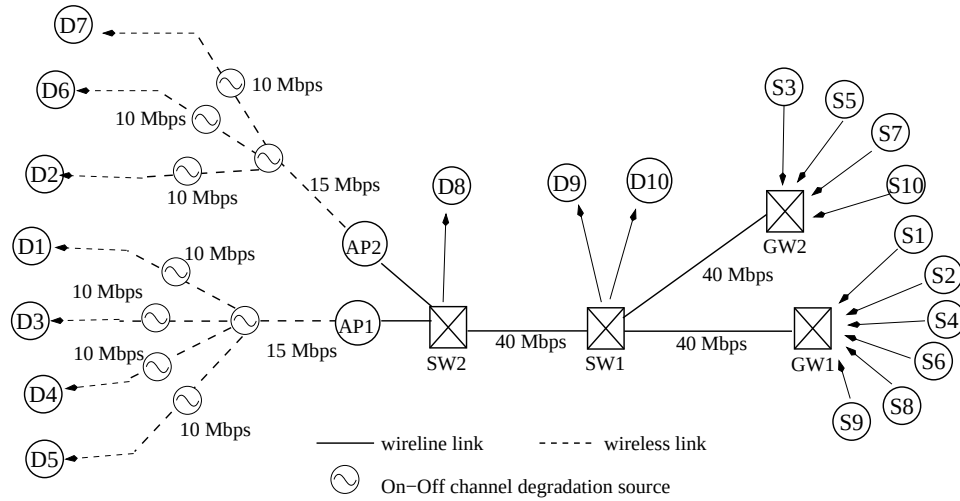


Figure 4-7: Simulated Mobile Access Network Topology

4.6.1.1 Flow Parameters

A total of ten flows with various adaptation scripts are simulated. For clarity, we assume one flow per traffic aggregate. Table 4.1 illustrates each flow's utility function parameters. All flows have zero minimum bandwidth requirements. Flows 9 and 10 simulate aggregated cross traffic in the access network with 25 Mb/s maximum bandwidth requirement ($B_{,K}$), the others simulate the flows across wireless links with $B_{,K}$ varying from 2 to 6 and 10 Mb/s. Two types of utility functions are used during simulations, as shown in Figure 4-8. Flow 4 and 6 have convex shape

of utility functions $u(x) = (K - 1) \log_2(1 + x/B_{.,K}) + 1$ while the other flows have linear shape utility functions $u(x) = (K - 1)x/B_{.,K} + 1$.

The route of each flow is designed carefully to give a fairness comparison under different utility functions and/or different $B_{.,K}$. To test the correctness of the flow aggregation operation, flows 1 and 4, 2 and 6, and 3 and 5 are aggregated into three flow aggregates, respectively. The remaining flows (7 - 10) are associated with different mobile devices.

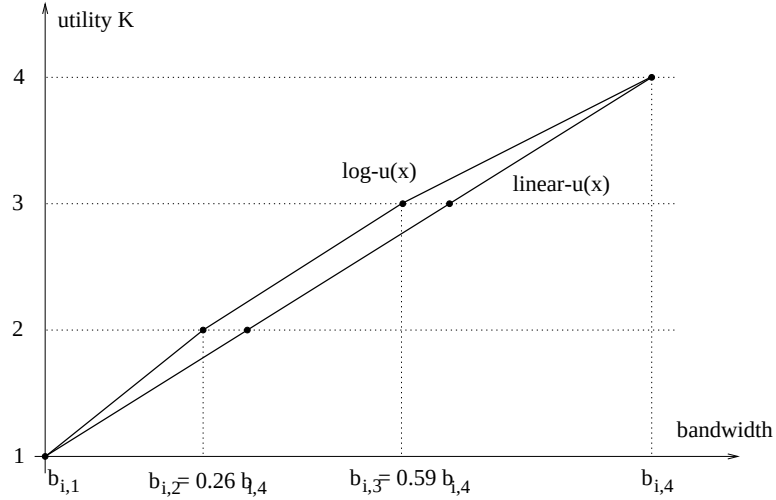


Figure 4-8: Utility Functions Used in Simulations

Table 4.1: Flow Utility Curve Parameters

Flow ID	1	2	3	4	5
U-C Shape	linear	linear	linear	log	linear
Max BW (Mb/s)	2	10	6	10	6
Flow ID	6	7	8	9	10
U-C Shape	log	linear	linear	linear	linear
Max BW (Mb/s)	2	6	6	25	25

4.6.1.2 Simulated Dynamics

We simplified the simulated wireless network by indirectly simulating the effect of user mobility and wireless channel variations on available bandwidth. There are

three levels of simulated dynamics related to available bandwidth variations in the simulated network, denoted as noise level 0 to 2.

Noise level 0 denotes the setting where the wireless links are ideal with no degradation. The available bandwidth variations are caused solely by the flows coming online, handing off and terminating. The scenario used in the simulation comprises a sequential setup of 10 flows in the first 10 sec, one flow is established every second, followed by flow 6 terminating at 85 sec, flow 4 terminating at 127 sec, flow 1 handing off at 171 sec and returning again at 204 sec into the scenario. Under noise level 0, the network topology shown in Figure 4-7 has three bottleneck links. The most congested bottleneck in this scenario is the link $SW2 \rightarrow AP1$, followed by the link $SW2 \rightarrow AP2$, and finally link $GW1 \rightarrow SW1$.

Noise level 1 denotes a setting where wireless link degradation is added to each base station, as well as the dynamics introduced by noise level 0. The links between a base station and router SW2 (illustrated in Figure 4-7 as links SW2–AP1 and SW2–AP2) have a capacity of 15 Mb/s, which models the overall air interface capacity between a base station and the mobile devices in its cell. Under noise level 1 conditions, the air interface capacity is changed by three random ON-OFF noise processes to simulate the effect of the wireless channel degradation common to all the mobile devices within the same cell. For each random noise process, during the ON interval, a uniform decrement of up to 3 Mb/s is deducted from the link capacity. During the OFF interval, no degradation is introduced in the link capacity. The ON and OFF intervals are exponentially distributed with a mean of 5 and 40 sec, respectively.

Noise level 2 denotes a setting where channel-dependent degradation is further added to each mobile device. This is in addition to all the dynamics introduced by noise level 0 and 1. To model channel-dependent degradation, we introduce links

between the mobile devices (D1 to D7) and their base stations (AP1 and AP2). These links have 10 Mb/s channel capacity under ideal channel conditions. This capacity is large enough to prevent the links from becoming bottlenecks because 10 Mb/s is no less than the maximum bandwidth requirement from each mobile device. When a random ON-OFF noise process is introduced at each link, however, the link capacity may be reduced to a level so that the per-mobile link becomes a bottleneck.

It should be noted that these common-channel and channel-dependent random ON-OFF noise models are not intended to closely capture the characteristics of a wireless channel under a fast time-scale. Rather, they coarsely simulate the effect of persistent fading, flow setup, release and handoff events on the time-scale comparable to the resource probing interval which is under study and is in the order of seconds.

4.6.2 Fairness Metric

The simulator implements two versions of the utility-based max-min allocation algorithms. A centralized scheme has global information and reacts instantaneously to any bandwidth change in the network. Therefore, the centralized scheme serves as the optimal (but not realistic) allocation reference. In our experiments, a distributed scheme implements our distributed utility-based max-min fair algorithm, as described in Section 4.4.3.

The fairness metric of the distributed scheme is calculated using the “fairness index” proposed in [53]. More specifically, denoting t_i the time instants when the bandwidth change occurs, the instantaneous fairness index of our distributed scheme

during time interval $[t_i, t_{i+1})$ is calculated as:

$$FI(t_i) = \frac{\left(\sum_{j=1}^N \gamma_j(t_i)\right)^2}{N \sum_{j=1}^N \gamma_j(t_i)^2} \quad \text{and} \quad \gamma_j(t_i) = \frac{x_j(t_i)}{\tilde{x}_j(t_i)}, \quad (4.5)$$

where N is the total number of flows, $x_j(t_i)$ is the allocation for flow j at time t_i under our distributed scheme, and $\tilde{x}_j(t_i)$ is the corresponding allocation under centralized scheme.

The average fairness metric between time t_0 and t_L is the time weighted average of the instantaneous fairness index, that is:

$$\overline{FI} = \frac{\sum_{i=1}^L FI(t_{i-1})(t_i - t_{i-1})}{t_L - t_0} \quad (4.6)$$

4.6.3 Results

In what follows, we show that our utility-based adaptation framework is capable of meeting the needs of a wide range of application adaptation strategies under diverse network conditions.

4.6.3.1 Greedy Adaptation

Figure 4-9 shows the simulation results under two noise levels. The fairness index under noise level 0 shows that the distributed algorithm converges to the theoretical utility-based max-min allocation. For example, in the figure, after the initial batch of flow setups up to 10 s; tear-downs at 85 and 127 s; and finally one flow hands off at 171 s and returns at 204 s, the fairness index reaches the maximum value of one within 20 s in all instances. This verifies the convergence property of our algorithm, as discussed in Section 4.4.2.

The fairness index under noise level 2 illustrates the system performance under

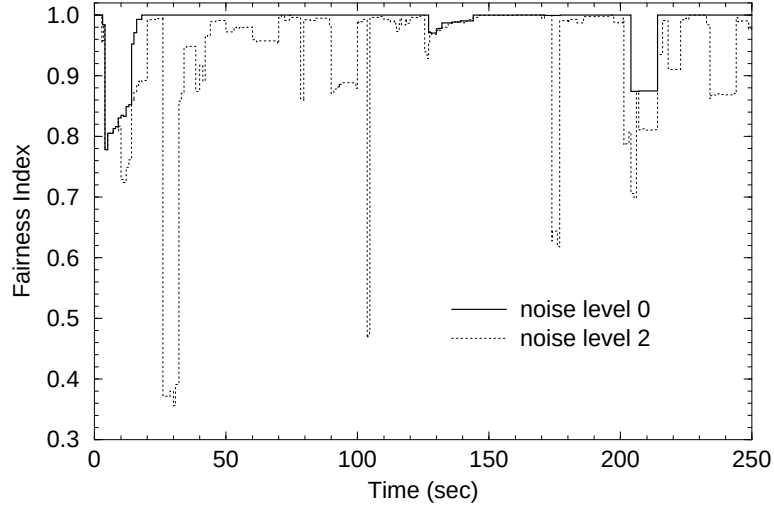


Figure 4-9: Greedy Adaptation Results: Fairness Index

severe channel-dependent degradation across wireless links. Here the deep drops in the fairness index imply that the allocation cannot react to instantaneous channel degradations. However, the fact that all drops are less than 10 second in duration indicates that the system rectifies the allocation inaccuracy within its probing interval of 10 second. This verifies that the periodic resource probing algorithm should be applied to persistent channel degradations and bandwidth variations at a time-scale slower than the probing interval.

The effect of the probing interval on allocation accuracy is further illustrated in Figure 4-10. This figure shows that under noise level 2 conditions, the average fairness index ($\overline{FI}$ averaged over 100000 seconds) decreases as the probing interval increases. A good choice of the probing interval needs to balance the trade-off between a desired fairness index value and increased signaling generated by a short probing interval.

To visualize the effect of utility-based fair allocation, we use Figure 4-11 to compare the allocated utility value of flows with different utility functions and maximum bandwidth requirements. For a fair comparison, we invoke noise level 1 and consider

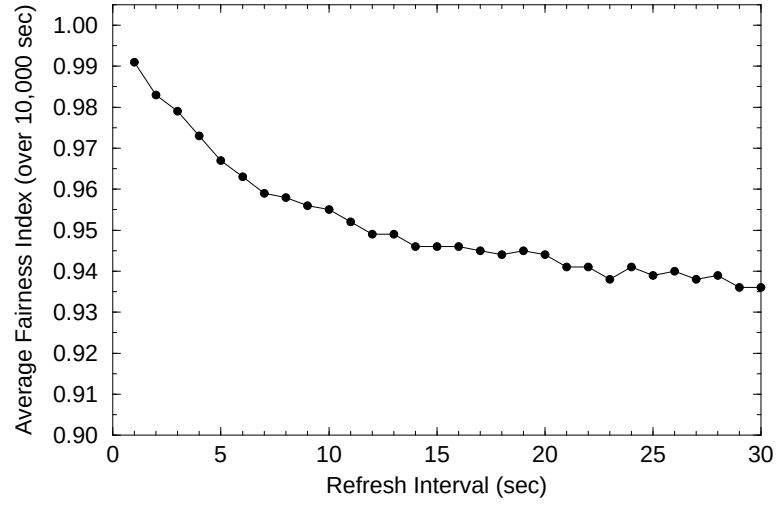


Figure 4-10: Greedy Adaptation Results: $\overline{FI}$ v.s. Probing Cycle

flows 1, 3, 4, 5 which are in the same cell. These flows experience the same common-channel degradation under noise level 1 because the air interface is simulated to be the same for each of the flows. We observe that after allocation convergence, all four flows observe the same utility value regardless of their utility function parameters.

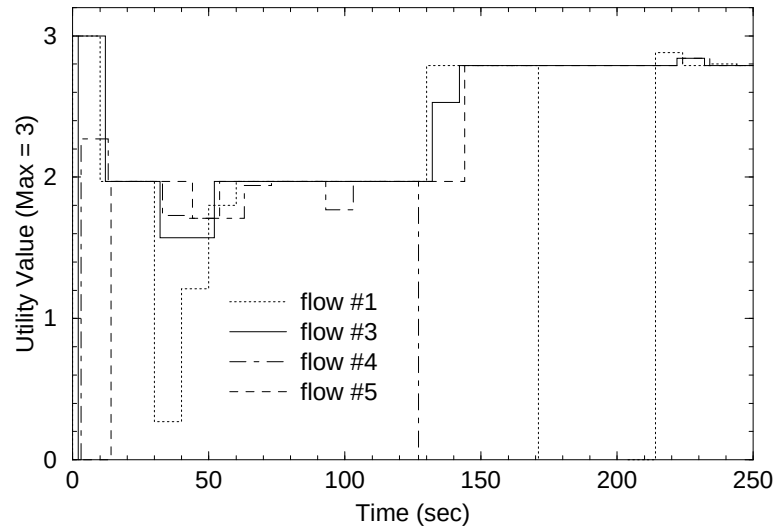


Figure 4-11: Greedy Adaptation Results: Utility Value

4.6.3.2 Discrete Adaptation

Figure 4-12 presents results from a simulation (see Section 4.6.) that has the same configuration as the greedy adaptation scenario. However, in this case the utility functions for flows 1 to 7 are made discrete, with three critical utility levels. Even though these seven flows have different shapes of discrete utility functions, they all operate at the same set of critical utility values $\{1, 2, \text{ and } 3\}$.

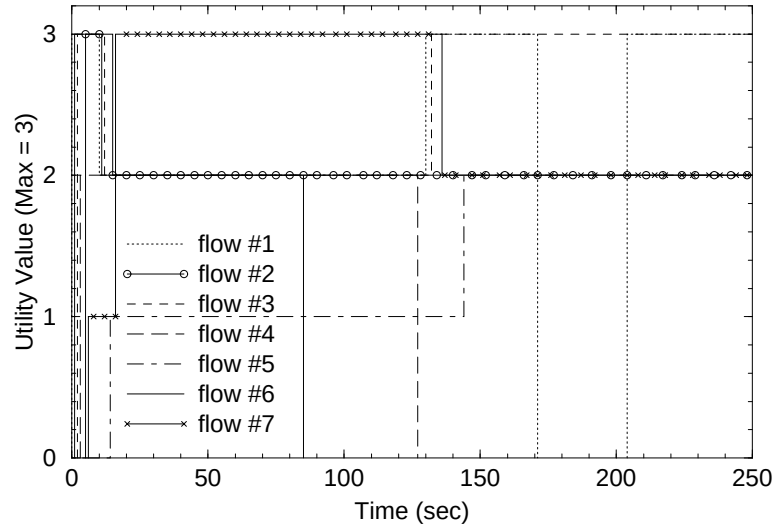


Figure 4-12: Discrete Adaptation Script Results: Utility Value

We observe that, strategy A can cause allocation disparity, as shown in Figure 4-13 (which is under the same scenario as in Figure 4-12 but for flow 3 and 5 only). Because flow 3 and 5 have the same utility function parameters and the same route, they should receive the same bandwidth allocation. However, as shown in Figure 4-13(a), under strategy A, flow 3 consistently receives allocation levels one level higher than flow 5. Strategy B corrects this disparity. The approach taken switches to a greedy adaptation script when a reduction of the assigned bandwidth is detected. The effect of this strategy is shown in Figure 4-13(b), where the two flows take turns receiving additional bandwidth. By successfully taking advantage of the distributed and asynchronous nature of adaptation, Strategy B resolves the

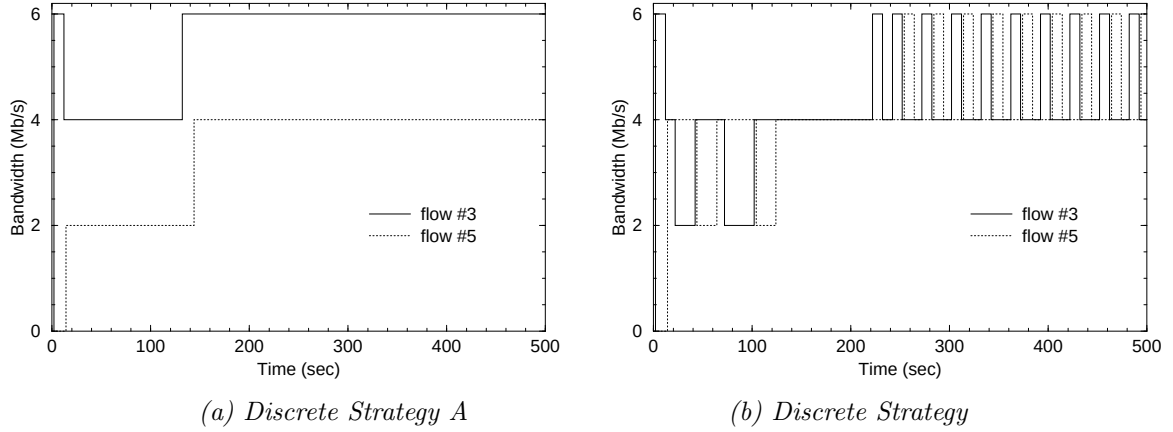


Figure 4-13: Comparison of Discrete Adaptation Strategies

fairness problem experienced by Strategy A, which otherwise, would be difficult to resolve in a deterministic manner.

4.6.3.3 Smooth Adaptation

Figure 4-14 presents the simulation results under the same network configuration as the previous simulations except that, flow 1, 3 and 7 have smooth adaptation strategies, while the other flows are greedy. For comparison purposes, only flows 1, 2, 3 and 7 are shown in Figure 4-14 (with noise level 2 to introduce large bandwidth variations).

The smooth adaptation script consists of parameters: δ , τ and κ as defined above. In the simulation, all the smooth adaptation flows have the same $\kappa = 80\%$. Flow 1 is constrained by τ and δ , where $\tau = 20$ sec and $\delta = 0.66$ Mb/s, one third of its maximum bandwidth requirement. Flow 2 is not constrained by either τ or δ , so its $\tau = 10$ sec, the default probing cycle, and $\delta = 10$ Mb/s, its maximum bandwidth requirement. Flow 3 is constrained only by τ , where $\tau = 40$ sec and $\delta = 6$ Mb/s, its maximum bandwidth requirement. Finally, flow 7 is constrained only by δ , where $\delta = 0.2$ Mb/s and $\tau = 10$ sec, the default probing cycle. In

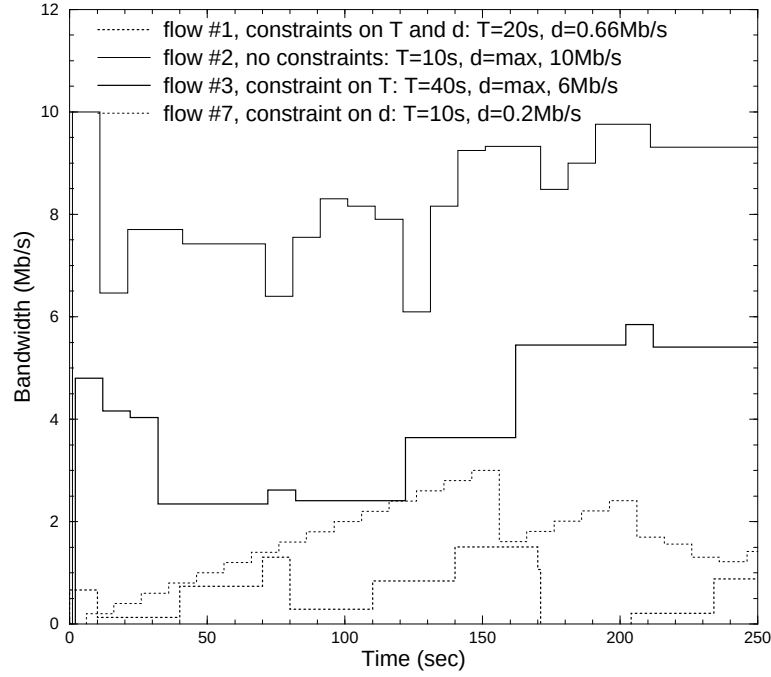


Figure 4-14: Smooth Adaptation Script Results

comparison with the greedy adaptation flow 2, smooth adaptation flows experience less bandwidth oscillations (i.e., “QOS flapping”), as shown in Figure 4-14. When τ increases with multiples of the probing interval, the allocated bandwidth becomes more stable, as shown by flows 1 and 3. One can also employ small but frequent increments (i.e., small τ and δ) and achieve the behavior like flow 7 to follow the trend of bandwidth variation.

We note from this experiment that the mean allocated bandwidth under smooth adaptation is smaller than greedy adaptation. This is because the gain on smoothness is traded off with the allocation increment one can accept each time.

4.7. Summary

The contribution of this chapter is as follows. We introduced a utility-based adaptation framework for mobile networking that enables users and service providers to

program application-specific adaptive services in wireless packet access networks. The architecture provides split-level support for adaptive QOS control at the network and application levels based on bandwidth utility functions and adaptation scripts. We have discussed the detailed design of network-level and application-level adaptation control mechanisms that maintain a balance between architectural scalability and flexibility. Our network-level algorithms and protocols employ utility functions to support common adaptation needs in a manner that is scalable for traffic aggregates and across multiple network hops. Our application-level adaptation handlers operate at mobile devices realizing application-specific adaptation scripts. Through simulations we have shown the convergence property of the network-level adaptation algorithm. In addition, we have demonstrated the operation of different styles of adaptation scripts that can be programmed by the user or application service provider.

In this chapter, we focused on applications needs and scalable resource allocation algorithm design for edge-based wireless access networks. In the next chapter, we study new mechanisms that support scalable dynamic provisioning of core IP networks in support of quantitative differentiated services.

A. Pseudo-code for the Utility-weighted Max-min Fair Allocation Algorithm

// a reserve message arrives at a network node

reserve(*id*, req^{ideal} , req^{actual} , **u**) {

if (*id* is new flow) {

// first flow, init states

$x_{id}^{ideal} = 0$; $x_{id}^{actual} = 0$;

$\mathbf{u}_{id} = \mathbf{u}$;

$state_{id} = \text{SATISFIED}$;

```

    update u into uALL;
    add u into uU;
} else {
    update uALL
    if (stateid == SATISFIED) {
        // prepare for allocation
        add u into uU
         $\mathcal{B}_{U+} = x_{id}^{ideal};$ 
    }
     $\mathcal{B}_{free+} = x_{id}^{actual};$ 
}
alloc_bw(id, reqideal, reqactual);
}

// utility-fair algorithm: Eqn (4.2)
alloc_bw(id, reqideal, reqactual) {
    xalloc = util_fair_alloc(id);
    // without feasibility constraint
     $x_{id}^{ideal} = \min\{x_{alloc}, req^{ideal}\};$ 
    // with feasibility constraint
     $x_{id}^{actual} = \min\{x_{alloc}, req^{actual}, \mathcal{B}_{free}\};$ 
    // adjust flow state
    if ( $x_{id}^{ideal} < req^{ideal}$ ) stateid = BTLNECKED;
    else stateid = SATISFIED;
    if (stateid == SATISFIED) {
        remove u from uU
         $\mathcal{B}_{U-} = x_{id}^{ideal};$ 
    }
     $\mathcal{B}_{free-} = x_{id}^{actual};$ 
    reqideal =  $x_{id}^{ideal};$ 
    reqactual =  $x_{id}^{actual};$ 
}

```

```

}

// a commit message arrives at a network node
commit(id, ackideal, ackactual) {
     $\mathcal{B}_{free+} = (x_{id}^{actual} - ack^{actual});$ 
     $x_{id}^{actual} = ack^{actual};$ 
    if (ackideal  $\geq$   $x_{id}^{ideal}$ ) return;
    // ackideal <  $x_{id}^{ideal}$ , adjust flow states
    if (stateid == SATISFIED)  $\mathcal{B}_U+ = x_{id}^{ideal};$ 
     $x_{id}^{ideal} = ack^{ideal};$ 
    // update index for  $\mathbf{u}_{max}$ 
    if ( $x_{id}^{ideal} > x_{max}$ ) {
         $x_{max} = x_{id}^{ideal};$ 
        max_id = id;
    } else if ( $x_{id}^{ideal} < x_{max}$  and id == max_id) {
         $x_{max} = x_{id}^{ideal};$  // reduce  $x_{max}$ 
    }
    if (id == max_id) {
        if (stateid == SATISFIED) {
            stateid = BTLNECKED;
            add  $\mathbf{u}_{id}$  into  $\mathbf{u}_U$ ;
        }
    } else {
        if (stateid == BTLNECKED) {
            stateid = SATISFIED;
            remove  $\mathbf{u}_{id}$  from  $\mathbf{u}_U$ ;
        }
         $\mathcal{B}_{U-} = x_{id}^{ideal};$ 
    }
}

```

```

// flow release or timeout
release(id) {
    remove  $\mathbf{u}_{id}$  from  $\mathbf{u}_{ALL}$ ;
    if ( $state_{id} == \text{BTLNECKED}$ )
        remove  $\mathbf{u}_{id}$  from  $\mathbf{u}_U$ ;
    else
         $\mathcal{B}_{U+} = x_{id}^{ideal}$ ;
    // return BW to free pool
     $\mathcal{B}_{free+} = x_{id}^{actual}$ ;
    delete flow states;
}

```

Chapter 5

Quantitative Service Differentiation for Traffic Aggregates in Core Networks

5.1. Introduction

Provisioning differentiated services in core IP networks presents a number of significant technical challenges above and beyond those associated with engineering efficient allocation algorithms for edge-based access networks, as discussed in the first part of this thesis. The contribution of this chapter is to study the needs of core network infrastructure and develop scalable dynamic bandwidth provisioning algorithms that can support quantitative differentiated services across core networks. Engineering efficient bandwidth provisioning for core networks (e.g., IETF Differentiated Services “DiffServ” [35]) is, however, considerable more challenging than in circuit-based networks such as the Asynchronous Transfer Mode (ATM) networks. First, there is a lack of detailed control information (e.g., per-flow states) and supporting mechanisms (e.g., per-flow queueing) in the network. Second, there is a need to provide increased levels of service differentiation over a single global IP infrastructure. In traditional telecommunication networks, where traffic characteristics are well understood and well controlled, long-term capacity planning can be

effectively applied. We argue, however, that in a DiffServ Internet more dynamic forms of control will be required to compensate for coarser-grained state information and the lack of network controllability, if service differentiation is to be realistically delivered.

There exists a trade-off intrinsic to the DiffServ service model (i.e., qualitative v.s. quantitative control). DiffServ aims to simplify the resource management problem thereby gaining architectural scalability through provisioning the network on a per-aggregate basis, which results in some level of service differentiation between service classes that is *qualitative* in nature. Although under normal conditions, the combination of DiffServ router mechanisms and edge regulations of service level agreements (SLA) could plausibly be sufficient for service differentiation in an over-provisioned Internet backbone, network practitioners have to use *quantitative* provisioning rules to automatically re-dimension a network that experiences persistent congestion or device failures while attempting to maintain service differentiation [98, 84]. Therefore, a key challenge for the emerging DiffServ Internet is to develop solutions that can deliver suitable network control granularity with scalable and efficient network state management.

The contribution of this chapter is as follows. We propose adaptive bandwidth management algorithms to dynamically provision quantitative differentiated services within a service provider's core IP network (i.e., the intra-domain aspect of the provisioning problem). Our SLA provides quantitative per-class delay guarantees with differentiated loss bounds across core IP networks. We introduce a distributed *node provisioning algorithm* that works with class-based weighted fair (WFQ) schedulers and queue management schemes. This algorithm prevents transient service level violations by adjusting the service weights for different classes after detecting the onset of SLA violations. The algorithm uses a simple but effective approach (i.e., the

virtual queue method proposed in [44, 59]) to predict persistent SLA violations from measurement data and reports to our network *core provisioning algorithm*, which in turn coordinates rate regulation at the ingress network edge (based on results in Chapter 3 and [70]).

In addition to delivering a quantitative SLA, another challenge facing DiffServ provisioning is the rate control of any traffic aggregate comprising of flows exiting at different network egress points. This problem occurs when ingress rate control can only be exerted on a per traffic aggregate basis, (i.e., at the root of a traffic aggregate’s point-to-multipoint distribution tree). Under such conditions, any rate reduction of an aggregate would penalizes traffic flowing along branches of the tree that are not congested. We call such a penalty *branch-penalty*. One could argue for breaking down a customer’s traffic aggregate into per ingress-egress pairs and provisioning in a similar manner to circuit-based Multi-Protocol Label Switching (MPLS) [90] tunnels. Such an approach, however, would only imply that per-tunnel routing and packet accounting is available, but not per-tunnel queueing in the core and edge routers due to the complexity of implementing buffer management and scheduling functions necessary for per-tunnel queueing. In addition, provisioning of per ingress-egress pair tunnels in a dynamic environment is not scalable because adding or removing an egress point to the network would then require reconfiguration of all ingress points in order to add/remove the point-to-point tunnels from all ingress points to the target egress point¹. Our approach comprises a suite of policies in our core provisioning algorithm to address the provisioning issues that arises when supporting point-to-multipoint traffic aggregates. Our solution includes

¹We should note that our approach does not exclude support for MPLS tunnels. In fact, our approach would also benefit from the availability of MPLS tunnels because MPLS per-tunnel traffic accounting statistics will improve the measurement accuracy of our traffic matrix, as discussed in Section 5.5.1. Our approach also solves the scalability problem of per-MPLS-tunnel traffic shaping by supporting traffic regulation for MPLS aggregates.

policies that minimize branch-penalty, deliver fairness with equal reduction across traffic aggregates, or extend the max-min fairness for point-to-multipoint traffic aggregates.

The structure of this chapter is as follows. In Section 5.2., we discuss related work and contrast our approach to the literature. In Section 5.3., we introduce a dynamic bandwidth provisioning architecture and service model for core IP networks. Following this, in Section 5.4., we present our dynamic node provisioning mechanism, which monitors buffer occupancy, self-adjusts scheduler service weights and packet dropping thresholds at core routers. In Section 5.5., we describe our core provisioning algorithm, which dimensions bandwidth at ingress traffic conditioners located at edge routers taking into account the fairness issue of point-to-multipoint traffic aggregates and SLAs. In Section 5.6., we discuss our simulation results demonstrating that the proposed algorithms are capable of supporting the dynamic provisioning of SLAs with guaranteed delay, differential loss and bandwidth prioritization across per-aggregate service classes. We also verify the effect of rate allocation policies on traffic aggregates. Finally, in Section 5.7., we present some concluding remarks to the chapter.

5.2. Related Work

Dynamic bandwidth provisioning algorithms are complementary to scheduling and admission control algorithms. These bandwidth management algorithms operate on a medium time-scale, as illustrated in Figure 1-1. In contrast, packet scheduling and flow control operate on fast time-scales (i.e., sub-second time-scales); admission control and dynamic bandwidth provisioning operate on medium time-scales in the range of seconds to minutes; and traffic engineering, including rerouting and capacity planning, operate on slower time-scales on the order of hours to months.

Significant progress has been made in the area of scheduling and flow control, (e.g., dynamic packet state and its derivatives [95, 96]). In the area of traffic engineering, solutions for circuit-based networks have been widely investigated in literature (e.g., [76, 57]). There has been recent progress on developing measurement techniques for IP networks [37, 36, 29]. In contrast, for the medium time-scale mechanisms, most research effort has been focused on admission control issues including edge [21] and end host based admission control [17]. However, these algorithms do not provide fast mechanisms that are capable of reacting to sudden traffic pattern changes. Our bandwidth provisioning algorithms are capable of quickly restoring service differentiation under severely congested and device failure conditions.

Delivering quantitative service differentiation for the DiffServ service model in a scalable means has attracted a lot of attentions recently. A number of researchers have proposed effective scheduling algorithms. Stoica et. al. propose the Dynamic Packet State [95] to maintain per-flow rate information in packet headers leading to fine-grained per-flow packet-dropping that is locally fair (i.e., at a local switch). However, this scheme is not max-min fair due to the fact that any packet drop inside the core network wastes upstream link bandwidth that otherwise could be utilized. In [96], Stoica and Zhang extend the solution of [95] to support per-flow delay guarantees in a DiffServ network. Our work operates on top of per-class schedulers with emphasis on bandwidth allocation and the maintenance of service differentiation and network-wide fairness properties. The proportional delay differentiation scheme [27] defines a new qualitative “relative differentiation service” as oppose to quantifying “absolute differentiated services”. The node provisioning algorithm presented in this chapter also adopts a self-adaptive mechanism to adjust service weights at core routers. However, our service model differs from [27] by providing delay guarantees across a core network while maintaining relative loss differentiation. The work dis-

cussed in [71] has similar objectives to our node provisioning algorithm. However, it is motivated by a more comprehensive set of objectives in comparison to our work because it attempts to support optimization objectives that include multiple constraints for both relative and absolute loss and delay differentiation.

The idea of using virtual queues in scheduler design is a well accepted technique. For example, in [48] a duplicate queue is constructed to support two “Alternative Best-Effort” services (viz. low delay vs. high throughput). In our work, we use virtual queues to predict the onset of SLA violations. The idea was originally proposed in [44, 59] as a good traffic prediction technique for traffic with complex characteristics, such as, self similarity, because its stochastic properties share the same dominant time-scale with the original queue. Our algorithm extends this work by dynamically adjusting the virtual queue scaling parameter with respect to queueing conditions.

Our approach to dynamic bandwidth provisioning is complementary to the work on edge/end-host based admission control [21, 17], with admission control at the edge of core networks and provisioning algorithms operating inside core networks. An alternative approach that solely uses admission control for a DiffServ network can support stricter QOS guarantee but also lead to large complexity in the QOS control plane. For example, in [111], a complex bandwidth broker algorithm is presented to maintain the control states of core routers and perform admission control for the whole network. In contrast, our provisioning algorithm uses a distributed node algorithm to detect and signal the need for bandwidth re-allocation. The centralized core algorithm only maintains network load matrix and coordinates the allocation algorithm for fairness purposes.

Currently network service providers use rerouting based traffic engineering approaches to cope with network traffic dynamics on slow time-scales. In the inter-

domain case where one provider has no direct control of her peering networks, absence of direct control leads to the use of intra-domain routing policy as the only viable technique, with potential solutions ranging from optimal planning of routes for circuits/virtual path [16], to traffic measurement-based adjustment on OSPF weights and BGP route policies [37]. In the intra-domain case where direct control is possible, bandwidth provisioning can offer faster response to service degradations.

Our bandwidth provisioning method bears similarity to the work on edge-to-edge flow control [4] but differs in that we provide a solution for point-to-multipoint traffic aggregates unique to a DiffServ network rather than the point-to-point approach discussed in [4]. In addition, our emphasis is on the delivery of multiple levels of service differentiation.

5.3. A Dynamic Bandwidth Provisioning Model for Core Networks

5.3.1 Architecture

We assume a DiffServ framework where edge traffic conditioners perform traffic policing/shaping. Nodes within the core network use a class-based weighted fair (WFQ) scheduler and various queue management schemes for dropping packets that overflow queue thresholds.

The dynamic bandwidth provisioning architecture illustrated in Figure 5-1 comprises dynamic core and node provisioning modules for bandwidth brokers and core routers, respectively, as well as the edge provisioning modules that are located at access and peering routers. The edge provisioning module [70] performs ingress link sharing at access routers, and egress capacity dimensioning at peering routers.

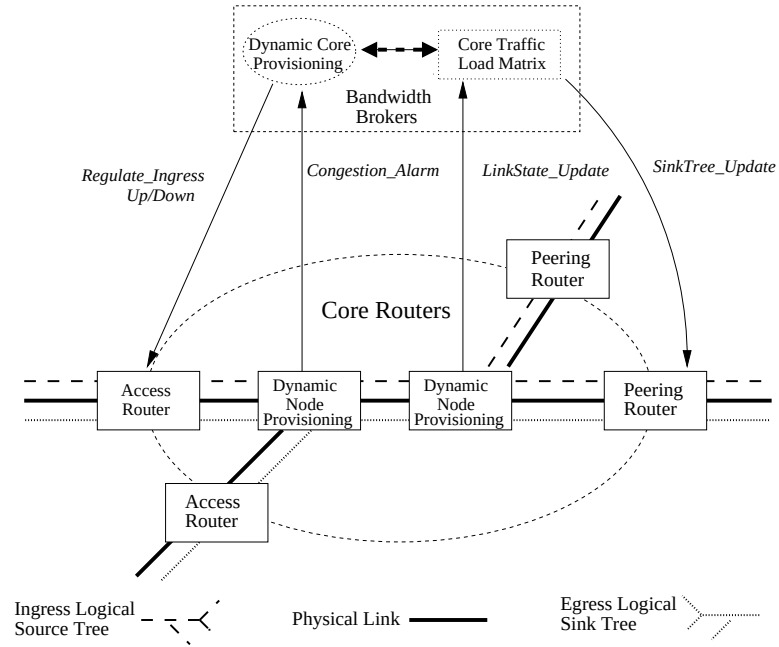


Figure 5-1: Dynamic Bandwidth Provisioning Model for Core Networks

5.3.2 Control Messaging

Dynamic core provisioning sets appropriate ingress traffic conditioners located at access routers by utilizing a *core traffic load matrix* to apply rate-reduction (via a *Regulate_Ingress Down* signal) at ingress conditioners, as shown in Figure 5-1. Ingress conditioners are periodically invoked (via the *Regulate_Ingress Up* signal) over longer restoration time-scales to increase bandwidth allocation restoring the max-min bandwidth allocation when resources become available. The core traffic load matrix maintains network state information. The matrix is periodically updated (via *LinkState_Update* signal) with the measured per-class link load. In addition, when there is a significant change in the rate allocation at egress access routers, a core bandwidth broker uses a *SinkTree_Update* signal to notify egress dimensioning modules at peering routers when renegotiating bandwidth with peering networks, as shown in Figure 5-1. We use the term “sink-tree” to refer to the topo-

logical relationship between a single egress link (representing the root of a sink-tree) and two or more ingress links (representing the leaves of a sink-tree) that contribute traffic to the egress point.

Dynamic core provisioning is triggered by *dynamic node provisioning* (via a *CongestionAlarm* signal as illustrated in Figure 5-1) when a node persistently experiences congestion for a particular service class. This is typically the result of some local threshold being violated. Dynamic node provisioning adjusts service weights of per-class weighted schedulers and queue dropping thresholds at local core routers with the goal of maintaining delay bounds and differential loss and bandwidth priority assurances.

5.3.3 Service Model

The proportional delay differentiation service proposed in [27] defines the relative service differentiation of a single node and not a path through a core network. In contrast, our work produces service assurances that are quantitative in terms of delay bound, loss differentiation, and support bandwidth allocation priorities across service classes within a DiffServ core network.

Our SLA comprises:

- a *delay guarantee*: where any packet delivered through the core network (not including the shaping delay of edge traffic conditioners) has a delay bound of D_i for network service class i ;
- a *differentiated loss assurance*: where network service classes are loss differentiated, that is, for traffic routed through the same path in a core network, the long-term average loss rate experienced by class i is no larger than $P_{loss,i}^*$. The thresholds $\{P_{loss,i}^*\}$ are differentiated, i.e., $P_{loss,(i-1)}^* < P_{loss,i}^*$;

- a *delay bound precedence over loss bound*: when both the delay and loss bounds can not be maintained for class i , the loss bound will be revoked first before the delay bound;
- a *bandwidth allocation priority*: where the traffic of class j never affects the bandwidth/buffer allocation of class $i, i < j$, that is, the delay and loss bounds of class i will be revoked only after there is no bandwidth available (excluding the minimum bandwidth for each class) in classes $j, j > i$;
- a *bandwidth utility function*: which provides an application programming interface (API) for edge service differentiation. The utility function serves as a user-approved per-class QOS degradation trajectory used by network provisioning algorithms under network congestion or failure conditions.

We design the service model such that maintaining a quantitative delay bound takes precedence over maintaining the packet loss bound. This precedence helps to simplify the complexity of jointly maintaining both loss and delay bounds at the same time. In addition, such a service is suitable for TCP applications that need packet loss as an indicator for flow control while guaranteed delay performance can support real-time applications. The precedence to delay bound does not mean that the loss bound will be ignored. For a service class with higher bandwidth allocation priority, its loss bound will be maintained at the cost of violating lower priority classes' loss and delay bounds.

In addition, the Congestion_Alarm signal from the node provisioning algorithm will give an early warning to the core provisioning algorithm, which can work with the admission control algorithm and edge-based traffic regulation algorithm to remove congestion inside the core network. One benefit of our dynamic provisioning algorithms is its ability to maintain service differentiation under unavoidable pre-

diction errors made by the admission control algorithm.

The granularity of per-node delay bounds D_i is limited by the nature of slow time-scale aggregate provisioning. The choice of D_i has to take into consideration the sum of a single packet transmission time at the link rate and a single packet service time through various fair queue schedulers [61]. This is in addition to the queueing delays due to traffic aggregates inside the core network.

The choice of loss threshold $P_{loss,i}^*$ in an SLA also needs to consider the application behavior. For example, a service class intended for data applications should not specify a loss threshold that can impact steady-state TCP behavior. Studies [75] indicate that the packet drop threshold $P_{loss,i}^*$ should not exceed 0.01 for data applications to avoid the penalty of retransmission timeouts.

We define a service model for the core network that includes a number of algorithms. A node provisioning algorithm enforces delay guarantees by dropping packets and adjusting service weights accordingly. A core provisioning algorithm maintains the dropping-rate differentiation by dimensioning the network ingress bandwidth. Edge provisioning modules perform rate regulation based on utility functions. Even though these algorithms are not the only solution to supporting the proposed SLA, their design is tailored toward delivering quantitative differentiation in the SLA with minimum complexity.

Note that, utility-based edge dimensioning has been investigated in Chapter 3. In the remaining part of this chapter, we focus on core network provisioning algorithms which are complementary components to the edge algorithms of our dynamic bandwidth provisioning architecture shown in Figure 5-1.

5.4. Dynamic Node Provisioning

The design of the node provisioning algorithm follows the typical logic of measurement-based closed-loop control. The algorithm is responsible for two tasks: (i) to predict SLA violations from traffic measurements; and (ii) to respond to potential violations with local reconfigurations. If violations are severe and persistent, then reports are sent to the core provisioning modules to regulate ingress conditioners, as shown in Figure 5-1.

The detection of SLA violation is triggered by the virtual queue method proposed in [44, 59]. A virtual queue has exactly the same incoming traffic as its corresponding real queue but with both the service rate and buffer size scaled down by a factor of $\kappa \in (0, 1)$. The virtual queue technique offers a generic and robust traffic control mechanism without assuming any traffic model (e.g., the Poisson arrivals, etc.). It performs well under complex traffic arrival processes including self similarity [59]. In our node provisioning algorithm, we extend the technique to queues with multiple classes served by a weighted fair queueing scheduler by dynamically adjusting the scaling parameter κ_i for each class.

The algorithm is invoked either by the event of detecting an onset of SLA violation, or periodically over an *update_interval* interval. The value of the *update_interval* does not affect the detection of SLA violations because the virtual queue mechanism can trigger algorithm execution immediately without the constraint of the *update_interval*. However, the *update_interval* will affect the speed to detect the system under-load, and the measurement of traffic statistics. In Section 5.6.2.2, we investigate the appropriate choice of the *update_interval* value.

The SLA service model introduced in Section 5.3.3 is intended to be simple for ease of implementation. However, it still requires non-trivial joint control of both service weight allocation and buffer dimensioning to maintain the delay and loss

bounds D_i and $P_{loss,i}^*$, respectively.

5.4.1 Loss Measurement

When $P_{loss,i}^*$ is small, solely counting rare packet loss events can introduce a large bias. Instead, the algorithm works with the inverse of the loss rate which essentially tracks the number of consecutively accepted packets. For each class, a target loss control variable $lossfree_cnt_i$ is measured upon each update epoch t_n . Denote $cnt_{accepted}$ the number of accepted packets during the interval $(t_{n-1}, t_n]$, and $cnt_{dropped}$ the number of dropped packets in the same interval, then we have

$$lossfree_cnt_i(t_n) = (cnt_{dropped} + 1)/P_{loss,i}^* - cnt_{accepted}. \quad (5.1)$$

In other words, $lossfree_cnt_i$ represents the number of packets that have to be accepted consecutively under the $P_{loss,i}^*$ bound before the next packet drop. $lossfree_cnt_i \leq 0$ signifies that the $P_{loss,i}$ bound is not violated; $lossfree_cnt_i > 1/P_{loss,i}^*$ indicates the opposite; while $lossfree_cnt_i \in (0 \ 1/P_{loss,i}^*]$ indicates that there have not been sufficient packet arrivals yet.

The measurement of $cnt_{accepted}$ and $cnt_{dropped}$ uses a measurement window τ_l , which is one order of magnitude larger than the product of $1/P_{loss,i}^*$ and the mean packet transmission time in order to have a statistically accurate calculation of the packet loss rate. In the simulation section, we use $\tau_l \geq 10s$. However, a large τ_l means that a currently partial measurement sample has to be considered for the instantaneous packet loss. To improve statistical reliability, we also use the complete sample in the preceding window for calculation, that is:

$$\begin{aligned} cnt_{accepted} &= accept_count(prev) + accept_count_partial(now) \\ cnt_{dropped} &= drop_count(prev) + drop_count_partial(now). \end{aligned} \quad (5.2)$$

5.4.2 Delay Constraint

Our algorithm controls delay by buffer dimensioning and service weight adjustment. Exact calculation of the maximum delay of all enqueued packets is expensive since it requires tracking the queueing delay incurred by every enqueued packet. Instead, we calculate the current maximum queueing delay with its upper bound:

$$d_i \leq \bar{d}_i \triangleq d_i(HOL) + N_q/\mu_i, \quad (5.3)$$

where $d_i(HOL)$ is the queue delay of the head-of-line (HOL) packet, N_q is the queue size, and μ_i is the lower bound of the packet service rate calculated from the proportion of service weights in a WFQ scheduler². The benefit of Equation (5.3) is that we only need to calculate the delay of the HOL packet. Consequently, the down-side of this is that $\bar{d}_i$ becomes an approximation of the current maximum queueing delay. In fact, it represents an upper bound of the current maximum queueing delay because the first portion of Equation (5.3) represents the maximum queueing delay incurred by any of the enqueued packets handled so far. The bound can be reached when all the enqueued packets arrived at the same time. Note that the same technique is used in [71] to measure the maximum queueing delay.

Now with $\bar{d}_i \leq D_i$, and Inequality 5.3, we obtain a lower bound for service rate μ_i :

$$\mu_i(new) \geq N_q/(D_i - d_i(HOL)). \quad (5.4)$$

This means that $\mu_i(new)$ needs to be above the lower bound in order to meet the delay bound of the enqueued packets. Subsequently, the dimensioning of buffer size

² μ_i is a lower bound because the actual service rate will be higher when some of the other class queues are idle.

Q_i for the i th class queue can be derived as:

$$Q_i(\text{new}) = \tilde{D}_i, \text{ where } \tilde{D}_i = \begin{cases} D_i - d_i(\text{HOL}) & \text{if } D_i > d_i(\text{HOL}), \text{ delay bound} \\ & \text{not violated;} \\ D_i & \text{otherwise, delay bound violated.} \end{cases} \quad (5.5)$$

5.4.3 Virtual Queue Scaling

The virtual queue technique proposed in [44, 59] needs to be extended for a WFQ scheduler with multiple queues. Denote w_i the service weight of class i , then the minimum service rate is:

$$\mu_i = \frac{w_i}{\sum_i w_i} \text{linerate}. \quad (5.6)$$

Denote κ_i the scaling parameter for the i th queue, then the buffer size of each class queue is scaled down by κ_i . For the total service rate of the WFQ scheduler, we have:

$$\text{linerate}_{VQ} = \sum_i \kappa_i \mu_i = \frac{\sum_i \kappa_i w_i}{\sum_i w_i} \text{linerate}. \quad (5.7)$$

Namely, the scaling parameter for the total service rate is $\sum_i \kappa_i w_i / \sum_i w_i$, which is the weighted average of the individual scaling parameters.

The setting of κ_i takes into consideration the speed mismatch between instantaneous arrival rate and service rate, and the response time of the queueing system to the adjustment of service weights. The purpose is to choose κ_i such that the early warning generated from the virtual queue will give enough time for the WFQ scheduler to react.

Since the node provisioning algorithm targets operating at the buffer half-full point to counter both queue under-load and overload, we can assume that the available buffer space at the beginning of an *update_interval* is $Q_i/2$. In addition, we

focus on the case where the traffic load $\rho_i \triangleq \lambda_i/\mu_i > 1$, which represents the extend of rate mismatch between queue arrival and departure. Therefore, the time that it takes to fill the real queue buffer is:

$$t_{RQ} = \frac{Q_i/2}{(\rho_i - 1)\mu_i}. \quad (5.8)$$

For the virtual queue, with κ_i scaling down Q_i and μ_i , we have the time that takes to fill the virtual queue buffer as:

$$t_{VQ} = \frac{\kappa_i Q_i/2}{(\rho_i - \kappa_i)\mu_i}. \quad (5.9)$$

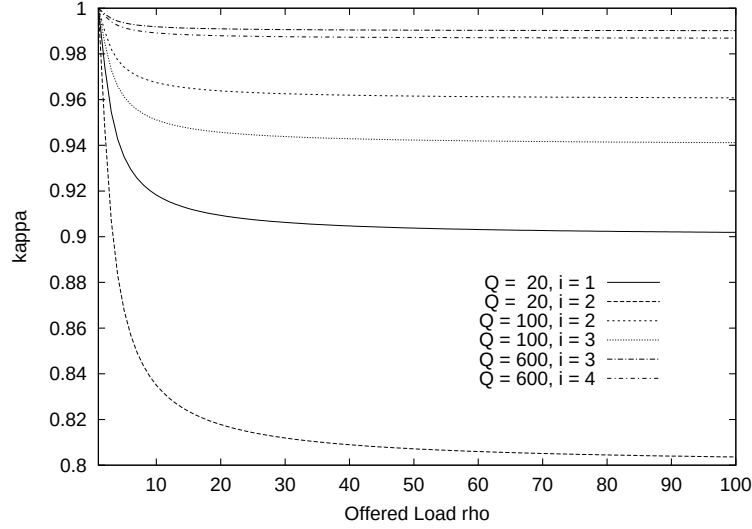
For a WFQ style (e.g., Weight Round Robin) scheduler, we estimate the system response time to the change in service weights as i/λ_i . That is, the response time is proportional to the number of queueing classes that have higher or equal allocation priority than i , and inversely proportional to the line rate. Here we use λ_i to approximate the line-rate. Therefore, we have the following inequality in order to achieve the early warning of buffer over-flow:

$$t_{RQ} - t_{VQ} = \frac{Q_i}{2\mu_i} \frac{\rho_i(1 - \kappa_i)}{(\rho_i - 1)(\rho_i - \kappa_i)} \geq \frac{i}{\rho_i \mu_i}. \quad (5.10)$$

Solving this inequality, we have the upper bound for setting κ as:

$$\kappa_i = \frac{\frac{Q_i}{2i} \rho_i^2 - \rho_i(\rho_i - 1)}{\frac{Q_i}{2i} \rho_i^2 - (\rho_i - 1)}. \quad (5.11)$$

Figure 5-2 shows some typical values of κ as a function of ρ_i , Q_i and i . The value of κ_i is sensitive to the buffer size Q_i and the number of higher or equal priority queueing classes i . However, the value of κ does not vary much for large values of ρ_i , which represents extremely bursty traffic conditions. This indicates that the

Figure 5-2: Example of κ Values

dynamic adjustment of the virtual queue scaling parameter is applicable to a wide range of traffic conditions. Indeed, taking the limit of ρ_i on (5.11), we have:

$$\lim_{\rho_i \rightarrow \infty} \kappa_i = 1 - \frac{2i}{Q_i}. \quad (5.12)$$

This limit is also the lower bound of κ_i . It is desirable to keep the scaling parameter of a virtual queue not too small because otherwise, the virtual queue will generate a lot of false positive alarms. That is, $\frac{2i}{Q_i}$ should remain close to zero. Because $\frac{2i}{Q_i}$ increases as Q_i decreases, a small $Q_i \approx \mu_i D_i$ also means small delay requirement usually for higher allocation priority classes, therefore i is necessarily small as well. As a result, κ_i will stay away from values close to zero.

5.4.4 Control Action

The control action is to adjust the service rate (weight) as well as buffer size based on the short-term measurement of the traffic arrival rate $\bar{\lambda}_i$ and the queue length

$\bar{N}_{q,i}$. The measurement method is the same as the dual-window averaging method used for loss measurement in Section 5.4.1, except that the window size is much shorter, set to be the same as *update_interval* (i.e., the samples are averaged over an interval between 1 to 2 times the *update_interval*). We find that this dual-window measurement is better than the widely used exponentially-weighted moving-average method for closely tracking the short-term variations in the sampled statistics.

The baseline assignment of the service rate is to use the measured arrival rate $\mu_i(new) = \bar{\lambda}_i$. In addition to this, we decrease/increase the service rate based on the under-load/overload conditions, respectively.

We decide that a queue is overloaded when $lossfree_cnt_i > -burst_loss/p_{loss,i}^*$. Here the meaning of a negative target loss-free count $-loss_burst/P_{loss,i}^*$ is to provide an early response when the loss rate is within an additional *burst_loss* packet drops away from $P_{loss,i}^*$. In this work, we set *burst_loss* = 5 to account for simultaneous packet drops resulted from simultaneous arrivals at a full queue. In the case of queue overload, $\mu_i(new)$ has an additional increment from queue-length adjustment: $\left(\frac{\bar{N}_{q,i} - Q_i/2}{update_interval}\right)^+$. The purpose of this is to use an additional workload to bring the queue length down to the half-point of the buffer size when $\bar{N}_{q,i} > Q_i/2$. After replacing Q_i with $\mu_i(new) \tilde{D}_i$ based on (5.5), we have:

$$\mu_i(new) = \bar{\lambda}_i + \left(\frac{\bar{N}_{q,i} - \mu_i(new) \tilde{D}_i/2}{update_interval}\right)^+ \quad (5.13)$$

The solution is:

$$\mu_i(new) = \begin{cases} \frac{\bar{\lambda}_i + \bar{N}_{q,i}/update_interval}{1 + \tilde{D}_i/(2update_interval)} & \text{if } \bar{N}_{q,i} \geq \bar{\lambda}_i \tilde{D}_i/2 \\ \bar{\lambda}_i & \text{otherwise} \end{cases} \quad (5.14)$$

Similarly, we decide a queue is under-loaded when $lossfree_cnt_i \leq -burst_loss/p_{loss,i}^*$.

In this case, we set

$$\mu_i(new) = \max\{\mu_i(prev), \bar{\lambda}_i\}, \quad (5.15)$$

The calculated $\mu_i(new)$ is then checked against constraint (5.4) and we have:

$$\mu_i(new) = \max\{\mu_i(new), \bar{N}_q/\tilde{D}_i, \mu_{\min}\}, \quad (5.16)$$

where $\mu_{\min}$ is the minimum service rate reserved for each class to avoid starving a traffic class particularly when it transitions from idle to active.

The service rate $\mu_i(new)$ is then converted to service weight $w_i(new)$ for a WFQ scheduler. Note that $\mu_i(new)$ is the minimum service rate in a WFQ style scheduler because the unused service rate (weight) for some temporally idle classes will be proportionally allocated to the other busy classes. When there is congestion, i.e., not enough bandwidth to satisfy every $\mu_i(new)$, we use a strict priority in the service weight allocation procedure. That is, higher priority classes can “steal” service weights from lower priority classes until the service weight of a lower priority class reaches its minimum ($\mu_i(\min)$). In addition, we always change local service weights first before sending a Congestion_Alarm signal to the core provisioning module (discussed in Section 5.5.) to reduce the arrival rate which would require a network-wide adjustment of ingress traffic conditioners at edge nodes.

Similarly, when there is a persistent under-load in the queues, an increasing arrival rate is signalled (via the LinkState_Update signal) to the core provisioning module. An increase in the arrival rate is deferred to a periodic network-wide rate re-alignment algorithm which operates over longer time-scales. In other words, the control system’s response to rate reduction is immediate, while, on the other hand, its response to rate increase to improve utilization is delayed to limit any oscillation in rate allocation. In general, the time-scale of changing ingress router bandwidth

should be one order of magnitude greater than the maximum round trip delay across the core network in order to smooth out the traffic variations due to the transport protocol's flow control algorithm. Therefore, we introduce two control hystereses to the dynamic adjustment algorithm (Figure 5-3 line (18)), in the form of a 10% bandwidth threshold and a 5 second delay.

The pseudo code for the node algorithm is detailed in Figure 5-3.

dynamic adjustment algorithm

```

(1) upon the expiration of the update_interval timer or
    the arrival of early warning events from the virtual queues:
(2) IF early warning event, reset update_interval timer
(3) FOR all classes  $1, \dots, n$ 
(4)   retrieve measurements:  $\bar{\lambda}_i$ , and lossfree_cnti
(5)   IF lossfree_cnti >  $-burst\_loss/p_{loss,i}^*$  // overload
(6)     use Eqn(5.14) to calculate service weight
(7)   ELSE // under-load
(8)     use Eqn(5.15) to calculate service weight
(9)   use Eqn(5.16) to enforce a lower bound on  $\mu(new)$ 
(10)  IF remaining service bandwidth <  $\mu_i(new)$ 
(11)    adjust  $\mu_i(new)$  and set all  $\mu_j(new), j > i$  to  $\mu_{min}$ 
(12)    send CongestionAlarm signal
(13)    RETURN
(14)  adjust buffer size based on Eqn(5.5)
(15)  calculate  $\kappa_i$  for virtual queue based on Eqn(5.11)
(16)  scale virtual queue service rate to  $\kappa_i \mu_i(new)$ , and buffer size to  $\kappa_i Q_i(new)$ 
(17) END FOR
(18) IF remaining service bandwidth > 10% linerate for a duration > 5s
(19)   send LinkStateUpdate signal to increase arrival rate.
(20) RETURN

```

virtual queue prediction algorithm

```

(1) upon the arrival of class i packets:
(2) IF lossfree_cnti(now) >  $1/P_{loss,i}^*$  AND lossfree_cnti(now) > lossfree_cnti(prev)
    AND CongestionAlarm signal not present for classes  $j \leq i$ 
(3)   invoke the dynamic adjustment algorithm
(4)   lossfree_cnti(prev) = lossfree_cnti(now)
(5)   RETURN

```

Figure 5-3: Node Provisioning Algorithm Pseudo-code

5.5. Dynamic Core Provisioning

Our core provisioning algorithm has two functions: to reduce edge bandwidth immediately after receiving a *Congestion_Alarm* signal from a node provisioning module, and to provide periodic bandwidth re-alignment to establish a modified max-min bandwidth allocation for traffic aggregates. We will focus on the first function and discuss the latter function in Section 5.5.3.

5.5.1 Core Traffic Load Matrix

We consider a core network with a set $\mathcal{L} \triangleq \{1, 2, \dots, L\}$ of link identifiers of unidirectional links. Let c_l be the finite capacity of link $l, l \in \mathcal{L}$.

A core network traffic load distribution consists of a matrix $\mathbf{A} = \{a_{l,i}\}$ that models per-DiffServ-aggregate traffic distribution on links $l \in \mathcal{L}$, where the value of $a_{l,i}$ indicates the portion of the i th traffic aggregate that passes link l . Let the link load vector be $\mathbf{c}$ and ingress traffic vector be $\mathbf{u}$, whose coefficient u_i denotes a traffic aggregate of one service class at one ingress point³. In addition, this matrix formulation also supports multiple service classes. Let J be the total number of service classes. Without loss of generality, we can rearrange the columns of $\mathbf{A}$ into J sub-matrices, one for each class, which is: $\mathbf{A} = [\mathbf{A}(1):\mathbf{A}(2):\dots:\mathbf{A}(J)]$. Similarly, $\mathbf{u} = [\mathbf{u}(1):\mathbf{u}(2):\dots:\mathbf{u}(J)]$.

The constraint of link capacity leads to: $\mathbf{A}\mathbf{u}^T \leq \mathbf{c}$. Figure 5-4 illustrates an example network topology and its corresponding traffic matrix. In this figure, node 1, 2, 3, and 4 are edge nodes, while node 5 and 6 are core nodes. All the links are unidirectional. To better explain the construct of the traffic load matrix, we use the construct of the third column of the matrix $\mathbf{A}$: $\mathbf{a}_{\cdot,3}$ as an example. $\mathbf{a}_{\cdot,3}$ represents the

³Note that a network customer may contribute traffic to multiple u_i for multiple service classes and at multiple network access points.

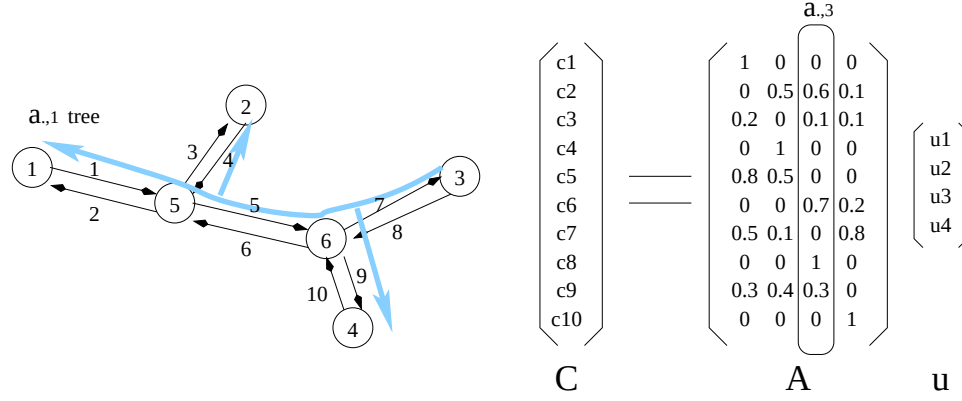


Figure 5-4: Example of a Network Topology and its Traffic Matrix

traffic distribution tree rooted at node 3, which is highlighted in the figure. Each entry $a_{l,3}$ represents the portion of node 3's incoming traffic that passes link l . For example, since 100% of node 3's incoming traffic passes through link 8, $a_{8,3} = 1$. Then at node 6, node 3's traffic is split between link 6 and 9 with a ratio of 7 : 3, therefore $a_{6,3} = 0.7$, and $a_{9,3} = 0.3$. The 70% of traffic on link 6 is further split between link 2 and 3 with a ratio of 6 : 1, as a result, we have $a_{2,3} = 0.6$, and $a_{3,3} = 0.1$. All the other entries in $\mathbf{a}_{\cdot,3}$ are zero since they model the reserve links.

The construction of matrix $\mathbf{A}$ is based on the measurement of its column vectors $\mathbf{a}_{\cdot,i}$, each represents the traffic distribution of an ingress aggregate u_i over the set of links $\mathcal{L}$. In addition, the measurement of u_i gives the trend of external traffic demands. In a DiffServ network, ingress traffic conditioners need to perform per-profile (usually per customer) policing or shaping. Therefore, traffic conditioners can also provide per-profile packet counting measurements without any additional operational cost. This alleviates the need to place measurement mechanisms at customer premises. We adopt this simple approach to measurement, which is advocated in [36] and measure both u_i and $\mathbf{a}_{\cdot,i}$ at the ingress points of a core network, rather than measuring at the egress points which is more challenging. The external traffic demands u_i is simply measured by packet counting at profile meters using ingress

traffic conditioners. The traffic vector $\mathbf{a}_{\cdot,i}$ is *inferred* from the flow-level packet statistics collected at a profile meter. Some additional packet probing (e.g., traceroute) or sampling (e.g., see [30]) methods can be used to improve the measurement accuracy of intra-domain traffic matrix. Last, with the addition of MPLS tunnels, fine granularity traffic measurement data is available for each tunnel. In this case, the calculation of the traffic matrix can be made more accurate. For example, in Figure 5-4, if there is an MPLS tunnel from node 3 to node 1 to accurately report the traffic volume, $a_{2,3}$ can be calculated exactly, and the inference of $a_{9,3}$, $a_{6,3}$, and $a_{3,3}$ can also be more accurately determined after knowing the value of $a_{2,3}$.

5.5.2 Edge Rate Reduction Policy

Given the measured traffic load matrix $\mathbf{A}$ and the required bandwidth reduction $\{-c_l^\delta(i)\}$ at link l for class i , the allocation procedure *Regulate_Ingress_Down()* needs to find the edge bandwidth reduction vector $-\mathbf{u}^\delta = -[\mathbf{u}^\delta(1):\mathbf{u}^\delta(2):\dots:\mathbf{u}^\delta(J)]$ such that: $\mathbf{a}_{l,\cdot}(j) * \mathbf{u}^\delta(j)^T = c_l^\delta(j)$, where $0 \leq u_i^\delta \leq u_i$.

When $\mathbf{a}_{l,\cdot}$ has more than one nonzero coefficients, there is an infinite number of solutions satisfying the above equation. We will choose one based on optimization policies such as fairness, minimizing the impact on other traffic and a combination of both. For clarity, we will drop the class (j) notation since the operations are the same for all classes.

The policies for edge rate reduction may be optimized for two quite different objectives.

5.5.2.1 Equal Reduction

Equal reduction minimizes the variance of rate reduction among various traffic aggregates, i.e.,

$$\min_i \left\{ \sum_{i=1}^n \left(u_i^\delta - \frac{\sum_{i=1}^n u_i^\delta}{n} \right)^2 \right\} \quad (5.17)$$

with constraints $0 \leq u_i^\delta \leq u_i$ and $\sum_{i=1}^n a_{l,i} u_i^\delta = c_l^\delta$. Using Kuhn-Tucker condition [62], we have:

Proposition 11 *The solution to the problem of minimizing the variance of rate reductions comprises three parts:*

$$\forall i \text{ with } a_{l,i} = 0, \text{ we have } u_i^\delta = 0; \quad (5.18)$$

then for notation simplicity, we re-number the remaining indices with positive $a_{l,i}$ as $1, 2, \dots, n$; and

$$u_{\sigma(1)}^\delta = u_{\sigma(1)}, \dots, u_{\sigma(k-1)}^\delta = u_{\sigma(k-1)}; \text{ and} \quad (5.19)$$

$$u_{\sigma(k)}^\delta = \dots = u_{\sigma(n)}^\delta = \frac{c_l^\delta - \sum_{i=1}^{k-1} a_{l,\sigma(i)} u_{\sigma(i)}}{\sum_{i=k}^n a_{l,\sigma(i)}}, \quad (5.20)$$

where $\{\sigma(1), \sigma(2), \dots, \sigma(n)\}$ is a permutation of $\{1, 2, \dots, n\}$ such that $u_{\sigma(i)}$ is sorted in increasing order, and k is chosen such that:

$$c_{eq}(k-1) < c_l^\delta \leq c_{eq}(k), \quad (5.21)$$

where $c_{eq}(k) = \sum_{i=1}^k a_{l,\sigma(i)} u_{\sigma(i)} + u_{\sigma(k)} \sum_{i=k+1}^n a_{l,\sigma(i)}$.

Equal reduction gives each traffic aggregate the same amount of rate reduction until the rate of a traffic aggregate reaches zero.

Remark: A variation of the equal reduction policy is proportional reduction: to reduce each of the aggregates contributing traffic to bottleneck link l an amount proportional to its total bandwidth. In particular, with $\alpha = c_l^\delta / (\sum_{\forall i, a_{l,i} > 0} a_{l,i} u_i)$, we have:

$$u_i^\delta = \begin{cases} 0 & \forall i \text{ with } a_{l,i} = 0 \\ \alpha u_i & \text{else.} \end{cases} \quad (5.22)$$

5.5.2.2 Minimal Branch-Penalty Reduction

A concern that is unique to DiffServ provisioning is to minimize the penalty on traffic belonging to the same regulated traffic aggregate that passes through non-congested branches of the routing tree. We call this effect the “branch-penalty”, which is caused by policing/shaping traffic aggregates at an ingress router. For example, in Figure 5-4, if link 7 is congested, the traffic aggregate #1 is reduced before entering link 1. Hence penalizing a portion of traffic aggregate #1 that passes through link 3 and 9.

The total amount of branch-penalty is $\sum_{i=1}^n (1 - a_{l,i}) u_i^\delta$ since $(1 - a_{l,i})$ is the proportion of traffic not passing through the congested link. Because of the constraint that $\sum_{i=1}^n a_{l,i} u_i^\delta = c_l^\delta$, we have $\sum_{i=1}^n (1 - a_{l,i}) u_i^\delta = \sum_{i=1}^n u_i^\delta - c_l^\delta$. Therefore, minimizing the branch-penalty is equivalent to minimizing the total bandwidth reduction, that is:

$$\min \sum_{i=1}^n (1 - a_{l,i}) u_i^\delta \iff \min \sum_{i=1}^n u_i^\delta \quad (5.23)$$

with constraints $0 \leq u_i^\delta \leq u_i$ and $\sum_{i=1}^n a_{l,i} u_i^\delta = c_l^\delta$.

Proposition 12 *The solution to the minimizing branch-penalty problem comprises three parts:*

$$u_{\sigma(1)}^\delta = u_{\sigma(1)}, \dots, u_{\sigma(k-1)}^\delta = u_{\sigma(k-1)}; \quad (5.24)$$

$$u_{\sigma(k)}^\delta = \frac{c_l^\delta - \sum_{i=1}^{k-1} a_{l,\sigma(i)} u_{\sigma(i)}^\delta}{a_{l,\sigma(k)}}; \text{ and} \quad (5.25)$$

$$u_{\sigma(k)}^\delta = \cdots = u_{\sigma(n)}^\delta = 0, \quad (5.26)$$

where $\{\sigma(1), \sigma(2), \dots, \sigma(n)\}$ is a permutation of $\{1, 2, \dots, n\}$ such that $a_{l,\sigma(i)}$ is sorted in decreasing order, and k is chosen such that:

$$c_{br}(k-1) < c_l^\delta \leq c_{br}(k), \quad (5.27)$$

where $c_{br}(k) = \sum_{i=1}^k a_{l,\sigma(i)} u_{\sigma(i)}^\delta$.

Proof: A straightforward proof by contradiction can be constructed as follows:

Let's assume that there is another rate reduction vector $\mathbf{v}^\delta \neq \mathbf{u}^\delta$ such that $\mathbf{v}^\delta$ minimizes the objective function (5.23), that is $\sum_{i=1}^n v_i^\delta < \sum_{i=1}^n u_i^\delta$. This inequality, together with the fact that $u_{\sigma(i)}^\delta$ ($\forall i < k$) reaches the maximum possible value, lead to the existence of at least one pair of indices j and m , where $j < k$ and $m \geq k$, such that $a_{l,j} > a_{l,m} > 0$; $v_{\sigma(j)}^\delta < u_{\sigma(j)}^\delta$ and $v_{\sigma(m)}^\delta > u_{\sigma(m)}^\delta$. Now we can construct a third vector $\mathbf{w}^\delta$ as follows: $w_{\sigma(i)}^\delta = v_{\sigma(i)}^\delta$, $i \neq j, m$, $w_{\sigma(j)}^\delta = v_{\sigma(j)}^\delta + \epsilon/a_{l,\sigma(j)}$, and $w_{\sigma(m)}^\delta = v_{\sigma(m)}^\delta - \epsilon/a_{l,\sigma(m)}$. Here $0 < \epsilon < \min \left\{ a_{l,\sigma(j)} (v_{\sigma(j)}^\delta - v_{\sigma(j)}^\delta), a_{l,\sigma(m)} v_{\sigma(m)}^\delta \right\}$ so that both $w_{\sigma(j)}^\delta$ and $w_{\sigma(m)}^\delta$ are positive. It is clear that $\sum_{i=1}^n a_{l,i} w_i^\delta = \sum_{i=1}^n a_{l,i} v_i^\delta = c_l^\delta$. However, because $a_{l,\sigma(j)} > a_{l,\sigma(m)}$, we have $\sum_{i=1}^n w_i^\delta = \sum_{i=1}^n v_i^\delta - \epsilon(1/a_{l,\sigma(m)} - 1/a_{l,\sigma(j)}) < \sum_{i=1}^n v_i^\delta$. This contradicts the assumption that $\mathbf{v}^\delta$ minimizes the objective function (5.23). $\square$

The solution is to sequentially reduce the u_i with the largest $a_{l,i}$ to zero, and then move on to the u_i with the second largest $a_{l,i}$ until the sum of reductions amounts to c_l^δ .

Remark: A variation of the minimal branch-penalty solution is to sort based on

$a_{l,\sigma(i)}u_{\sigma(i)}$ rather than $a_{l,\sigma(i)}$. This approach first penalizes the aggregates with the largest volume across the link (i.e., the “elephants”). This solution minimizes the number of traffic aggregates affected by the rate reduction procedure.

5.5.2.3 Penrose-Moore Inverse Reduction

It is clear that equal reduction and minimizing branch-penalty have conflicting objectives. Equal reduction attempts to provide the same amount of reduction to all traffic aggregates. In contrast, minimal branch-penalty reduction always depletes the bandwidth associated with the traffic aggregate with the largest portion of traffic passing through the congested link. To balance these two competing optimization objectives, we propose a new policy that minimizes the Euclidean distance of the rate reduction vector u^δ :

$$\min \left\{ \sum_{i=1}^n (u_i^\delta)^2 \right\}, \quad (5.28)$$

with constraints $0 \leq u_i^\delta \leq u_i$ and $\sum_{i=1}^n a_{l,i}u_i^\delta = c_l^\delta$.

Similar to the solution of the minimizing variance problem in the equal reduction case, we have:

Proposition 13 *The solution to the problem of minimizing the Euclidean distance of the rate reduction vector comprises three parts:*

$$\forall i \text{ with } a_{l,i} = 0, \text{ we have } u_i^\delta = 0; \quad (5.29)$$

then for notation simplicity, we re-number the remaining indices with positive $a_{l,i}$ as $1, 2, \dots, n$; and

$$u_{\sigma(1)}^\delta = u_{\sigma(1)}, \dots, u_{\sigma(k-1)}^\delta = u_{\sigma(k-1)}; \text{ and} \quad (5.30)$$

$$\frac{u_{\sigma(k)}^\delta}{a_{l,\sigma(k)}} = \dots = \frac{u_{\sigma(n)}^\delta}{a_{l,\sigma(n)}} = \frac{c_l^\delta - \sum_{i=1}^{k-1} a_{l,\sigma(i)} u_{\sigma(i)}^\delta}{\sum_{i=k}^n a_{l,\sigma(i)}^2}, \quad (5.31)$$

where $\{\sigma(1), \sigma(2), \dots, \sigma(n)\}$ is a permutation of $\{1, 2, \dots, n\}$ such that $u_{\sigma(i)}/a_{l,\sigma(i)}$ is sorted in increasing order, and k is chosen such that:

$$c_{pm}(k-1) < c_l^\delta \leq c_{pm}(k), \quad (5.32)$$

where $c_{pm}(k) = \sum_{i=1}^k a_{l,\sigma(i)} u_{\sigma(i)}^\delta + (u_{\sigma(k)}/a_{l,\sigma(k)}) \sum_{i=k+1}^n a_{l,\sigma(i)}^2$.

Equation (5.31) is equivalent to the Penrose-Moore (P-M) matrix inverse [19], in the form of

$$\begin{aligned} & [u_{\sigma(k)}^\delta u_{\sigma(k+1)}^\delta \cdots u_{\sigma(n)}^\delta]^T = \\ & [a_{l,\sigma(k)} a_{l,\sigma(k+1)} \cdots a_{l,\sigma(n)}]^+ * (c_l^\delta - \sum_{i=1}^{k-1} a_{l,\sigma(i)} u_{\sigma(i)}^\delta), \end{aligned} \quad (5.33)$$

where $[\cdots]^+$ is the P-M matrix inverse. In particular, for an $n \times 1$ vector $\mathbf{a}_{l,\cdot}$, the P-M inverse is a $1 \times n$ vector $\mathbf{a}_{l,\cdot}^+$, where $a_{l,i}^+ = a_{l,i}/(\sum_{i=1}^n a_{l,i}^2)$.

We name this policy as the ‘‘P-M inverse reduction’’ because of the property of P-M matrix inverse. The P-M matrix inverse always exists and is unique, and gives the least Euclidean distance among all possible solution satisfying the optimization constraint.

Proposition 14 *The performance of the P-M inverse reduction lies between the equal reduction and minimal branch-penalty reduction. In terms of fairness, it is better than the minimal branch-penalty reduction and in terms of minimizing branch-penalty, it is better than the equal reduction.*

Proof: By simple manipulation, the minimizing objective of P-M inverse is equivalent to the following:

$$\min \left\{ \sum_{i=1}^n \left(u_i^\delta - \left(\sum_{i=1}^n u_i^\delta \right) / n \right)^2 + \left(\sum_{i=1}^n u_i^\delta \right)^2 / n \right\}. \quad (5.34)$$

The first part of this formula is the optimization objective of the equal reduction policy. The second part of formula (5.34) is scaled from the optimization objective of the minimizing branch penalty policy by squaring and division to be comparable to the objective function of equal reduction; that is, the P-M inverse method minimizes the sum of the objective functions minimized by the equal reduction and minimal branch penalty methods, respectively. Therefore, the P-M inverse policy has a smaller value in the first part of formula (5.34) than what the minimal branch penalty policy has; and a smaller value in the second part of (5.34) than the corresponding value the equal reduction policy has. Hence, the P-M inverse method balances the trade-off between equal reduction and minimal branch penalty. $\square$

It is noted that the P-M inverse reduction policy is not the only method that balances the optimization objectives of fairness and minimizing branch penalty. However, we choose it because of its clear geometric meaning (i.e., minimizing the Euclidean distance) and its simple closed-form formula.

5.5.2.4 Algorithm Implementation

The implementation complexity of the preceding three reduction algorithms lies in the boundary conditions where the rates of some traffic aggregates are reduced to zero. Because all three algorithms have the similar structure, we can show the procedure of these algorithms in a coherent manner as in Figure 5-5.

5.5.3 Edge Rate Alignment

Unlike edge rate reduction, which is triggered locally by a link scheduler that needs to limit the impact on ingress traffic aggregates, the design goal for the periodic rate alignment algorithm is to re-align the bandwidth distribution across the network for

-
- (1) sort the indices i of traffic aggregates
based on :
 - the increasing order of u_i for ER,
 - the decreasing order of $a_{l,i}$ for BR,
 - the increasing order of $u_i/a_{l,i}$ for PM;
 - (2) locate the index k in the sorted index list
based on :
 - Inequality (5.21) for ER,
 - Inequality (5.27) for BR,
 - Inequality (5.32) for PM;
 - (3) calculate reduction based on:
 - equations (5.18–5.20) for ER,
 - equations (5.24–5.26) for BR,
 - equations (5.29–5.31) for PM.
-

Figure 5-5: Edge Rate Reduction Algorithm Pseudo-code

various classes of traffic aggregates and to re-establish the ideal max-min fairness property.

However, we need to extend the max-min fair allocation algorithm given in [10] to reflect the point-to-multipoint topology of a DiffServ traffic aggregate. Let $\mathcal{L}^u$ denote the set of links that are not saturated and $\mathcal{P}$ be the set of ingress aggregates that are *not bottlenecked*, (i.e., have no branch of traffic passing a saturated link). Then the procedure is given as in Figure 5-6.

-
- (1) identify the most loaded link l in the set
of non-saturated links:

$$l = \arg \min_{j \in \mathcal{L}^u} \left\{ x_j = \frac{c_j - \text{allocated_capacity}}{\sum_{i \in \mathcal{P}} a_{j,i}} \right\};$$
 - (2) increase allocation to all ingress
aggregates in $\mathcal{P}$ by x_l , and update the
allocated_capacity for links in $\mathcal{L}^u$;
 - (3) remove ingress aggregates passing l from $\mathcal{P}$,
and remove link l from $\mathcal{L}^u$;
 - (4) if $\mathcal{P}$ is empty, then stop; else go to (1).
-

Figure 5-6: Edge Rate Alignment Algorithm Pseudo-code

Our modification of step (1) changes the calculation of remaining capacity from

$(c_l - \text{allocated_capacity})/|\mathcal{P}|$ to $(c_l - \text{allocated_capacity})/\sum_{i \in \mathcal{P}} a_{l,i}$.

Remark: The convergence speed of max-min allocation for point-to-multipoint traffic aggregates is faster than for point-to-point aggregate because it is more likely that two traffic aggregates send traffic over the same congested link. In the extreme case, when all the traffic aggregates have portions of traffic over all the congested links, these aggregates are constrained by only one most congested bottleneck link. In this case, the algorithm takes one round to finish, and the allocation effect is equivalent to the equal reduction (in this case, “equal allocation”) method with respect the capacity of the most congested bottleneck link.

The edge rate alignment algorithm involves increasing edge bandwidth, which makes the operation fundamentally more difficult than the reduction operation. The problem is essentially the same as that found in multi-class admission control because we need to calculate the amount of offered bandwidth $c_l(i)$ at each link for every service class. Rather than calculate $c_l(i)$ simultaneously for all the classes, we take a sequential allocation approach. In this case, the algorithm waits for an interval (denoted `SETTLE_INTERVAL`) after bandwidth allocation for a higher priority. This allows the lower priority queues to take measurement on the impact of the changes, and to invoke `Regulate_Down()` if rate reduction is needed. The procedure is on a per class basis and follows the decreasing order of allocation priority.

5.6. Simulation Results

5.6.1 Simulation Setup

We evaluate our algorithms by simulation using the ns-2 simulator [100]. Unless otherwise stated, we use the default values in the standard ns-2 release for the simulation parameters.

We use the Weighted-Round-Robin scheduler which is a variant of the WFQ algorithm. In our simulation, we consider the performance for the four service classes which loosely correspond to the DiffServ Expedited Forwarding (EF), Assured Forward (AF1, and AF2), and best-effort (BE) classes. The order above represents the priority for bandwidth allocation. The initial service weights for the four class queues are 30, 30, 30 and 10, respectively, with a fixed total of 100. The minimum service weight $w_i(\min)$ for each class is 1. The initial buffer size is 30 packets for the EF class queue, 100 packets each of the AF1 and AF2 class queues, respectively, and 200 packets for the BE class queue.

The simulation network comprises eight nodes with traffic conditioners at the edge, as shown in Figure 5-7. The backbone links are configured with 6 Mb/s capacity with a propagation delay of 1 ms. The three backbone links (C1, C2 and C3) highlighted in the figure are overloaded in various test cases to represent the focus of our traffic overload study. The access links leading to the congested link have 5 Mb/s with a 0.1 ms propagation delay. The ingress traffic conditioners serve the purpose of ingress edge routers. Each conditioner is configured with one profile for each traffic source. The EF profile has a default peak rate of 500Kb/s and a bucket size of 10Kbits. The AF profile has a default peak rate of 1Mb/s and a token bucket of 80Kbits. For simplicity, we program the conditioners to drop packets that are not conforming to the leaky-bucket profile. The core provisioning algorithm will regulate the ingress traffic rates by changing the profiles in the traffic conditioners.

A combination of Constant-Bit-Rate (CBR), Pareto On-Off and Exponential On-Off traffic sources are used in the simulation, as well as applications including a large number of greedy FTP sessions and HTTP transactions. The starting time of all sources is a random variable uniformly distributed in $[0 \text{ } 5s]$. During the simulations, we will vary the peak rate or the number of sources to simulate different traffic load

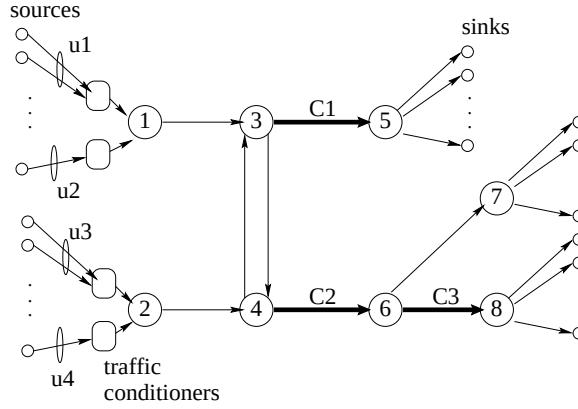


Figure 5-7: Simulated Network Topology

conditions. Except where specifically noted, we use the default values for all ns simulation parameters.

Throughout the simulations, we use the same set of DiffServ SLAs:

- for the EF class, the delay bound $D_1 = 0.1s$, the loss bound $P_1^* = 5 * 10^{-5}$;
- for the AF1 class, the delay bound $D_2 = 0.5s$, the loss bound $P_2^* = 5 * 10^{-4}$;
- for the AF2 class, the delay bound $D_3 = 1s$, the loss bound $P_3^* = 5 * 10^{-3}$.

For the BE class, there is no SLA that needs to be supported.

5.6.2 Dynamic Node Provisioning

The dynamic node provisioning algorithm interacts with the core provisioning algorithm via the Congestion_Alarm and LinkState_Update signals. To better stress test the node provisioning algorithm, we disable the alarm and update signals to the core provisioning algorithms in the simulations described in this section. In addition, we simplify the network shown in Figure 5-7 into a dumb-bell topology (by combining nodes 1 to 4 into one node, and nodes 5 to 8 into another node). The 5Mb/s link between these two “super” nodes will be the focus of simulations in this sub-section.

5.6.2.1 Service Differentiation Effect

We first use traces to highlight the impact of enabling and disabling the node provisioning algorithm on our service model. We compare the results where the algorithm is enabled and disabled.

We use 100 traffic sources: 20 CBR sources for the EF class; 30 Pareto On-Off sources for the AF1 class; and 40 and 10 Exponential On-Off sources for the AF2 and BE classes, respectively. Each source has the same average rate of 55 Kb/s, which translates into an average of a 110% load on the 5 Mb/s target link when all the sources are activated. The simulation trace lasts 100 s. To simulate the dynamics of traffic overload, we activate and stop the EF and AF1 class sources in a slow-start manner, i.e., the activation time for the EF and AF1 class traffic sources is uniformly distributed over the first 30 s. The stop time for the EF and AF1 sources is uniformly distributed over the last 40 s. With respect to the AF2 and BE sources, their slow-start activation time lies within the first 5 s. The stop time for the AF2 and BE sources corresponds to the end of the simulation period. As a result, congestion occurs between 30 and 60 seconds in the trace. The node provisioning algorithm *update_interval* is set to a value of 200 ms.

Accurately setting the service weights is very important to the operation of the scheduler in the case where the node provisioning algorithm is disabled because its service weights are not adjusted during the simulation. We use the exact information of the traffic load mixture to set the service weights to 23, 33, 43 and 1 for the EF, AF1, AF2, and BE classes, respectively. These settings yield a traffic intensity of 96%, 100% and 102% for the EF, AF1 and AF2 queues, respectively, while leaving a $w_{\min} = 1$ for the BE traffic during the congestion interval. These setting represent the best-case scenario for the scheduler (in the case where the node provisioning algorithm is disabled) to maintain service differentiation for services classes that

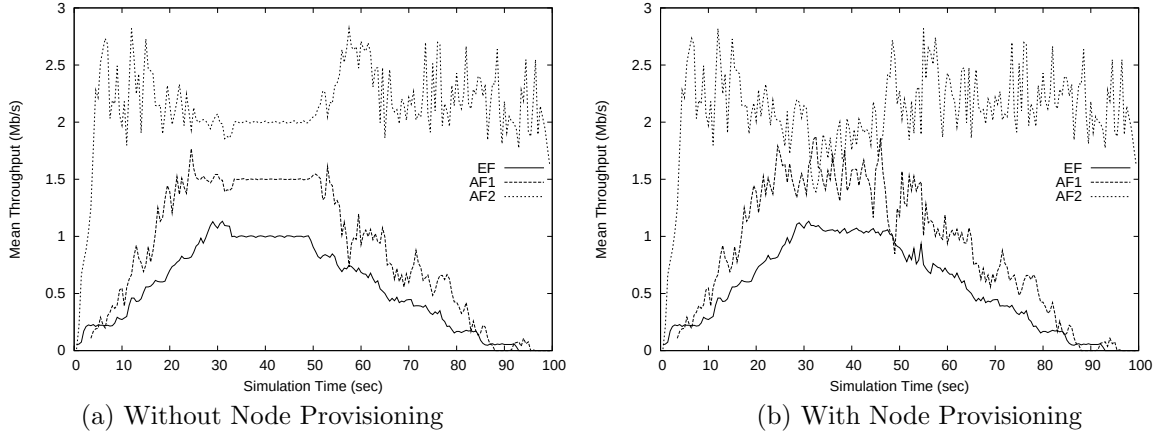


Figure 5-8: Node Provisioning Service Differentiation Effect: Throughput

have SLA concerns. We note that in practice, however, there is no prior knowledge of traffic load during congestion. Therefore, the setting of service weights in practice would be less ideal when comparing the performance of the scheduler in a system where the node provisioning algorithm is disabled. As we will show later, even with such a best-case advantage, the scheduler still under-performs the node provisioning algorithm in both delay and loss performance because a fixed set of service weights can not deal with the varying mixture of traffic loads from different classes.

The statistical traces collected in this simulation are end-to-end throughput, packet loss rate, and mean delay for all the classes except BE. Each sample is averaged over a window of 0.5 s from the per-packet samples.

Figure 5-8 shows the throughput trace. When the system is not overloaded, both plots exhibit the same shape of curve. During congestion between 30 and 60s into the trace, however, the plot with node provisioning disabled (5-8(a)) shows almost flat throughput curves for the EF, AF1 and AF2 classes, with a ratio of 2:3:4 matching the service weight setting, respectively. In contrast, significant variations occur for the results with the node provisioning algorithm enabled, as shown in Figure 5-8(b).

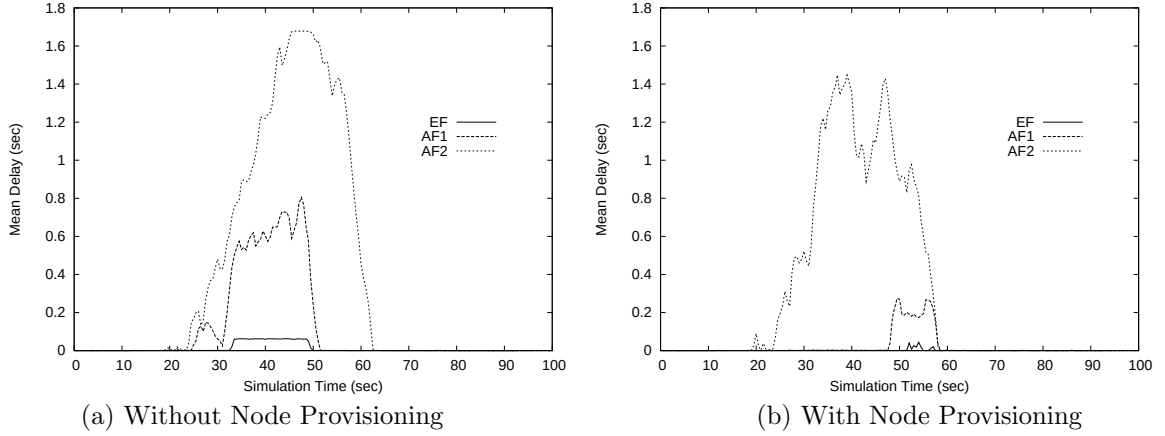


Figure 5-9: Node Provisioning Service Differentiation Effect: Mean Delay

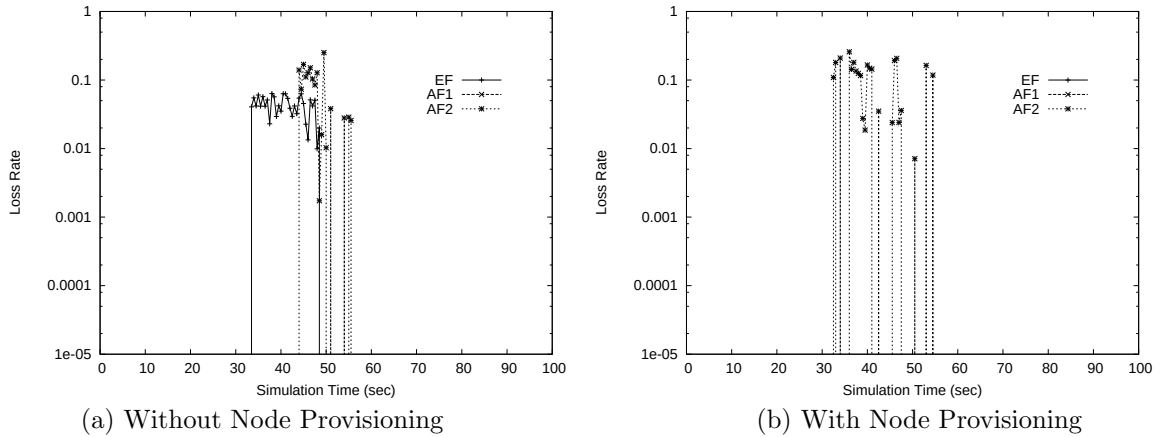


Figure 5-10: Node Provisioning Service Differentiation Effect: Loss

The effect of the node provisioning algorithm can be clearly observed in the delay plots of Figure 5-9(a) and (b). Unlike 5-9(a) where both AF1 and AF2 delays exceed their bound of 0.5s and 1s, respectively, 5-9(b) shows that only the AF2 class exceeds its delay bound. In addition, the delay values for all three classes are smaller than the results shown in 5-9(a).

In the packet loss comparison, the lack of loss differentiation is clearly evident in Figure 5-10(a), where both EF and AF2 classes have the same magnitude of loss rate of approximately 10%. In contrast, in 5-10(b) with node provisioning enabled,

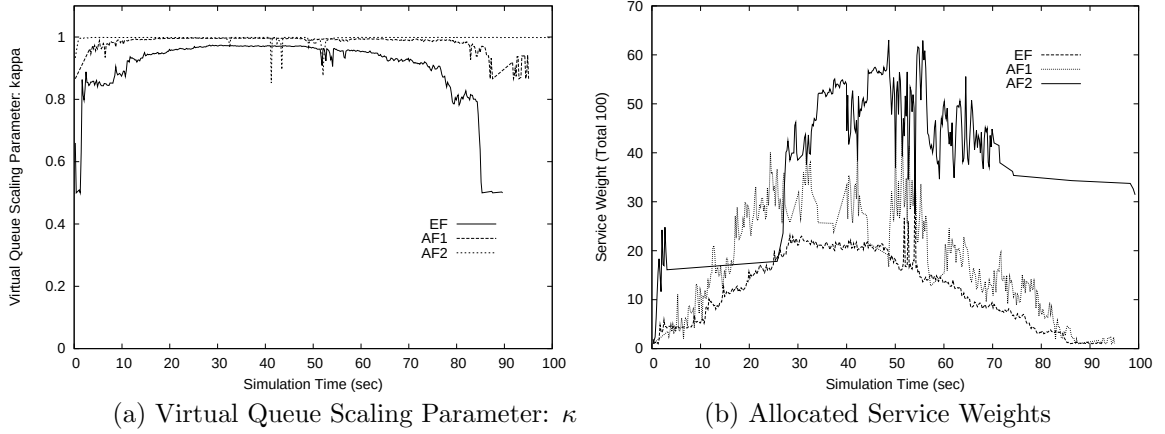


Figure 5-11: Node Provisioning Control Parameters

only AF2 has packet loss and the loss rate is comparable to the result shown in 5-10(b).

To better illustrate the internal operation of the node provisioning algorithm, in Figure 5-11(a) and (b), we present the trace of κ and allocated service weights. During congestion, κ remains close to 1 while during under-load κ value drops. This is because the under-load condition provide more room for traffic intensity to grow and therefore a smaller κ has to be used to provide early enough prediction of SLA violation. The service weight plot in 5-11(b) clearly shows the effect of the node provisioning algorithm in dealing with bursty arrivals.

5.6.2.2 Update Interval

In this set of simulation, we investigate the appropriate time-scale for the *update_interval* when invoking the node provisioning algorithm. An *update_interval* that is too small increases the variations in the measured traffic arrival rate and leads to frequent oscillations in bandwidth allocation. In contrast, an *update_interval* that is too large delays the detection of under-load in some traffic classes and hurts service differentiation.

We experiment with five different values of *update_interval*: 50ms, 100ms, 200ms, 500ms, 1s and 2s. There are a total 70 traffic sources, with 20% for the EF class, 30% for the AF1 class, 40% for the AF2 class and 10% for the BE class. The EF source is CBR with a peak rate of 100 Kb/s. The AF1 and AF2 sources are Pareto On-Off sources with default ns values: an average 0.5s for the on and off intervals, and a shape parameter with the value of 1.5. The AF2 sources have a peak rate of 200 Kb/s. The BE class sources are CBR with 100 Kb/s rate. We vary the peak rate for the AF1 class to change the offered load. The offered load is calculated as the ratio between the total arrival rate of the AF1 class and the available bandwidth to AF1 (which is the link capacity subtracted by the total EF traffic arrival rate). We note that this offered load is always no greater than the AF1 class queue traffic intensity because the AF1 class does not always consume all the available bandwidth excluding the EF traffic load.

Extensive statistics (e.g., delay, loss, service rate, arrival rate, etc.) are collected for each queueing classes at each network node, and for each flow from end-to-end. Most samples are collected when the node provisioning algorithm is invoked. Therefore the maximum sampling interval is the *update_interval*. The collected samples are then consolidated by the time-weighted average for statistics requiring averaging (e.g., traffic load, mean delay, and loss rate, etc.). Statistics like maximum delay are calculated from the maximum of all the collected samples. The loss rate samples are accumulated using the dual-window approach described in Section 5.4., with the measurement window τ_l set to 30 seconds for the EF class and 10 seconds for all the other classes. The collected samples are then consolidated by time-weighted average for statistics including loss rate, mean delay, and arrival rate.

Figure 5-12 shows both the packet loss and maximum delay performance. For the purpose of clarity, we only show the results for the AF1 class. Each sample

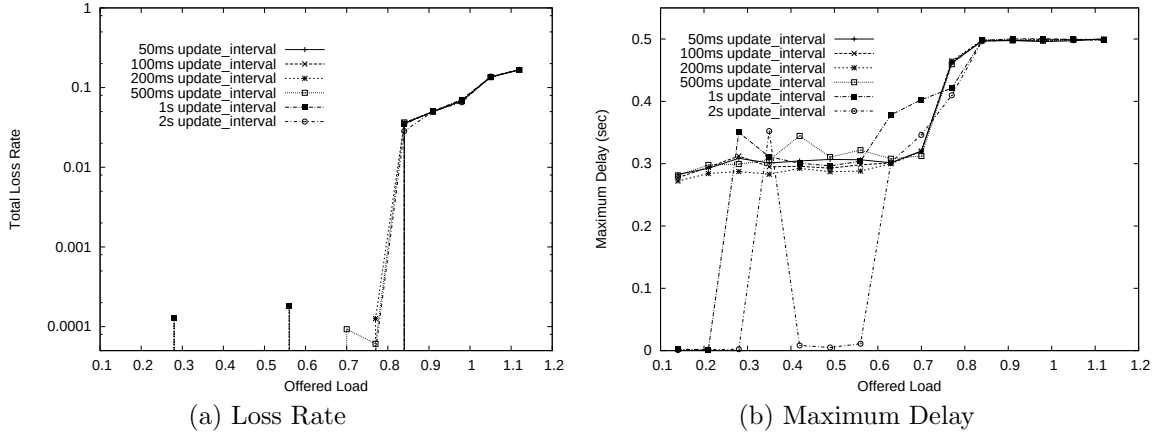


Figure 5-12: Node Provisioning Sensitivity to *update_interval*, AF1 Class with Pareto On-Off Traffic

point on the plot is a simulation run of 100s. In general, the algorithm performance is not very sensitive to the value of *update_interval*. This is expected because the node provisioning algorithm can also be invoked by the virtual queues detecting an onset of SLA violation. Among the small differences, we observe that *update_interval* $\geq 1s$ is not good because it has packet losses and large variation of the maximum delay under low offered load. In addition, we observe that an *update_interval* value of 200ms achieves low maximum delay relative to the other curves. This is consistently observed across the whole range of offered loads below 80%. When the offered load increases beyond 80% the system become over-loaded and the impact of different *update_interval* becomes negligible. In what follows, we will use an *update_interval* = 200ms for all the simulations.

It is also interesting to observe one feature of the node provisioning algorithm: namely the algorithm always tries to guarantee the delay bound first. We observe that beyond 80% load the loss rate starts to exceed the $5 * 10^{-4}$ bound, while the delay bound of 0.5s is always maintained even for an offered load exceeding 1.

5.6.2.3 Stress Test Under Bursty Traffic

We continue the preceding simulation runs with different traffic sources for the AF1 classes, including Pareto On-Off, Exponential On-Off and CBR traffic sources. Each sample point represents a simulation run of 1000s. We use the CBR traffic source to provide a baseline reference for the two bursty On-Off traffic types.

Figure 5-13 presents four sets of consolidated statistics for comparison. Figure 5-13(a) plots the percentage of time that the Congestion_Alarm is raised for the AF1 class. Since we disable the dynamic core provisioning algorithm to stress test the node algorithm, the alarm frequency becomes a good indicator of the node algorithm's capability of handling bursty traffic. It is also a convenient indicator of the performance boundary below which the delay bound $D_2 = 0.5s$ and loss bound $P_{loss,2}^* = 5 * 10^{-4}$ should hold and above which the loss rate and maximum delay will grow to exceed these bounds. We observe that the algorithm performs equally well for both Pareto and Exponential On-Off sources, even though the Pareto source is heavy-tailed and more bursty. It is clear that the algorithm can handle up to 70% load for both the Pareto and Exponential On-Off traffic under the D_2 and $P_{loss,2}^*$ bounds. For the CBR traffic, the sustainable load reaches 85% as observed from the loss and delay measurements in Figures 5-13(c) and (d), respectively. This is falls short of 100% because the CBR traffic is also bursty being an aggregate of 21 individual CBR sources.

Figure 5-13(b) shows the measured traffic intensity in the AF1 queue. Even though measuring the arrival rate is trivial, measuring the per-class service time is not easy for a multi-class queueing system. In the simulations, we use the sum of the per-packet transmission time and the Head-of-the-Line (HOL) waiting time as the total service time. The HOL waiting time is the time after a packet enters the HOL position of the queue, waiting for scheduler to finish serving the HOL packets of

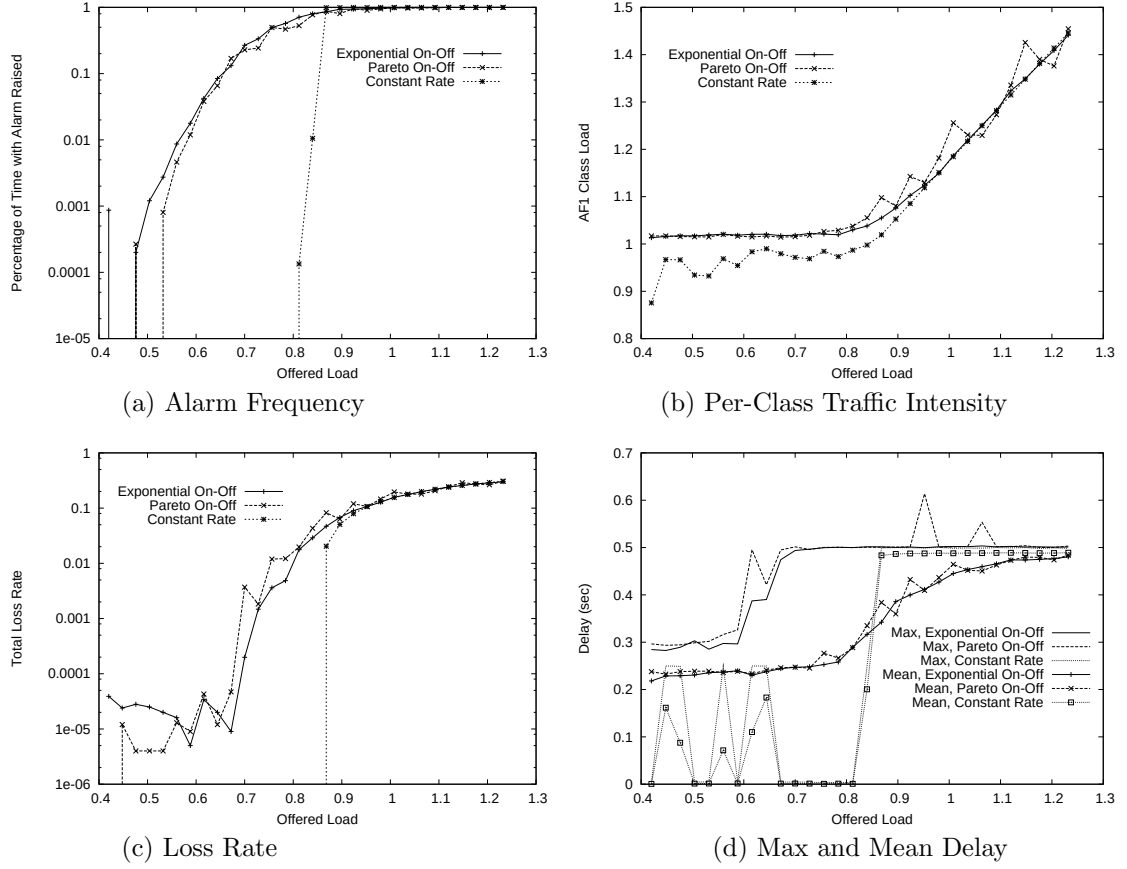


Figure 5-13: Node Provisioning Algorithm Performance, AF1 Class with Bursty Traffic Load

other queues. From this plot we can observe the algorithm's efficiency in allocating bandwidth. For the CBR traffic, the service bandwidth utilization remains at 100% until the incoming traffic exceeds the maximum service capability. For the Pareto and Exponential On-Off traffic, the utilization stays at 100% until the offered load reaches 50%. After that the utilization dips by about 10%. This is the amount of over allocation necessary to maintain the SLA.

Figures 5-13(c) and 5-13(d) plot the loss rate and maximum delay measured at this AF1 class queue, respectively. The results verify that when the alarm signal is not raised, the system performance will remain below the SLA bounds. Once again we observe that the algorithm gives precedence in guaranteeing the delay

bound first. Except two spikes for the Pareto source, all the maximum delay curves are below the 0.5s bound. In addition, one can also discover the fact that only when alarm frequencies exceed 10%, the loss rate will exceed the loss bound of $5 * 10^{-4}$. This is true for both the Pareto and Exponential On-Off sources, where the 10% alarm frequency corresponds well to the 70% maximum sustainable load, and for the CBR source, where the 10% alarm frequency matches the 85% maximum sustainable load. This information is important for the core provisioning algorithm as it allows the core algorithm to gauge the overload severity from the frequency of Congestion_Alarm signals sent by the node provisioning algorithms.

5.6.2.4 Scalability with Adaptive Applications

We further test our scheme with TCP applications including greedy FTP and transactional HTTP applications. Because TCP congestion control reacts to packet losses, the packet dropping action alone is also effective in reducing congestion for TCP. However, the adaptive flow control of TCP also will push the traffic load to 100% even with a small number of sources. To test our algorithm's performance in supporting a large number of TCP sources, we repeat the above test but instead of varying the peak rate of each source, we vary the number of TCP applications that are connected to the target node.

The results are shown in Figure 5-14 in the same setting as Figure 5-13. The traffic load for the EF, AF2 and BE classes remain the same as in the previous tests. We vary the number of the AF1 sessions: from 2 to 40 for greedy FTP traffic, and from 20 to 400 for web traffic. To better understand these results, we plot the FTP and HTTP results with a corresponding 1:10 ratio in the number of sessions on the x axis.

The web traffic is simulated using the ns-2 "PagePoolWebTraf" module. The

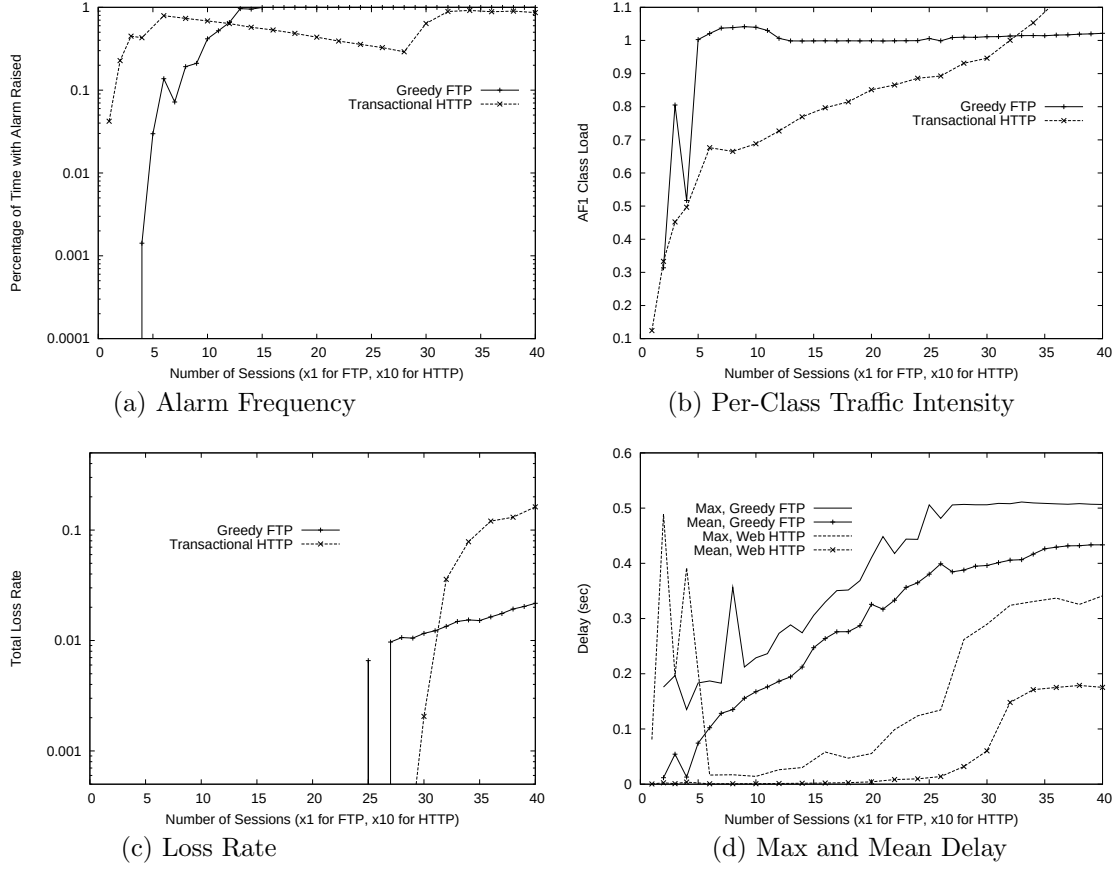


Figure 5-14: Node Provisioning Algorithm Performance, AF1 Class with TCP Applications

parameters for the web traffic are set to increase the traffic volume of each web session so that on the target link of 5Mb/s, queueing overload can occur. The inter-session time is exponentially distributed with a mean of 0.1 s. Each session size is a constant of 100 pages. The inter-page time is also exponentially distributed but with a mean of 5 s. Each page size is a constant of 5 objects, while the inter-object time is exponentially distributed with a mean of 0.05 s. Last, the object size has a distribution of Pareto of the Second Kind (also known as the Lomax distribution) with a shape value of 1.2 and average size of 12 packets (which is 12Kbytes).

In Figure 5-14(a), for both traffic sources, the alarm frequency rises above 10% for a small number of sessions, i.e., 5 sessions for FTP and 20 sessions for HTTP,

respectively. The average traffic intensity shown in 5-14(b), however, shows a difference. The FTP traffic intensity increases quickly to 100% and then stays at 100% after 5 sessions, while the HTTP traffic intensity increases gradually and reaches 100% much later at 220 sessions. These two plots indicate that the HTTP traffic is more bursty than the FTP traffic because for the HTTP traffic, its alarm frequency rises quicker while its average traffic intensity rises much slower than the FTP traffic. The FTP traffic, on the other hand, is less bursty only because its average load reaches 100% for most of the cases. However, even with a large value of alarm frequency, the system performs well for a wide range of number of sessions. The loss rate exceeds $5 * 10^{-4}$ at 25 FTP sessions or 300 HTTP sessions. The delay bound of 0.5 s is always met for the HTTP traffic. For the FTP traffic, because of the heavy traffic load, the delay bound is first violated at 25 FTP sessions, but is not exceeded much after that point, as shown in 5-14(d).

In summary, the stress test results from both bursty On-Off and TCP application traffic have shown that the node provisioning algorithm will guarantee the delay and loss bounds when there is no alarm raised, and also with a alarm frequency below 10%. When there is a SLA violation, the algorithm will first meet the delay bound sacrificing the loss bound. For adaptive applications like TCP which respond to packet losses, this approach has shown to be effective even without the involvement of core provisioning algorithms.

5.6.3 Dynamic Core Provisioning

5.6.3.1 Effect of Rate Control Policy

In this section, we use test scenarios to verify the effect of different rate control policies in our core provisioning algorithm. We only use CBR traffic sources in the following tests to focus on the effect of these policies.

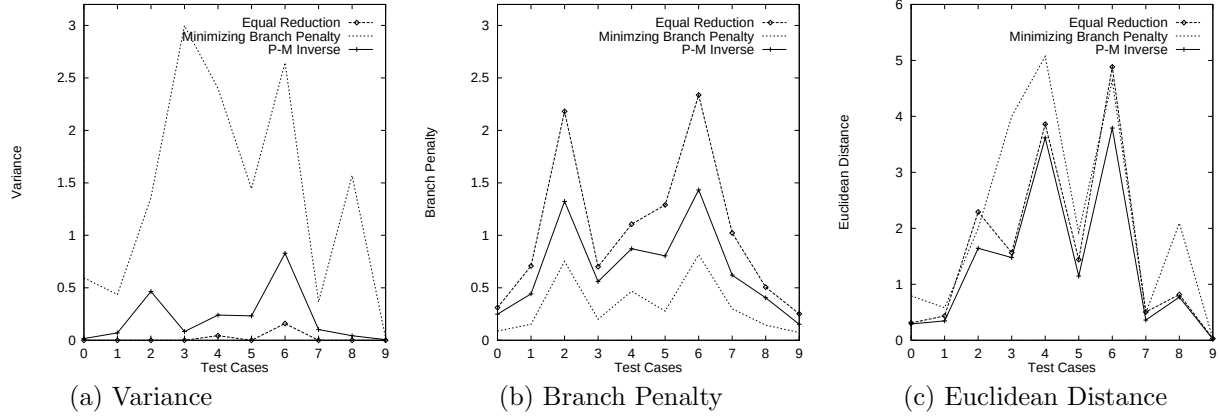


Figure 5-15: Reduction Policy Comparison (Ten Independent Tests)

Table 5.1 gives the initial traffic distribution of the four EF aggregates comprising only CBR flows in the simulation network, as shown in Figure 5-7. For clarity, we only show the distribution over the three highlighted links (C_1 , C_2 and C_3). The first three data-rows form the traffic load matrix $\mathbf{A}$, and the last data-row is the input vector $\mathbf{u}$.

In Figure 5-15, we compare the metrics for equal reduction, minimal branch-penalty and the P-M inverse reduction under ten randomly generated test cases. Each test case starts with the same initial load condition, as given in Table 5.1. The change is introduced by reducing the capacity of one backbone link to cause congestion which subsequently triggers rate reduction.

Figure 5-15(a) shows the fairness metric: the variance of rate reduction vector

Table 5.1: Traffic Distribution Matrix

Bottleneck Link	User Traffic Aggregates			
	U_1	U_2	U_3	U_4
C_1	0.20	0.25	0.57	0.10
C_2	0.80	0.75	0.43	0.90
C_3	0.40	0.50	0.15	0.80
Load (Mb/s)	1.0	0.8	1.4	2.0

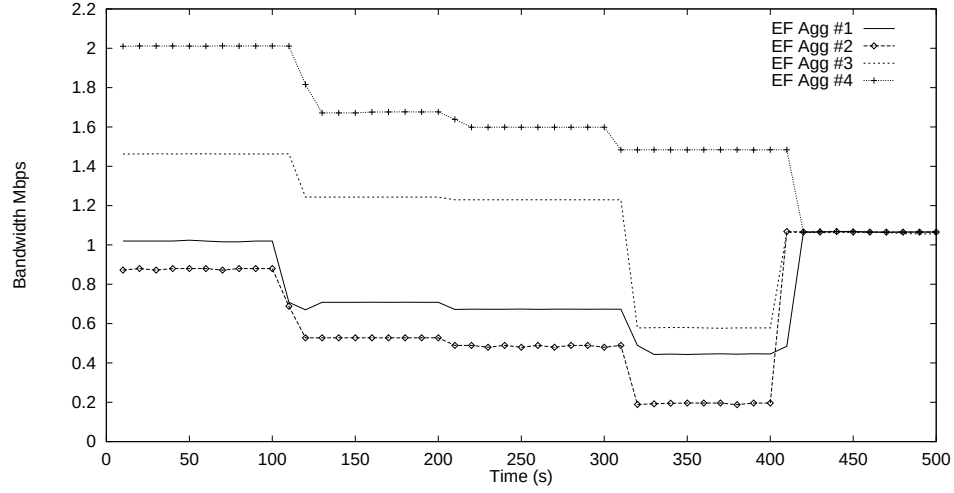


Figure 5-16: Core Provisioning Allocation Result, Default Policies

$\mathbf{u}^\delta$. The equal reduction policy always generates the smallest variance, in most of the cases the variances are zero, and the non-zero variance cases are caused by the boundary conditions where some of the traffic aggregates have their rates reduced to zero. Here we observe that the P-M inverse method always gives a variance value between those of equal reduction and minimizing branch penalty. Similarly, Figure 5-15(b) illustrates the branch penalty metric: $\sum_i (1 - a_{l,i}) u_i^\delta$. In this case, the minimizing branch penalty method consistently has the lowest branch penalty values, followed by the P-M inverse method. The last figure, Figure 5-15(c), shows the Euclidean distance of $\mathbf{u}^\delta$, i.e., $\sum_i (u_i^\delta)^2$. In this case, the P-M inverse method always has the lowest values, while between the equal reduction and minimizing branch penalty methods, there is no clear winner.

The results support our assertion that the P-M Inverse method balances the trade-off between equal reduction and minimal branch penalty.

In Figure 5-16, we plot the time sequence of rate-regulating results using the default policies of our core provisioning algorithm, i.e., the P-M inverse method for rate reduction and the modified max-min fair rate alignment method for rate

re-alignment. The traffic dynamics are introduced by sequentially changing link capacity of C_1 , C_2 and C_3 as follows:

1. at 100s into the trace, C_2 capacity is reduced to 3 Mb/s and requires a bandwidth reduction of 0.8 Mb/s from ingress traffic conditioners
2. at 200s into the trace, C_3 capacity is reduced to 2 Mb/s, and requires a bandwidth reduction of 0.1 Mb/s,
3. at 300s into the trace, C_1 capacity is reduced to 0.5 Mb/s, and requires a bandwidth reduction of 0.6 Mb/s, and
4. at 400s into the trace, C_1 notices a capacity increase to 6 Mb/s, which leaves C_3 the only bottleneck.

The first three cases of reduction are also the first three test cases used in Figure 5-15⁴. The last case invokes a bandwidth increment rather than a reduction. In this case, we use the modified max-min fair allocation algorithm to re-align the bandwidth allocation of all ingress aggregates. The allocation effect is the same as “equal allocation” because all the traffic aggregates share all the congested links.

5.6.3.2 Responsiveness to Network Dynamics

We use a combination of CBR and FTP sources to study the joint effect of our dynamic core provisioning algorithm (i.e., the P-M Inverse method for rate reduction and max-min fair for rate alignment) and our node provisioning algorithm. Periodic edge rate alignment is invoked every 60s. We use CBR and FTP sources for EF

⁴We note that it does not make sense to plot the performance metrics shown in Figure 5-15 in the same time sequence style as of Figure 5-16. The reason is that in a time-sequenced test, after the first test case, the load conditions prior to each rate reduction could become different for different allocation methods, and the results from the comparison metrics would not be comparable.

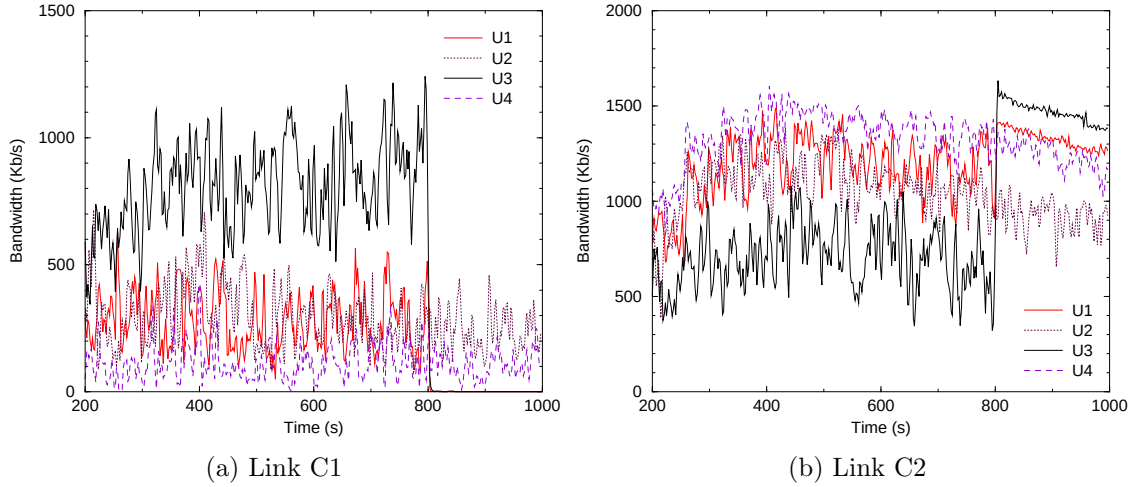


Figure 5-17: Average Bandwidth Allocation for AF1 Aggregates

and AF1 traffic aggregates, respectively. Each traffic class comprises four traffic aggregates entering the network in the same manner as shown in Figure 5-7. A large number (50) of FTP sessions are used in each AF1 aggregate to simulate a continuously bursty traffic demand. The distribution of the AF1 traffic across the network is the same as shown in Table 5.1.

The number of CBR flows in each aggregate varies to simulate the effect of varying bandwidth availability for the AF1 class (which could be caused in reality by changes in traffic load, route, and/or network topology). The changes in available bandwidth for AF1 class includes: at time 400s into the trace, $C2$ (the available bandwidth at link 2) is reduced to 2Mb/s; at 500s into the trace, $C3$ is reduced to 0.5 Mb/s; and at 700s into the trace, $C3$ is increased to 3 Mb/s. In addition, at time 800s into the trace, we simulate the effect of a route change, specifically, all packets from traffic aggregate $u1$ and $u3$ to node 5 are rerouted to node 8, while the routing for $u2$ and $u4$ remains intact.

Figure 5-17 and 5-18 illustrate the allocation and delay results for the four AF1 aggregates. We observe that not every injected change of bandwidth availability

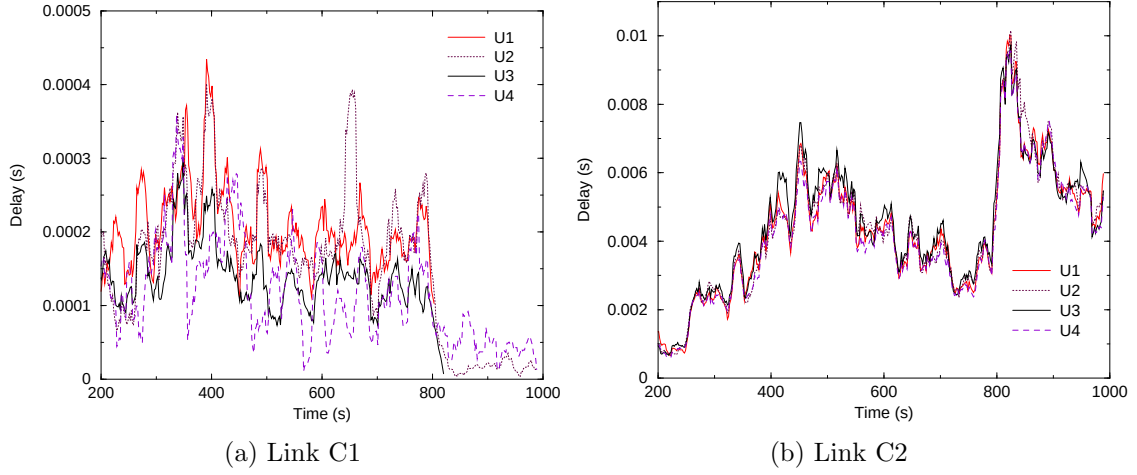


Figure 5-18: Delay for AF1 Aggregates (Averaged over 10s)

triggers an edge rate reduction, however, in such a case it does cause changes in packet delay. Since the measured delay is within the performance bound, the node provisioning algorithm does not generate *Congestion_Alarm* signals to the core provisioning module. Hence rate reduction is not invoked. In most cases, edge rate alignment does not take effect either because the node provisioning algorithm does not report the needs for an edge rate increase. Both phenomena demonstrate the robustness of our control system.

The system correctly responds to route changes because the core provisioning algorithm continuously measures the traffic load matrix. As shown in Figure 5-17, after time 800s into the trace, the allocation of $u1$ and $u3$ at link C1 drops to zero, while the corresponding allocation at link C2 increases to accommodate the surging traffic demand.

5.7. Summary

This chapter makes two main contributions. First, our node provisioning algorithm prevents transient service level violations by dynamically adjusting the service

weights of a weighted fair queueing scheduler. The algorithm is measurement-based and effectively uses the multi-class virtual queue technique to predict the onset of SLA violations. Second, our core provisioning algorithm is designed to address the difficulty of provisioning DiffServ traffic aggregates (i.e., rate-control can be exerted only on the basis of traffic aggregates at the root of their traffic distribution tree). We proposed the P-M Matrix Inverse method for edge rate reduction which balances the trade-off between fairness and minimizing the branch-penalty. We extended max-min fair allocation for edge rate alignment and demonstrated its convergence property.

Collectively, these algorithms contribute toward a more quantitative differentiated service Internet, supporting per-class delay guarantees with differentiated loss bounds across core IP networks. We have argued that such a dynamic provisioning is superior to static provisioning for DiffServ because it affords network mechanisms the flexibility to regulate edge traffic maintaining service differentiation under persistent congestion and device failure conditions when observed in the core network.

The complexity of our algorithms mainly resides in the node provisioning algorithm, which is distributed to each core router and is scalable to a large network. The challenge of implementing the centralized core provisioning algorithm, however, lies in the continuous monitoring of traffic matrix across the core network. To improve scalability, one can enlarge the monitoring granularity and time-scale, for example, focusing on a few potential bottleneck links instead of every internal link of a network; or increasing the provisioning time-scale *update_interval*. In addition, the recent work on network measurement [36, 30, 29] using the AT&T backbone network provides valuable insights on how to scale the monitoring process up to handle large networks.

In the preceding chapters we have focused on the design of efficient bandwidth

allocation algorithms for edge and core based IP networks. The correct operation of these network mechanisms, however, require end-users' cooperation (e.g., in choosing the appropriate service class, or truthfully declare their types of utility functions). In general, these cooperation requirements are in conflict with end-users' self-optimizing goals unless the network optimization goal is incentive compatible with end-users' selfish goals. In the next chapter, we address this problem by creating incentives for differentiated service mechanisms in edge-based wireless networks.

Chapter 6

Incentive Engineering for Service Differentiation in Wireless Access Networks

6.1. Introduction

The emergence of public wireless extensions to the Internet predominantly build around edge-based IEEE 802.11 Wireless Local Area Networks (WLANs) [50] highlights the lingering problem of how to price wireless data. Market evidence has shown that the prevailing charging model for wireless access service is block-rate charging, which comprises a fixed charge for usage within a block of air time or bytes delivered, and a higher flat rate for any usage that exceeds the block amount. This type of charge is preferred by users because of price stability and predictability, and by service providers because of the simplicity of the design of the billing infrastructure [81, 38, 5]. However, this type of charging model is not sensitive to the difference between stable allocation for real-time applications like streaming video and best effort allocation for bursty data applications like web transactions. Therefore, without an incentive structure a stable allocation service could be easily “overrun” by non real-time sensitive data applications. Under such conditions lower priority packets take advantage of service differentiation by transiting their packets

using the higher priority service class. This practice leads to the “tragedy of the commons” phenomenon [46]. In this chapter we address this problem by proposing incentive engineering techniques for edge-based wireless networks that create incentives for mobile users to truthfully self-differentiate their service needs based on their application needs.

The current engineering approach taken by cellular networks introduces a tightly controlled (e.g., circuit-based) environment for both wireless voice and data. However, this approach does not scale well given the increasing diversity of applications and device programmability emerging. In contrast, edge-based WLANs inherit both the simplicity and the best-effort service model of the Internet. However, in order to deliver better than best-effort services (e.g., IEEE 802.11e) in a WLAN-based access networks, there is a need for rate regulation techniques [8]. These techniques include traffic shaping at both mobile devices and access points, with the addition of admission control at network access points to enforce service differentiation and fend off any potentially abusive usage such as bandwidth hogging or denial of service attacks. However, access rate regulation is very challenging to get right. Static rate regulation mechanisms are too simple to efficiently control bursty transactional applications such as web browsing, and measurement-based schemes can potentially generate large amounts of control messaging. Additionally, bandwidth reservation mechanisms involve a difficult trade-off between guaranteeing the full length of bandwidth reservation and inhibiting excessive bandwidth hogging. Hard reservation guarantees bear the complexity of admission control when multi-tiered service quality is required. This requires applications to declare the session length in advance, which none of the widely deployed applications can easily provide.

The contribution of this chapter is as follows. We address the lack of incentives and rate regulation challenges discussed above and propose a set of market-based

mechanisms to address these problems. Specifically, we introduce a service control parameter called *service purchasing power*, which plays the same role as a service budget but covers the internal price of resource usage. By defining service purchasing power as a non-accumulated and non-replenished budget, we create incentives for mobile users to self-differentiate based on their application needs, and hence, reduce the need for per-mobile rate-control messaging. Rather, a *price-service menu* is periodically broadcast by base stations to direct the rate adjustment of all mobile devices in a particular cell. The price-service menu comprises two incentive-based service classes: an *instantaneous allocation (IA)* class, which provides better throughput, and a *stable allocation (SA)* class, which provides better allocation stability. The IA and SA service classes trade off the average amount of allocated bandwidth with allocation stability; that is, a price-service menu provides a ranking of service classes with decreasing bandwidth allocation stability and per-unit internal price, but with increasing prospects for higher average bandwidth allocation. As a result, data applications can opt for the IA service class, which will on average, offer better bandwidth allocation to sessions, but at the cost of more instability in the offered bandwidth. To offset this instability in allocated bandwidth, real-time applications seeking better service quality can pay a premium to use the SA service class, which provides stable bandwidth allocation but usually results in smaller amounts of allocated bandwidth to a session in comparison to the IA service. There lies the inherent trade-off in the offered services between the two classes.

The rate control for IA service is measurement-based to efficiently regulate bursty transactional applications. The enforcement algorithm only resides inside the networks and does not require users to estimate their own bandwidth demands. We approach the bandwidth reservation problem with a “soft” guarantee on the length of SA service reservation. The “softness” of the guarantee follows the rank of the

reservation’s internal bid price, which is proportional to the SA service purchasing power. Users with higher SA service purchasing power will less likely see early termination of their reservations. As a result, the SA class does not require users to predict their session lifetime. To make the scheme more usable, a “warning interval” is implemented for sessions in danger of early termination to renegotiate. This feature practically mirrors application operating conditions because typically, applications cannot predict session bandwidth demands nor the session duration in advance, as discussed above.

The structure of the chapter is as follows. In Section 6.2. we present the economics background for incentive engineering and related work. This is followed in Section 6.3. by an overview of our model including the allocation price-service menu and messaging protocol. We discuss the incentive-based control algorithms for IA and SA classes in Section 6.4. and Section 6.5., respectively. In Section 6.6., we present the properties of our incentive engineering scheme in the context of the mobile user dominant strategy. We demonstrate that the best strategy for a user is to cooperate with network traffic control. In Section 6.7., we evaluate our algorithms in an experimental wireless testbed that also supports an emulation capability, which further helps evaluate the system under different conditions and scenarios. Finally, we present some concluding remarks in Section 6.8.

6.2. Economics Background and Related Work

Incentive engineering stems from the discipline of mechanism design in economics theory, which structures the strategy space of users such that a user’s self-optimizing choice of action is “incentive compatible” with the system optimization goal. One typical example is the Clarke-Groves mechanism [43] that charges a user the amount of payoff displaced from all the other users due to the allocation to the user, (i.e.,

the shadow price of this allocation on all other users). An example of the Clarke-Groves mechanism is the Vickrey's second-price auction [103], which charges winning bidders the highest losing bid. The salient property of the Clarke-Groves pricing mechanism is incentive-compatibility, namely truth-telling is a dominant strategy for users. Our work is inspired by the seminal work of Drexler and Miller [28] on mechanism design for operating systems for the dual purpose of inducing cooperative behavior over computational resources and reducing market transaction cost.

Our work can be viewed as a continuation of the argument advocated in [94]; that is, monetary charge for network service is better based on system level architectural issues rather than "economically optimal" marginal cost. We argue that to provide stable and predictable service charges, fast time-scale market-based traffic control mechanisms should be decoupled from monetary charge. When pricing is non-monetary, existing congestion pricing mechanisms [74, 58, 72] are not applicable because non-cooperative users have no incentive to truthfully respond to a non-monetary "price signal" offered by the network control system. Consequently, rather than maximizing the difference between the utility and cost functions, users will solely maximize their utility functions and ignore the cost functions as long as the non-monetary cost is below the non-monetary budget. Our incentive engineering design turns this non-cooperative game into an equivalence of a Nash bargaining solution [109, 78], whose operating point has better properties, (i.e., Pareto optimum and Nash bargaining fair), than the Nash equilibrium operating point for a non-cooperative congestion pricing market. We can achieve this goal because our engineering design effectively limits user strategy space, such that, the dominant strategy coincides with the Nash bargaining solution. In addition, the implementation of our incentive engineering mechanisms is more efficient than maintaining a non-cooperative market of multi-round auction or tatonnement process, where

renegotiation delay and convergence problem are common to users playing extensive strategies in a repeated-game setting [43].

There are two ways to design service models that promote self-differentiation among users: differentiated pricing or service classes tailored toward specific user groups. The Paris Metro Price [80] is an example of using two-tier pricing to realize differentiated services without any additional network mechanisms. In contrast, the Alternative Best Effort service proposed in [49] is a good example of designing two alternative service classes, each of which is preferred by data and multimedia users, respectively. Our service model employs both methods. We design differentiated pricing to regulate demand for stable allocation, and differentiated service classes by considering the trade-off between allocation stability and allocation quantity. Unlike [49] that leverages the trade-off between packet loss and throughput but requires modification of packet schedulers, our service differentiation is at the session level and is therefore independent of any packet scheduler.

The idea of pricing allocation stability is similar in spirit to the priority service pricing scheme for rationing supplies [108], which has been used as a basis for electric power distribution systems. Wilson proposes this scheme for industries where spot pricing is not efficiently deployable, due to pervasive transaction cost and technical limitations. In [26] priority pricing is used to price best-effort multi-QOS network services. The difference again is that in [108, 26] optimal prices are calculated assuming that users will maximize their payoff function, while in our case, users would rather use up their budget to maximize their utility functions because the budget is non-monetary and non-accumulative.

Budget control has been a largely overlooked problem, which is caused by an inherent time-scale mismatch; that is, monthly-based block-rate usage budget (in minutes) and usage accounting are not sensitive to bursts of usage at session-level

time-scales. Consequently such a charging model does not offer incentives for users to cooperate during periods of network congestion. The only work similar to our non-accumulated and non-replenished budget is the User Share Differentiation [105] proposed for differentiated service. Our proposal of service purchasing power is not only a parameter for relative service differentiation, as is the case of [105], but also a budget in the Nash bargaining solution driving market-based bandwidth allocation mechanisms at session-level time-scales.

There have been a number engineering proposals related to the design of practical market mechanisms for network traffic control. In [42] two separate markets are used, one for the spot bandwidth and the other for the reserved bandwidth. However, the pricing mechanisms for both markets are based on the demand-supply tatonnement process without consideration of the opportunity cost of bandwidth reservation over the spot market bandwidth. In [38, 104], engineering efforts are used to model opportunity cost for differentiated service classes. The proposed service charge involves congestion, time and volume based pricing components, each of which requires parameter tuning. This heuristic approach bears large complexity for multiple service types.

The exact calculation of the opportunity cost for bandwidth reservations is best represented by [91] in the form of a derivative pricing instrument over the bandwidth spot market. However, these schemes are not practical in support of wireless services because the access network bandwidth that is traded has a minuscule valuation over the fast congestion-control time-scale. For example, the widely used Black-Scholes [13] formula for calculating option premium relies on a reference risk-free investment instrument, (i.e., the interest rate income). In the case of traffic control the equivalent risk-free alternative cannot be interest rate income because the value

would be too small for any users to care about¹.

In mobile and wireless networks, market mechanisms have been applied in a very limited manner. In [11], a revenue framework is proposed to resolve some of the adaptation policy trade-offs. The scheme provides incentives for adaptation by charging sessions that benefit from the adaptation, while compensating sessions that suffer from adaptation. However, the exact calculation of credits and charges are challenging and are not formulated in [11]. In [54], the authors analyze the property of the Paris Metro Pricing [80] scheme within the context of wireless access service. The service offering is limited with no support for allocation stability. Because price is non-monetary in our scheme, we use conventional measurement-based traffic prediction and handoff admission control to assure handoff performance. There has been a large body of work in the literature on handoff admission control. For a recent survey and performance comparison, see [24].

6.3. Incentive Engineering Model for WLAN Access Networks Overview

6.3.1 Network Model

Figure 6.3.1 illustrates a wireless access network architecture in the context of IEEE 802.11b WLAN networks. Note, that the particular cellular network topology shown is for illustration only, and not essential to our framework. We use the terms “mobile device” and “access point” in a generic sense. At access points, per-mobile and per-class traffic regulators are used to regulate downlink traffic. In addition, each mobile device optionally uses per-class traffic regulators in the form of policers

¹For example, a 5% annual interest return on a few dollars worth of wireless access bandwidth over a session lifetime of one hour would result in a credit on the order of 10^{-6} of a dollar!

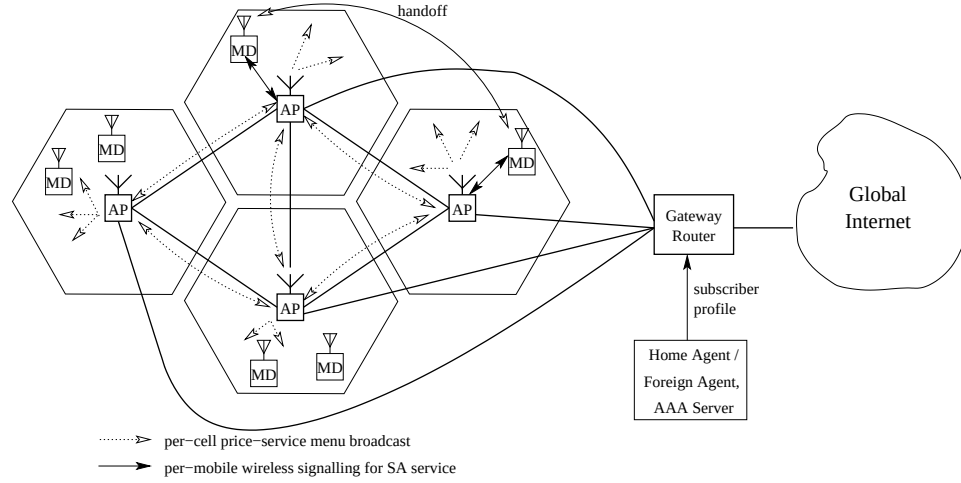


Figure 6-1: Wireless LAN Based Mobile Access Network

or shapers to self-regulate uplink traffic. User profiles containing service specific resource allocation policies are stored at the Authentication, Authorization and Accounting (AAA) server at the mobile device's home network, and delivered to a visiting network by a mobility management protocol. We assume that there is a broadcast channel at the media access control (MAC) layer from the access point to all the mobile devices in a particular cell. Our incentive engineering mechanisms are applied at the session bandwidth allocation level, involving traffic regulator modules at mobile devices and access points. Fast time-scale packet scheduling algorithms are not affected.

6.3.2 Service Purchasing Power

Typically, service budget (e.g., the number of “free” minutes within a service plan) is not allowed to be accumulated, otherwise, idle users could carry-over large amounts of unused budget, distorting the market mechanism. Therefore, users have no incentive to conserve budget toward the end of the budget replenishing cycle, and could subsequently start a spending spree, distorting the market mechanism as well.

To address this problem we introduce a parameter called *service purchasing power*, which plays the same role as a service budget but covers the internal price of resource usage. By defining service purchasing power as a non-accumulated and non-replenished budget, we avoid the difficulty of budget control. With the service purchasing power known to the network, a user's strategy space is essentially constrained by the price-service menu, which induces user cooperation, avoiding over allocation of bandwidth, and enforcing the truthful declaration of reservation bandwidth. Allowing users to choose between service classes in this manner helps promote self-differentiation among user applications, enabling differentiated resource allocation.

Each user (i.e., mobile device) is assigned a service purchasing power ϑ_i , which plays the role of a "constant budget". The value of ϑ_i for each user is determined by the network. For example, a premium user may be given a large ϑ_i to afford relatively high priced bandwidth. ϑ_i is part of the user's profile stored at the AAA server. Alternatively, ϑ_i could be stored at the mobile device in an encrypted format together with the mobile device ID, and then passed to the wireless access network during registration and handoff operations. Each mobile device partitions its service purchasing power as it wishes into portions for IA and SA allocation, denoted as $\vartheta_{i,I}$ and $\vartheta_{i,S}$, respectively. Since access points are aware of ϑ_i and the corresponding allotment to SA reservation $\vartheta_{i,S}$ through a mobile device initiated reservation request, the IA portion of service purchasing power can be derived as,

$$\vartheta_{i,I} = \vartheta_i - \vartheta_{i,S}. \quad (6.1)$$

6.3.3 Price-Service Menu

Our incentive engineering mechanisms use a market-based price to distribute allocation information and regulate bandwidth usage. Each access point l periodically broadcasts a non-monetary price-service menu within its cell driven by price change. The price-service menu comprises the price of the IA and SA classes, $p_{l,I}$ and $p_{l,S}$ respectively, as well as $p_{l,H}$, the price for a subclass of SA called *handoff allocation (HA)*. The HA class enforces price differentiation that any mobile devices with $\vartheta_{i,S} < p_{l,H}$ will be denied handoff for lack of service purchasing power. We represent the price-service menu of access point l in a price vector: $\langle p_{l,I}, p_{l,S}, p_{l,H} \rangle$. The SA class requires a per-mobile reservation message between a mobile device and its access point as shown in Figure 6.3.1. An SA reservation has three parameters: the requested bandwidth quantities b_{i,S_U} b_{i,S_D} for uplink and downlink respectively, and the allotted SA portion of service purchasing power $\vartheta_{i,S}$. The handoff price $p_{l,H}(t)$ is derived as follows:

$$p_{l,H}(t) = \max \left\{ p_{l,S}(t), \max_{k \in \mathcal{A}(l)} \{p_{k,S}(t)\} \right\}, \quad (6.2)$$

where $\mathcal{A}(l)$ denotes the set of adjacent access points of l . This is equivalent to stating that a mobile device's service purchasing power $\vartheta_{i,S} \geq p_{l \ni i, H}$ will be high enough to acquire bandwidth at neighboring cells. Here with an abuse of notation, we use $l \ni i$ to denote the cell l where the mobile device i is active. We note that with mobility prediction, the size of $\mathcal{A}_i(l)$ for each mobile device i can be reduced. For example, in a mobile-initiated handoff where a mobile device notifies network of its future access point, $\mathcal{A}_i(l)$ contains only one access point.

The extent of handoff guarantees depends on the specific handoff admission control algorithms employed. In principle, any of the algorithms given in [24] can

be used with our scheme. These handoff algorithms differ in how handoff is executed, and in the number of neighboring cells that participate in the advanced reservation required for assured handoff. The performance guarantee during handoff is usually in the form of an upper bound on the handoff dropping probability. In the remaining part of this chapter, we will focus on the calculation of internal price for IA and SA services.

6.3.4 IA and SA Algorithms

The IA and SA services are designed not to offer hard guarantees of QOS, but to provide differentiated stability and instantaneous bandwidth allocation at session level. Each access point keeps the profiles of all users and uses the IA and SA algorithms to calculate the price-service menu dynamically based on the profile records and measured bandwidth usage.

The IA algorithm supports transactional data sessions, whose demand has to be measured and predicted rather than declared in advance by applications. The algorithm extends beyond simple price-demand calculation with a price calculation based on traffic load measurement. The load is with respect to the actual bandwidth usage, instead of reservation amount. In addition, to support our design goal of avoiding software installation in mobile devices, the IA algorithm needs to address the challenge of downlink traffic control with incomplete information due to the absence of mobile device participation in the control algorithm. This problem is solved by the optimistic rate allocation. Both of the IA algorithm features (i.e., measurement-based price calculation and optimistic rate allocation) are presented in Section 6.4..

The SA algorithm needs to reduce the early-termination probability for SA sessions. This is performed through the admission control algorithm, which calculates

the market price for the SA service. The algorithm is complicated by our usability goal of giving each SA session at least a warning interval amount of time before revoking its bandwidth reservation. The reason is that such a feature can potentially cause an arbitrage situation between IA and SA sessions, in which an IA session may benefit by switching to SA service. The arbitrage is removed with the IA allocation pegging algorithm presented in Section 6.5..

6.4. Incentive Engineering for IA Class

6.4.1 Baseline IA Algorithm

The baseline price calculation for the IA class is based on the aggregated price-demand function, where the IA price $p_{l,I}$ is interpreted as a common allocation signal for users with different service purchasing power.

Since service purchasing power is a non-accumulated and non-replenished budget, a mobile device i has no benefit in conserving its IA portion of the service purchasing power $\vartheta_{i,I}$. That is, with a given $\vartheta_{i,I}$, the best strategy for the mobile device i is to declare its IA bandwidth demand that uses up the service purchasing power $\vartheta_{i,I}$. Therefore, the IA price-demand function of each mobile device i is:

$$b_{i,I} = \min \left\{ \vartheta_{i,I} / p_{l \ni i, I}(t) , b_{i,I}^{\max} \right\}. \quad (6.3)$$

Here $b_{i,I}^{\max}$ is the maximum bandwidth of the IA class (e.g., the wireless channel capacity) that mobile device i may consume.

Summing up both sides of (6.3) for all the users i , we have the aggregated price-demand function,

$$q_{l,I} = \sum_{i \in l} b_{i,I} = \sum_{i \in l} \min \left\{ \vartheta_{i,I} / p_{l,I}(t) , b_{i,I}^{\max} \right\}, \forall l \quad (6.4)$$

where $q_{l,I}$ denotes the total available bandwidth for the IA class in cell l .

When all the $b_{i,I}^{\max}$ are set to channel capacity, the IA price can be simply derived from Equation (6.4) as:

$$p_{l,I} = \left(\sum_{i \in l} \vartheta_{i,I} \right) / q_{l,I}, \quad (6.5)$$

The allocation procedure follows two steps: access points use Equation (6.4) to update the IA price and broadcast price-service menu; mobile devices then use Equation (6.3) to derive their IA allocations.

6.4.2 Measurement-Based Price Calculation

The resulting allocation, however, could be largely under utilized by short-lived IA applications, such as, web transactions. A straightforward solution to this problem is to dynamically adjust $b_{i,I}^{\max}$ based on the measured bandwidth usage of each mobile device. Since $b_{i,I}^{\max}$ plays the role of limiting bandwidth allocation to a mobile device, by reducing $b_{i,I}^{\max}$ for a mobile device with light usage we can distribute the reduced bandwidth amount to heavy users, and hence, increase the overall bandwidth utilization. For clarity, we denote the potentially time-varying $b_{i,I}^{\max}$ as $b_{i,I}^{\max}(t)$. Therefore, Equation 6.3 is changed to

$$b_{i,I}(t) = \min \left\{ \vartheta_{i,I} / p_{l \ni i,I}(t), b_{i,I}^{\max}(t) \right\}. \quad (6.6)$$

To handle the boundary condition caused by varying $b_{i,I}^{\max}(t)$, we introduce $\zeta_{i,I}(t)$, the IA unit “bid price” for a mobile device i as

$$\zeta_{i,I}(t) = \vartheta_{i,I} / b_{i,I}^{\max}(t). \quad (6.7)$$

We sort $\zeta_{i,I}$ in descending order, and denote the k th highest IA unit bid price as

$\zeta_{(k),I}$. In addition, we denote $\mathcal{B}(k)$ the subset of users whose $\zeta_{i,I}$ are among the top k . Subsequently, Equation (6.4) can be formatted as:

$$q_{l,I} = \frac{\sum_{i \in \bar{\mathcal{B}}(k)} \vartheta_{i,I}}{p_{l,I}} + \sum_{i \in \mathcal{B}(k)} b_{i,I}^{\max}(t), \quad (6.8)$$

where $\zeta_{(k+1),I} \leq p_{l,I} < \zeta_{(k),I}$, $k = 0, 1, \dots, N-1$,

and N is the total number of user. In addition, $\zeta_{(0),I} \triangleq \infty$.

By inverting this equation, we have the following formula for calculating the IA price:

$$p_{l,I} = \frac{\Theta_{all,I} - \Theta_{k,I}}{q_{l,I} - \mathbf{b}_{k,I}}, \quad \mathbf{q}_{k,I} < q_{l,I} \leq \mathbf{q}_{k+1,I}, \quad (6.9)$$

where the partial sums are defined as follows:

$$\Theta_{k,I} \triangleq \sum_{i \in \mathcal{B}(k)} \vartheta_{i,I}, \quad \Theta_{all,I} \triangleq \sum_i \vartheta_{i,I}, \quad (6.10)$$

$$\mathbf{b}_{k,I} \triangleq \sum_{i \in \mathcal{B}(k)} b_{i,I}^{\max}(t), \quad (6.11)$$

$$\mathbf{q}_{k,I} \triangleq \mathbf{b}_{k,I} + \frac{\Theta_{all,I} - \Theta_{k,I}}{\zeta_{(k),I}}. \quad (6.12)$$

Therefore, the additional work for access points to calculate the IA price is to maintain a sorted partial sums based on the IA bid price, and search for the bandwidth range $(\mathbf{q}_{k,I} \quad \mathbf{q}_{k+1,I}]$ within which the available IA bandwidth $q_{l,I}$ fits.

The aggregated price function in (6.8) has a piecewise $1/q$ form. Figure 6-2 illustrates one example with three users, whose $(\vartheta_{i,I}, b_{i,I}^{\max})$ pair are (30, 3), (50, 10), and (10, 5), respectively, where the bandwidth unit is 100 Kb/s. The corresponding $\zeta_{i,I} = \frac{\vartheta_{i,I}}{b_{i,I}^{\max}}$ are 10, 5 and 2, sorted in descending order. These values lead to the first order break points in price, as shown in the figure. Consequently, the aggregated

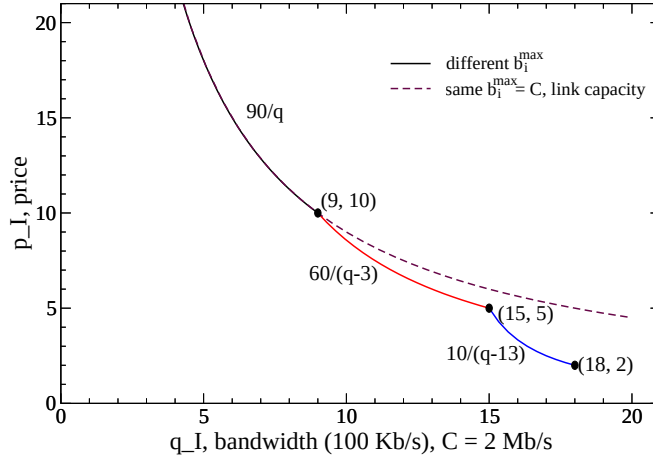


Figure 6-2: Example of Aggregated IA Price Function

price function is a cascade of three functions: $\frac{90}{q}$, $\frac{60}{q-3}$, and $\frac{10}{q-13}$, respectively. In addition, we also show a curve (the dotted curve in Figure 6-2) where all $b_{i,I}^{\max}$ are the same, equal to the channel capacity of 2 Mb/s. This example illustrates the effect of adjusting $b_{i,I}^{\max}(t)$ on the price function. The dotted curve is always above the solid curve, which results from the measurement-based adjustment on $b_{i,I}^{\max}$, (i.e., $b_{i,I}^{\max}(t) \leq b_{i,I}^{\max}$). The smaller IA price under $b_{i,I}^{\max}(t)$ leads to higher bandwidth allocation for active mobile devices and a higher overall bandwidth utilization.

6.4.3 Optimistic Rate Allocation with Incomplete Information

The measurement-based IA price calculation, however, requires a per-mobile messaging protocol to notify mobile devices and their corresponding access points of the changed $b_{i,I}^{\max}(t)$. Such an implementation would defeat our design goal of using only a single broadcast message for IA bandwidth allocation. In what follows, we present an enhancement to the IA pricing algorithm that tolerates incomplete information resulting from the reduction of messaging.

With the absence of per-mobile messaging to notify the change of $b_{i,I}^{\max}(t)$, each

mobile device simply uses the constant value $b_{i,I}^{\max}$ to derive its allocation rate from the broadcasted price $p_{l,I}$, which is

$$b_{i,I}^* = \min \left\{ \vartheta_{i,I}/p_{l,I}^*, b_{i,I}^{\max} \right\}. \quad (6.13)$$

Here we use the notation $b_{i,I}^*$ to represent the bandwidth allocation derived by a mobile device, and to differ it from the ideal bandwidth allocation $b_{i,I}$ defined in (6.6). Since $b_{i,I}^{\max}(t) \leq b_{i,I}^{\max}$, we have $b_{i,I}^* \geq b_{i,I}$. In the worst case, when every mobile device uses up its entire allocation $b_{i,I}^*$, the wireless link will be overloaded by a ratio of $p_{l,I}/p_{l,I}^*$, where $p_{l,I}^*$ is calculated from Equation (6.9) replacing $b_{i,I}^{\max}(t)$ with $b_{i,I}^{\max}$.

The rate allocation algorithm tolerates this discrepancy at mobile devices due to incomplete information. It “optimistically” controls the extent of over-allocation by measuring the actual system load and adjusting $b_{i,I}^{\max}(t)$ adaptively. The measurement algorithm operates over discrete time t_n slotted by τ , the same measurement window used for demand measurements. τ is limited by the response time of the control system to change the regulator shaping rate. In Section 6.7.2, we will measure the minimum value of τ sustainable in an experimental wireless testbed. The algorithm measures the up and downlink average rates $\bar{b}_{i,I}^{up}(t_n)$ and $\bar{b}_{i,I}^{down}(t_n)$, respectively, over the interval $(t_{n-1} \ t_n]$.

The value of $b_{i,I}^{\max}(t_n)$ is calculated according to the measured average rate $\bar{b}_{i,I}(t_n) = \bar{b}_{i,I}^{up}(t_n) + \bar{b}_{i,I}^{down}(t_n)$ in the past τ interval:

$$b_{i,I}^{\max}(t_n) = \min \{ \gamma \bar{b}_{i,I}(t_n), b_{i,I}^{\max} \}. \quad (6.14)$$

Here $\gamma \geq 1$ controls the extent of over-allocation. When $\gamma = 1$, $b_{i,I}^{\max}(t)$ is calculated based on the average rate, which leads to the maximum extent of over-allocation. When $\gamma \gg 1$, $b_{i,I}^{\max}(t) = b_{i,I}^{\max}$, namely the adjustment is disabled and no over-

allocation is allowed.

The IA traffic load measurement is calculated with respect to the actual usage (not the amount of reservations) of the SA traffic, that is:

$$\rho_{l,I} = \frac{\sum_i \bar{b}_{i,I}(t_n)}{\mathcal{C}(1 - \rho_{l,S}(t_n))}, \quad (6.15)$$

where $\mathcal{C}$ is the channel capacity, and $\rho_{l,S}(t_n)$ is SA traffic load. The value of γ is adjusted based on the system load condition ρ as follows:

$$\gamma(t_n) = \begin{cases} \min\{2\gamma(t_{n-1}), \gamma_{\max}\} & \rho > threshold \\ \max\{1, \gamma(t_{n-1})(1 - dec)\} & \rho < \kappa * threshold; \\ \gamma(t_{n-1}) & otherwise. \end{cases} \quad (6.16)$$

where $\gamma_{\max} \triangleq \max\{b_{i,I}^{\max}/b_{i,I}^{\max}(t) \mid b_{i,I}^{\max}(t) > 0\}$. Here $\gamma_{\max}$ caps the value of γ because increasing γ beyond $\gamma_{\max}$ has no effect when all the mobile devices have $b_{j,I}^{\max}(t)$ reaching the absolute maximum $b_{j,I}^{\max}$.

The goal of Equation 6.16 is to keep ρ within a range (i.e., between κ and 100%) of the threshold load that triggers excessive delay, as discussed in [8]. In this chapter, we set the *threshold* value to 90%. When ρ exceeds the threshold load, γ is doubled over every τ interval to quickly reduce the extent of over-allocation. When ρ falls below $\kappa * threshold$, γ is reduced by a factor of *dec* until reaching 1. The purpose of this is to increase the extent of over-allocation (i.e., broadcasting a smaller value of $p_{l,I}^*$), encouraging bandwidth usage. In Section 6.7.2, we will experiment with the setting of the parameters κ and *dec*. As a safe guard against frequent variations of $p_{l,I}^*$, we introduce a control parameter $\delta = 5\%$ such that $p_{l,I}^*(new)$ is only broadcast when the change is larger than δ percentage, (i.e., $|1 - p_{l,I}^*(old)/p_{l,I}^*(new)| > \delta$).

The rate enforcement procedure also executes only at the access points and

without the participation of mobile devices. In WLAN environments, this means traffic regulation is performed only on the downlink. Since mobile devices are only aware of $b_{i,I}^*$, the traffic regulation algorithm uses $b_{i,I}^*$ rather than $b_{i,I}$ as the rate limit. The peak rate of downlink shaping is set as follows:

$$b_{i,I}^{down}(t_n) = \begin{cases} b_{i,I}^* - \bar{b}_{i,I}^{up}(t_n) & \bar{b}_{i,I}^{up}(t_n) \leq b_{i,I}^*; \\ 0 & \text{otherwise, (deny any} \\ & \text{downlink access).} \end{cases} \quad (6.17)$$

This approach is effective enough to control web browsing because of the asymmetry of web traffic with most traffic over the downlink. For heavy uplink users, enforcement is indirectly performed by stopping downlink traffic. If this approach still fails, the corresponding mobile device will be treated as non-compliant, and any future access to network services can be denied. Note, that this downlink-only shaping approach is also practical for the SA reservations, as discussed in the next section.

The IA bandwidth allocation and traffic regulation algorithms are designed for heavy users. For light users, the traffic regulation algorithm regulates them together as a traffic aggregate as if they are from a single user (denoted by the *pseudo user* #0) to allow free flow of sporadic control traffic. The measurement algorithm detects the change of usage from light to heavy by always tracking the top few heavily used sessions of user #0. Reversely, the measurement algorithm detects an idle usage state when $b_{i,I}^{max}(t_n)$ is consistently below a threshold for a time-out interval. In this case, all the sessions of user i will be bundled into the traffic aggregate of user #0 and the measurement states for user i will be removed.

Remark: We note that even though the proposed traffic enforcement does not require mobile devices' participation, the IA price broadcast is necessary. Coop-

erative mobile devices can avoid throttling their downlink traffic by flow-control remote sender, or sending uplink traffic within their rate limits. In addition, the IA price broadcast is the minimum amount of signal required to notify all the mobile devices of their corresponding allocations. This reduction on signaling also contributes to power saving because only those mobile devices that are actively transmitting/receiving need to listen to the price broadcast.

The baseline IA allocation algorithm is shown in Figure 6-3. We can observe the simplicity of the mobile device algorithm, due to the price-service menu broadcast protocol.

```

calculate_IA_price { // Access Point Strategy
  maintain per mobile attributes:  $b_{i,I}^{\max}$ ,  $\vartheta_{i,I}$  and the sum  $\Theta_{all,I}$ ;
  maintain sorted list of  $\zeta_{(k),I}$  and the associated partial sums:  $\mathbf{b}_{k,I}$ ,  $\mathbf{q}_{k,I}$  and  $\Theta_{k,I}$ ;
  update  $q_{l,I} = \rho_{l,I} \mathcal{C}$ ; // IA class load
  locate  $j$  s.t. available bandwidth  $q_{l,I} \in (\mathbf{q}_{j,I}, \mathbf{q}_{j+1,I}]$ ;
  calculate price  $p_{l,I} = (\Theta_{all,I} - \Theta_{j,I}) / (q_{l,I} - \mathbf{b}_{j,I})$ ;
  broadcast price-service menu;
}
calculate_IA_allocation { // Mobile Device Strategy
  retrieve  $p_{l,I}$  from the broadcasted price menu;
  calculate allocation  $b_{i,I} = \min\{\vartheta_{i,I}/p_{l,I}, b_{i,I}^{\max}\}$ ;
}

```

Figure 6-3: Baseline IA Allocation Algorithm

6.5. Incentive Engineering for SA Class

6.5.1 Baseline SA Algorithm

The SA reservation message comprises the triplet of uplink and downlink bandwidth quantities, and the service purchasing power $(b_{i,S_U}, b_{i,S_D}, \vartheta_{i,S})$. Unlike the IA class, an SA bandwidth request needs to pass admission control based on resource availability and mobility prediction, which is extended from the conventional hand-off admission control algorithms found in the literature [24]. Once admitted, the

allocation is guaranteed as long as the corresponding SA unit bid price satisfies $\zeta_{i,S} \geq p_{l \ni i,S}(t)$, where,

$$\zeta_{i,S} \triangleq \frac{\vartheta_{i,S}}{b_{i,S_U} + b_{i,S_D}}, \quad (6.18)$$

and $p_{l \ni i,S}(t)$ denotes the non-monetary price of SA class in cell l where the mobile device i is active.

When $\zeta_{i,S} < p_{l \ni i,S}(t)$, mobile device i 's SA allocation is considered to be *under probation*. In this case, the allocation guarantee is revoked only when the SA allocation has been continuously under probation for an interval of T_S . Therefore, T_S is the minimum interval over which an SA allocation is guaranteed, and applications have at least T_S amount of time for rate-adaptation or renegotiation.

Figure 6-4 shows the price-demand functions for both IA and SA classes, with the IA price-demand function shown from left to right and the SA price-demand function from right to left. The intersection of these two functions gives the spot IA price² $p_{l,I}$. The figure also shows the decreasing allocation reliability for the SA class, following the descending bidding price. When $p_{l,S}$ increases, the SA sessions whose unit bid price $\zeta_{i,S} < p_{l,S}$ (e.g., the price block falls below the price line $p_{l,S}$ in the figure) will be put under probation. These mobile devices have a T_S interval to renegotiate for less quantity or increase $\vartheta_{i,S}$, the service purchasing power for the SA class. When T_S times out, the corresponding reservations will be revoked.

The SA price $p_{l,S}(t)$ is calculated based on the demand for the SA bandwidth from existing and handoff mobile devices. The purpose is to give preference to requests with higher bid prices. In addition, the IA price is considered as well to reduce the probability that it rises higher than the SA price, which leads to additional SA sessions entering probation, reducing the disincentives for switching from the IA to SA class.

²The actual IA price is lower because unused SA bandwidth is also allocated to IA traffic.

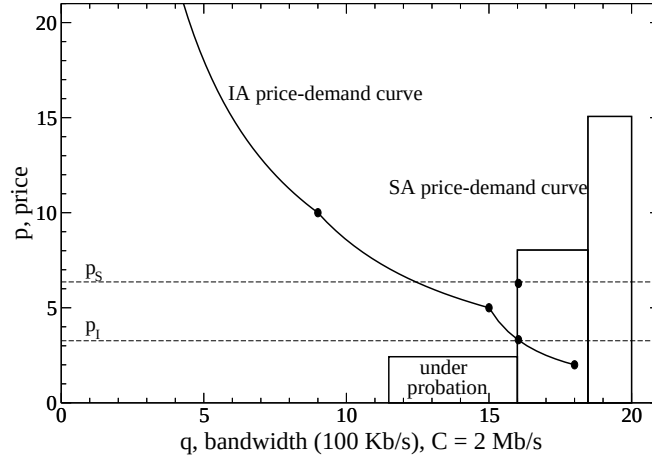


Figure 6-4: Example of Aggregated IA & SA Price Function

The SA admission control algorithm measures traffic load conditional on the bid price $\zeta_{i,S}$ in order to support the price-differentiated admission procedure. We denote the measured SA bandwidth demand as $\bar{\lambda}(t|p_{l,S})$. Here λ denotes the demand over a measurement window τ , which is set to be one order of magnitude larger than the session inter-arrival time-scale. In practice, we quantize the price $p_{l,S}$ into $\{p_K\}$, where $p_{l,S}$ price range $[0, \infty)$ is partitioned into $K + 1$ segments: $[0, p_1)$, $[p_1, p_2), \dots, [p_K, \infty)$. The quantization values p_i can be set from the measured bid-price histogram, so that each quantization segment will contain roughly the same probability mass. Since a bid price is inversely proportional to the corresponding bid quantity, as shown in Equation (6.18), with the assumption that the bid quantity is uniformly distributed and the SA service purchasing power does not vary much, we may set,

$$p_i = \begin{cases} \frac{p_K}{K-i+1} & i = 1, \dots, K \\ 0 & i = 0 \\ \infty & i = K + 1, \end{cases} \quad (6.19)$$

such that the probability mass function at each segment is the same, except for the first and last segments. Consequently, the quantization procedure needs only to

specify the maximum quantized price p_K and the number of quantization levels K . Denote t_n the end time of each measurement interval, we have,

$$\bar{\lambda}(t|p_k) = \alpha \frac{\text{cnt_b}_k}{t - t_{n-1}} + (1 - \alpha) \bar{\lambda}(t_{n-1}|p_k), \quad (6.20)$$

where $t_{n-1} < t \leq t_n$, $k = 0, \dots, K$ and cnt_b_k is the sum of $b_{i,S}$ arrived within $(t_{n-1} \ t]$ whose bid price $\zeta_{i,S} \in [p_k \ p_{k+1})$.

We calculate $p_{l,S}(t, \zeta_{i,S})$ over the interval t_{out} , which is the shortest time-out interval among all sessions under probation. When there is no session under probation, $t_{out} = T_S$. Therefore, t_{out} is the minimum interval at the end of which additional SA bandwidth is guaranteed to be available. Note that the departure of SA sessions within the t_{out} interval is not counted on, because their occurrence is statistical with no guarantee.

The SA price chosen from the quantized price set $\{p_k\}$ needs to satisfy two constraints. The first constraint is that future SA demand regulated by price p_k should not exceed the available SA bandwidth, as shown in Inequality (6.21). The right-hand-side of the inequality is the available SA bandwidth. The left-hand-side of the inequality is the predicted average SA demand whose bid price is no less than p_k . The control parameter $\gamma > 0$ is used to adjust SA demand estimation.

$$\gamma t_{out} \sum_{i \geq k} \bar{\lambda}(t|p_i) \leq (1 - \rho_{l,S}) \mathcal{C} \quad (6.21)$$

The second constraint represented by Inequality (6.22) relates to the condition that the SA price should remain higher than the IA price throughout the t_{out} interval. The right-hand-side of the inequality is the estimated IA price when all the allowable SA demand is met.

$$p_k \geq \frac{\Theta_{all,I}}{\rho_{l,I} \mathcal{C} - \gamma t_{out} \sum_{i \geq k} \bar{\lambda}(t|p_i)}. \quad (6.22)$$

The choice of $p_{l,S}$ is then decided as:

$$p_{l,S} = \min\{p_k \mid p_k \text{ satisfies Ineq (6.21) and (6.22)}\}. \quad (6.23)$$

The baseline SA allocation algorithm is shown in Figure 6-5.

```

admission_control { // arrival of reservation request
    update  $\bar{\lambda}(t)$ ;
    if ( $b_{i,S} > (1 - \rho_{l,S})\mathcal{C}$ )
        reject(); // no enough bandwidth
    elseif ( $\zeta_{i,S} < p_{l,S}$ )
        reject(); // bid price too low
    else
        calculate_SA_price();
}
calculate_SA_price {
    update  $(1 - \rho_{l,S})\mathcal{C}$ ; // available SA bandwidth
    update  $t_{out}$ ;
    search for the smallest  $p_k$  in the quantized
    price set  $\{p_k\}$  such that
         $\gamma t_{out} \sum_{i \geq k} \bar{\lambda}(t|p_i) \leq (1 - \rho_{l,S})\mathcal{C}$ ;
         $p_k \geq \frac{\Theta_{all,I}}{\rho_{l,I}\mathcal{C} - \gamma t_{out} \sum_{i \geq k} \bar{\lambda}(t|p_i)}$ ;
     $p_{l,S} = p_k$ ;
    broadcast price-service menu;
}
maintain_SA_allocation {
    maintain sorted list of  $\zeta_{(k),S}$ ;
    if ( $\zeta_{i,S} < p_{l,S}$  AND  $i$  is not under probation)
        put  $i$  under probation, start timeout timer;
    if ( $\zeta_{i,S} \geq p_{l,S}$  AND  $i$  is under probation)
        move  $i$  out of probation;
    if ( $i$  is under probation and timer expires)
        remove  $i$ 's SA reservation;
}

```

Figure 6-5: Baseline SA Allocation Algorithm at Access Point

6.5.2 IA Allocation Pegging

So far we address the bandwidth-hogging problem by allowing users with higher bid prices to preempt the incumbent lower-price users after a warning interval. However, additional mechanisms are needed to discourage bursty data applications

from switching from the IA to the SA service because preemption from the SA to IA service does not penalize bursty data applications.

One disincentive is the usage accounting model that can be used with the block-rate service charge. We can count the SA usage minutes by the holding time of the reservation regardless of actual bandwidth consumption. The second disincentive is the higher SA price over the IA price. The constraints (6.21) and (6.22) in Section 6.5.1 enforce that $p_{l,S}(t) > p_{l,I}(t)$. However, these two disincentives are insufficient to guarantee that the throughput offered to an IA service user will be always larger than the corresponding throughput the user would received using an albeit “hypothetical SA session”. The complexity comes from the fact that there is no admission control for the IA service class traffic. The IA price (and subsequently the SA price) can rise sharply with a surge in IA demand. However, in the case of previously admitted SA sessions, their reservations will be maintained for T_S seconds when their SA unit valuation fall below SA price, (i.e., under probation with $\zeta_{i,S} < p_{l\supset i,S}$). Therefore, the same SA allocation stabilizing mechanism also provides an incentive for IA users to switch to the SA service if they have prior knowledge of the increase in IA demand.

To remove this arbitrage possibility, we explicitly calculate $\Gamma_i(t)$, the accumulated throughput surplus of an IA session in comparison to its hypothetical SA session. When $\Gamma_i(t)$ is in danger of becoming negative, the allocation for the IA session is pegged to the previous amount. The simulated hypothetical SA session uses $\vartheta_{i,I}$ as its service purchasing power. We denote $\tilde{\zeta}_{i,S}(t)$ as its bid price. Because our purpose is to simulate a strategy that maximize the received SA allocation through a (hypothetical) continuous renegotiation, we have,

$$\tilde{\zeta}_{i,S}(t) = \min\{\tilde{\zeta}_{i,S}(t^-), p_{l\supset i,S}(t)\}. \quad (6.24)$$

When $\tilde{\zeta}_{i,S}(t) < p_{l\supset i,S}(t)$, the hypothetical SA session enters probation. At the end of probation, (i.e., $t + T_S$), it will request a new reservation with bid price $\tilde{\zeta}_{i,S}(t + T_S) = p_{l\supset i,S}(t + T_S)$ because $p_{l\supset i,S}$ represents the minimum bid price for the hypothetical SA session to be admitted.

The accumulated throughput difference is calculated as follows. Whenever there are changes in IA or SA prices, the throughput accumulation is updated as follows:

$$\Gamma_i(t_n) = \Gamma_i(t_{n-1}) + \left(\frac{1}{p_{l\supset i,I}^*(t_n^-)} - \frac{1}{\tilde{\zeta}_{i,S}(t_n^-)} \right) (t_n - t_{n-1}), \quad (6.25)$$

where t_n^- denotes the time just before price changes.

Equation (6.25) is a backward accounting of the accumulated $\Gamma_i(t)$ in the past. To enforce $\Gamma_i(t) \geq 0$, we need to predict the value of $\Gamma_i(t)$ when it is decreasing, (i.e., when $p_{l\supset i,I}^*(t) > \tilde{\zeta}_{i,S}(t)$). In this case, the hypothetical SA session has been under probation because $p_{l\supset i,S}(t) \geq p_{l\supset i,I}^*(t)$. We denote the remaining probationary period as $t_{i,prob}$. If $\Gamma_i(t) + \left(\frac{1}{p_{l\supset i,I}^*(t)} - \frac{1}{\tilde{\zeta}_{i,S}(t)} \right) t_{i,prob} < 0$, that is, $\Gamma_i(t)$ is not large enough to cover the throughput deficit in the next $t_{i,prob}$ interval, then the allocation for the IA session is pegged at the value $\tilde{q}_{i,I}$:

$$\tilde{q}_{i,I} = \frac{\vartheta_{i,I}}{p_{l\supset i,I}^*(t)} = \vartheta_{i,I} \left(\frac{1}{\tilde{\zeta}_{i,S}(t)} - \frac{\Gamma_i(t)}{t_{i,prob}} \right) \quad (6.26)$$

The IA allocation algorithm is modified so that that during allocation pegging of session i , its state will be temporally disabled in the algorithm.

Remark: We note that the algorithm is only executed for IA sessions that are under a busy period. A busy period is a consecutive interval when the measured throughput of an IA session is within, e.g., 70%, of its allocation, i.e., $\bar{b}_{i,I}^{up}(t_n) + \bar{b}_{i,I}^{down}(t_n) \geq 0.7b_{i,I}^*$.

Remark: The effect of IA allocation pegging on new sessions is that both IA and

SA prices will be temporarily elevated, and SA session arrivals may face admission failure. In the worst case, a surge of IA demand may cause all the incumbent IA sessions entering allocation pegging and all the incumbent SA sessions entering probation. This situation lasts at most T_s interval, and is usually shorter with first few idle IA sessions and terminated SA sessions.

6.6. Mobile Device Strategy

The incentive algorithms discussed in the previous section are designed to constrain a mobile user's strategy space while minimizing the amount of signaling overhead. The resulting allocation has the following properties.

6.6.1 Fairness

A bargaining solution is a rule that assigns a feasible agreement to an allocation problem, where feasibility means that the total amount of allocation is less the total available resource and the minimum required performance of each session is guaranteed. Nash proposed four independent properties and showed that they are simultaneously satisfied only by the Nash Bargaining solution [78].

Definition 4 *An allocation is Nash Bargaining Fair when it has the following properties:*

- *Pareto optimal, (i.e., it is impossible to strictly increase the allocation of a user without strictly decreasing another one);*
- *Independence of positive linear transformations, (i.e. the bargaining point is unchanged if the performance objectives are affinely scaled);*
- *Symmetry: users with the same minimum performance measures and the same utilities will have the same performance regardless of their specific labels; and*

- *Independence of Irrelevant Alternatives: allocation is not affected by enlarging the domain if a solution to the problem with the larger domain can be found on the restricted one.*

Proposition 15 *The IA allocation mechanism is asymmetric Nash Bargaining Fair with different budget $\vartheta_{i,I}$, maximum bandwidth $b_i^{\max}$ and zero minimum bandwidth requirement.*

Proof: This is a special case (i.e., a single link allocation) of the asymmetric Nash Bargaining solution given in [109], where asymmetry refers to the different values of $\vartheta_{i,I}$ serving as user budget³. Using the Kuhn-Tucker condition [62], we have that the IA allocation of $b_{i,I}(t) = \min \left\{ \vartheta_{i,I}/p_{l \ni i,I}(t), b_{i,I}^{\max}(t) \right\}$ solves the maximization problem: $\max_{\mathbf{x}} \{ \prod_i x_i^{\vartheta_{i,I}} \}$, $0 \leq x_i \leq b_{i,I}^{\max}$, and $\sum_i x_i = \mathcal{C}_l$. From Proposition 5.1 of [109], the unique solution to the above maximization problem is the asymmetric Nash Bargaining solution under the condition that the user performance function $f(x_i) = x_i$, which is a reasonable condition under our non-monetary pricing scheme as well. Therefore, the IA allocation mechanism is asymmetric Nash Bargaining Fair. $\square$

6.6.2 Dominant Mobile Strategy

Because our incentive engineering mechanisms are designed to constraint the strategy space of users to cooperative behaviors, we have the following properties:

Proposition 16 *For wireless users preferring high throughput, the dominant strategy is to subscribe to the IA service.*

³Asymmetric Nash Bargaining solution satisfies all the Nash Bargaining Fair properties except symmetry, and the asymmetry reflects in bandwidth allocation proportional to user budget

Proof: The best alternative strategy is the hypothetical SA session used in the allocation pegging algorithm, which subscribes to the SA service and aggressively renegotiates its bid quantity whenever $p_{i,S}$ is less than the previous bid price. However, the allocation pegging algorithm ensures that $\Gamma_i \geq 0$. Therefore, the accumulated throughput of an IA session is always the highest among any other alternative strategy. $\square$

Proposition 17 *For wireless users preferring allocation stability, the dominant strategy is to subscribe to the SA service.*

Proof: This is governed by the ranked allocation stability of SA service and the additional T_S warning interval when the bid price falls below $p_{i,S}$. $\square$

In contrast, the corresponding IA service allocation is constantly changing and can fall below the SA bid quantity. Note, that the condition $\Gamma_i \geq 0$ does not prevent this because it only acts on the accumulated throughput, not the instantaneous throughput value. Because the service purchasing power ϑ_i is non accumulated, mobile devices have no incentive to save it. Since the SA allocation stability is ranked by $\vartheta_{i,S}/q_{i,S}$, inflating $q_{i,S}$ will reduce allocation stability, while deflating $q_{i,S}$ will affect application performance. Therefore, we have,

Proposition 18 *The dominant strategy for a wireless user of the SA service is to truthfully declare the required bandwidth amount $q_{i,S}$.*

Remark: With the unique dominant strategy exists for a single-stage game, the unique Nash equilibrium of this game is for each player to play the dominant strategy; and the corresponding finitely repeated game has a unique subgame perfect outcome: i.e., the Nash equilibrium of the single-stage game is played at every stage [45].

For IA service users, the measurement-based allocation removes any strategic play by mobile devices. Because the access point subtracts $\vartheta_{i,S}$ from the remaining portion of the IA class, (i.e., $\vartheta_{i,I} = \vartheta_i - \vartheta_{i,S}$). The only decision remaining open to the mobile device is to decide how to split its service purchasing power ϑ_i into $\vartheta_{i,S}$ and $\vartheta_{i,I}$ amounts.

The actual partition of ϑ_i between the IA and SA service classes is determined by the utility function of a user, $u_i(q_{i,I}, p_{i,S})$, which is a function of the allocation quantity of the IA class, and the bid price (which is an indicator of allocation stability) of the SA reservation. Therefore, the optimum partition of ϑ_i is calculated by

$$\text{optimal_}\vartheta_{i,S} = \arg \max \{u_i(q_{i,I}, p_{i,S})\} \quad (6.27)$$

$$= \arg \max \left\{ u_i \left(\frac{\vartheta_i - \vartheta_{i,S}}{p_{l \ni i, I}}, \frac{\vartheta_{i,S}}{q_{i,S}} \right) \right\}. \quad (6.28)$$

A mobile device's strategy is to decide $\text{optimal_}\vartheta_{i,S}$ based on its SA service demand $q_{i,S}$ and the IA service price $p_{l \ni i, I}$.

An example of $u_i(q_{i,I}, p_{i,S})$ can have a form of

$$u_i(q_{i,I}, p_{i,S}) = \alpha q_{i,I} + \beta p_{i,S} = \alpha \left(\frac{\vartheta_i - \vartheta_{i,S}}{p_{l \ni i, I}} \right) + \beta \left(\frac{\vartheta_{i,S}}{q_{i,S}} \right), \quad (6.29)$$

where α and β are control parameters. In this case,

$$\text{optimal_}\vartheta_{i,S} = \begin{cases} \vartheta_i, & \beta/q_{i,S} > \alpha/p_{l \ni i, I} \\ 0, & \beta/q_{i,S} < \alpha/p_{l \ni i, I} \\ x, x \in [0, \vartheta], & \text{otherwise} \end{cases} \quad (6.30)$$

This example provides a good intuitive strategy: when the utility valuation for the stability of the SA allocation is more important, it is optimal to use all the service

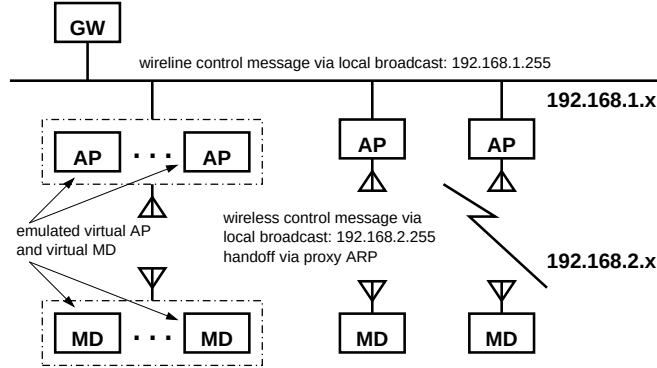


Figure 6-6: Experimental Wireless Testbed

purchasing power to bid for an SA allocation; when the utility valuation for the amount of IA allocation is higher, the opposite is optimal. This simple strategy can be further enhanced: a mobile device can reduce the SA bid price and only increase it at the end of the probation interval T_S .

6.7. Experimental Results

In this section, we present experimental and simulation results showing a number of beneficial properties of our incentive-based approach to delivering wireless services.

6.7.1 Wireless Testbed

As a means to test the feasibility of the proposed algorithms, we implement the proposed algorithms in an experimental wireless testbed call *wincent* (for wireless incentive engineering testbed)⁴. Figure 6-6 shows our testbed. We use Linux PCs and laptops as access points and mobile devices, respectively. Access points are interconnected to each other using 10BaseT Ethernet, forming a wireless packet cellular network using IEEE 802.11b wireless radios. The access points rely on the

⁴*wincent* open source code is available from the Web (<http://www.comet.columbia.edu/cubnet/wincent>)

Linux Traffic Control module (TC) for traffic shaping [64] in order to assign each mobile device its allocated bandwidth. We modified the IEEE 802.11 wireless device driver to enable traffic snooping for measuring the bandwidth consumption of each mobile device.

Importantly, our testbed can also operate in a simulation/emulation mode so that the same algorithms can be evaluated with a larger number of access points and mobile devices. We modified the IEEE 802.11 wireless device driver to enable traffic snooping for measuring the bandwidth consumption of each mobile device. In addition, the software part of the testbed is designed in a modular fashion. The testbed also requires modifying the wireless LAN driver traffic measurement. After collecting the required statistics, the access points make the required calculations and accordingly broadcast the new price for the mobile hosts, it also restricts the downlink of the participating active mobiles reflecting the new prices. This mechanism will allow mobile devices to either change their behavior or change the amount of their budget dedicated to that specific session.

The handoff control function is emulated at network layer because there is a single shared wireless LAN. Handoff rerouting is done by updating ARP tables at access points (i.e., multi-homed PCs), which is managed by proxy ARP. This coarse form of handoff control is sufficient for early experimentation. We have developed more sophisticated fast handoff protocols [101] that will be integrated into a future version of our testbed.

6.7.2 Parameter Tuning

The first test conducted focuses on measuring the response of the TC traffic shaper since the measurement-based algorithm relies heavily on it for restricting the bandwidth consumption of the mobile devices on the end of the access points. Figure 6-7

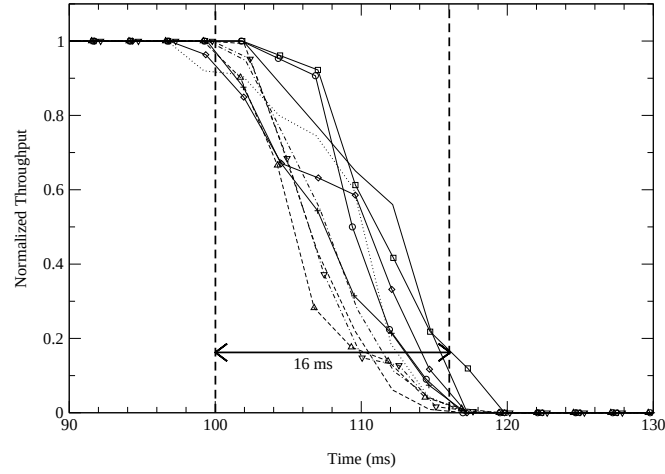


Figure 6-7: Linux Traffic Control Response

shows the result of the experiment. We measure the throughput from a large file transfer by a single mobile device. At time 100 ms, the shaping rate of Linux TC is changed to zero. Our intention is to measure the time elapse before Linux TC completely reduces the traffic flow to zero. In order to improve the reliability of the statistics, the test is repeated for ten times with different initial shaping rate. In the figure all throughput traces from the ten separate tests are shown together with throughput values normalized. We observe that all of the measured throughput values take an additional 16 ms to fall down to zero. This interval is highlighted in the figure by the region between the two vertical dot lines. In the figure, most of the curves fall down to zero inside the region. The few that do not are the ones with shaping rate cut at a later time than 100 ms. The value of 16 ms obtained in this test influences the choice of τ used in the IA measurement algorithm. However, after conducting several experiments on the testbed while varying the parameter τ we noticed that decreasing it below 30 ms does not bring any benefit for the overall performance of the system and algorithms. Thus, the experiment shown next uses 30 ms as the measurement interval as well as the price broadcasting time.

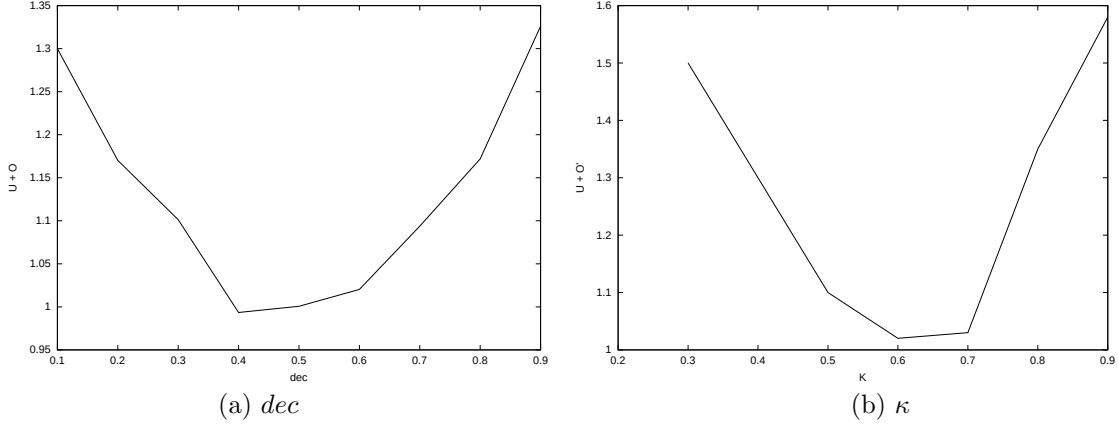


Figure 6-8: Parameter Setting

The second set of experiments focus on tuning the parameters dec and κ of Equation 6.16. Both parameters affect the trade-off between improving the utilization of IA bandwidth unused by idle mobile devices, and reducing the chance of congestion when those idle mobile devices become active again. The experimental scenario comprises two mobile devices consuming IA bandwidth. The first mobile device downloads a large file while the second mobile device performs a sequence of web transactions with on-off bursty traffic. When the second mobile device is idle (i.e., in an “off” interval), bandwidth could be under utilized. We measure the total amount of bits left unused during an “off” interval and denote it as U . When the second mobile device is active (i.e., in an “on” interval), congestion could happen. We measure the total amount of bandwidth allocation (i.e., $b_{i,I}^{\max}(t_n)$ in Equation 6.14) for both active mobile devices during an “on” interval and denote it as O . For the parameter dec , intuitively, a large dec leads to the quicker allocation of unused bandwidth to active mobile devices, but also to a larger chance of congestion when idle mobile devices become active. Therefore, a large dec means a smaller U but a larger O . In 6-8(a), we plot the value $U + O$ against different values of dec . It shows that to minimize $U + O$, the optimal choice of dec is 0.4.

We repeat the same experiments for κ in Equation 6.16, which decides the lower threshold to invoke adjustment of γ . A small κ means that the algorithm is satisfied with a lower bandwidth utilization, and not to redistribute unused bandwidth to active mobile devices. In contrast, a large κ will increase bandwidth utilization (reduce U) but also increase the chance of congestion (increase O) as well because the average load of the system will be operating at a higher level. In addition, a large κ also leads to more frequent oscillations in IA bandwidth and price changes. 6-8(b) shows the evaluation results by plotting $U + O$ against different values of κ . Once again, we observe that to minimize $U + O$, the optimal choice of κ should be in the region of $[0.6 \ 0.7]$, which basically means that bandwidth should be redistributed to active mobile devices only when the bandwidth is 60% to 70% under the *threshold* (0.9) value. In what follows, we set κ to 0.7 and *dec* to 0.4.

6.7.3 IA and SA Allocation Algorithm

In this test, we use four mobile notebooks sharing the IA bandwidth in a single cell. Three of the mobile devices (#1, #3 and #4) have identical service purchasing power ϑ_i , while the second mobile device (#2) subscribes to a premium service plan giving it a purchasing power ϑ_i that is double that of the other mobile devices.

The first experiment presented in this section is designed to show the behavior of applications using IA allocations. The applications are either bursty in nature such as web browsing or greedy such as FTP download. Figure 6-9(a) and (b) show the throughput and normalized throughput traces of the four mobile devices, respectively. In the experiment, mobile devices #1 and #2 generate web traffic, while mobile device #3 is checking emails and mobile device #4 is downloading using FTP. The throughput growth indicates that starting early has an initial advantage. However, as the traffic enforcement mechanism takes effect, the measured through-

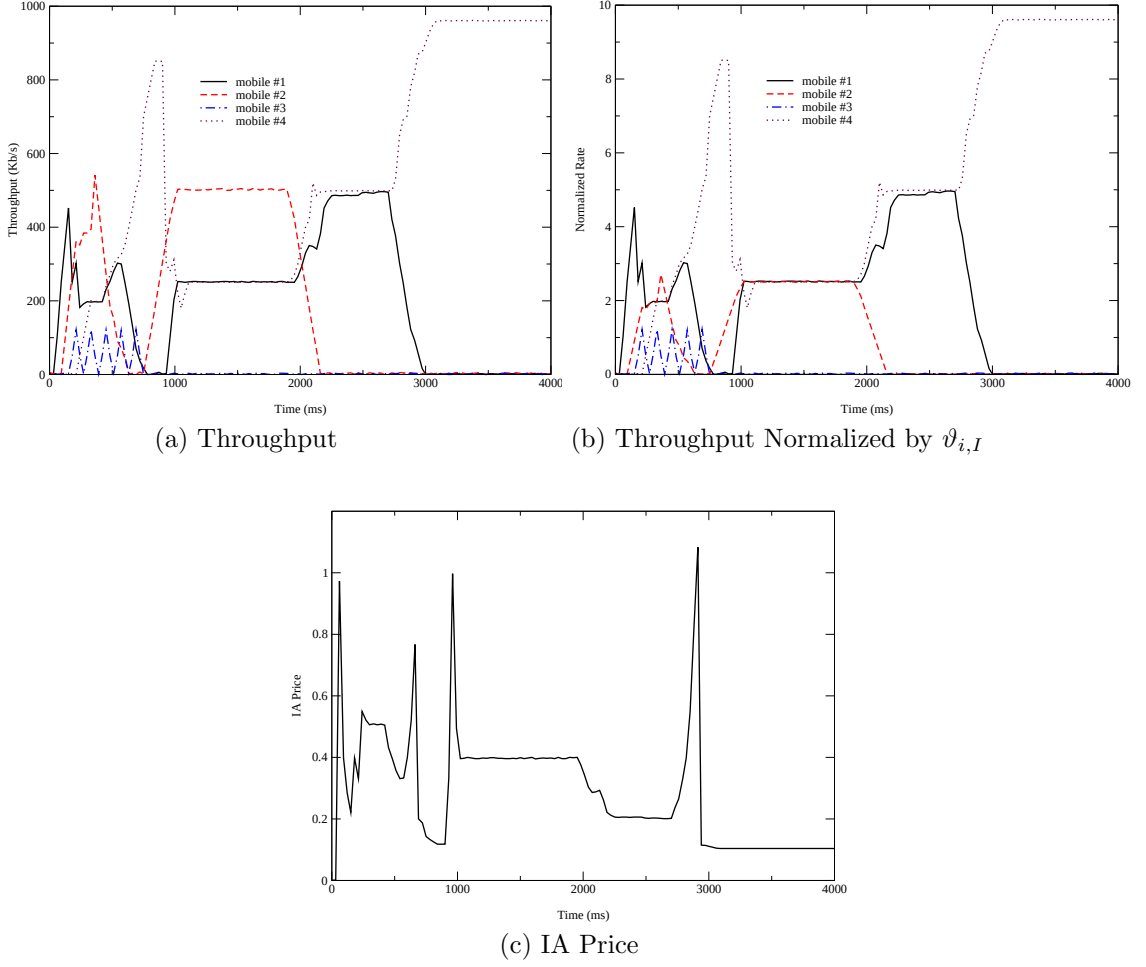


Figure 6-9: IA Allocation Experiment

put quickly settles to the theoretical allocation values, which are Nash Bargaining Fair, (i.e., mobile device #2 receives twice the throughput as mobile devices #1, #3 and #4 individually receive, as shown in Figure 6-9(a)). In Figure 6-9(b), because the throughput measurement is normalized by service purchasing power, therefore, between 1000 and 3000 ms, all mobile devices receive the same normalized throughput. Figure 6-9(c) shows the changes in price for the same experiment. The spikes in price correspond to the traffic surges in throughput measurement. These narrow price spikes also indicate the effectiveness of our pricing mechanism in regulating traffic.

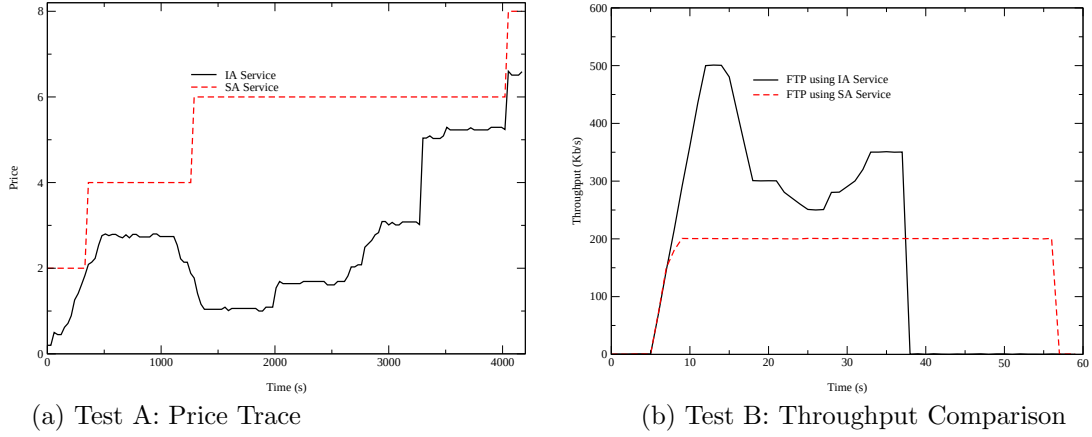


Figure 6-10: IA/SA Allocation Experiments

The second experiment is intended to show the interactions between IA and SA traffic and the incentive offered to users to declare truthfully whether their traffic should be carried as IA or SA. Figure 6-10(a) shows the IA and SA prices with respect to the change in traffic. The figure starts in an initial state where no traffic exists for both classes, then IA traffic is present in the network and drives the price higher. An increase in the SA price takes place as new traffic is generated. Then the IA price drops back as the traffic decreases while SA keeps increasing as more demand is generated. Finally, we can observe an increase in IA traffic as more traffic is generated. We also observe the discrete change in the SA price due to price quantization. We also conducted an experiment trying to compare using SA and IA for a greedy application in order to demonstrate the incentive of using IA for such applications. The trace in Figure 6-10(b) shows the download of the same file under similar network conditions for IA and SA. Although SA provides a stable allocation, IA proves more advantageous as bursts can occur allowing the download to complete earlier.

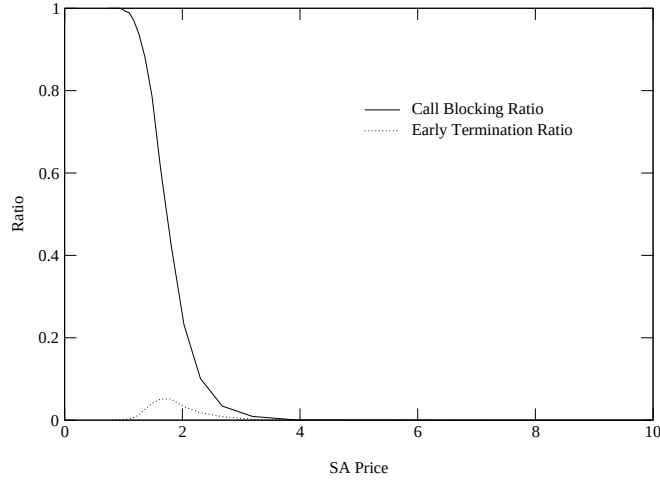


Figure 6-11: SA Service: Allocation Stability Ranking

6.7.4 Pricing Dynamics

In this experiment, we use the emulation platform to focus on the pricing dynamics between SA and IA allocations. The cell capacity is set to 1 Mb/s. We simulate 50000 SA service requests according to the Poisson process. The average arrival interval is 5s. The average holding time, (i.e, without early termination) is 15s. The request quantity is uniformly distributed in [10 100] Kb/s. This translate into an average SA load of 15%. The IA traffic activity is generated by activating a random number of mobile devices every second. The random number is uniformly distributed in [1 20]. Each user's service purchasing power ϑ_i is randomly assigned from two types: 50 and 100. The SA arrival measurement $\bar{\lambda}(t|p_k)$ is segmented over $K = 20$ quantized price segments, with $p_K = 10$.

The warning interval $T_S = 20s$ for most of the cases. In addition, the demand estimation parameter $\alpha = 0.7$. Figure 6-11 illustrates the call drop ratio and early termination ratio for each of the 20 quantized-prices. Here we clearly observe the effect of ranking in admission success probability and allocation stability. The early

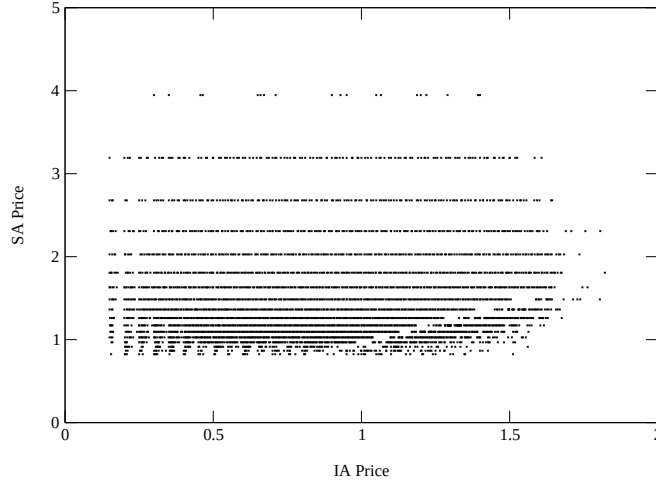


Figure 6-12: Relation between SA and IA Prices

termination ratio for small SA valuations are zero because all the calls within those quantized price segments are blocked given the market price $p_{l,s}$. The sharp drop in both call blocking and early termination ratio for high quantized-prices indicate the incentive for SA users not to inflate their bandwidth requests.

In Figure 6-12, we plot the price pair $\{p_{l,I}, p_{l,S}\}$. We observe that because the SA prices are chosen from a set of quantized price values, SA prices are concentrated at a few values, and hence are relatively stable with respect to changes in the IA prices.

The effect of the warning interval T_S on the SA price is shown in Figure 6-13. Here we rerun the simulation for different T_S . The results indicate that additional service purchasing power is needed for stabilizing allocations. This value increases with T_S because with a large T_S sessions under probation will have a longer lifetime, and the bandwidth market will have less liquidity.

Our incentive engineering mechanisms do not guarantee bandwidth reservation, to avoid distorting the market price. However, a third party broker may act as an additional source of service purchasing power to sessions under probation, and hence

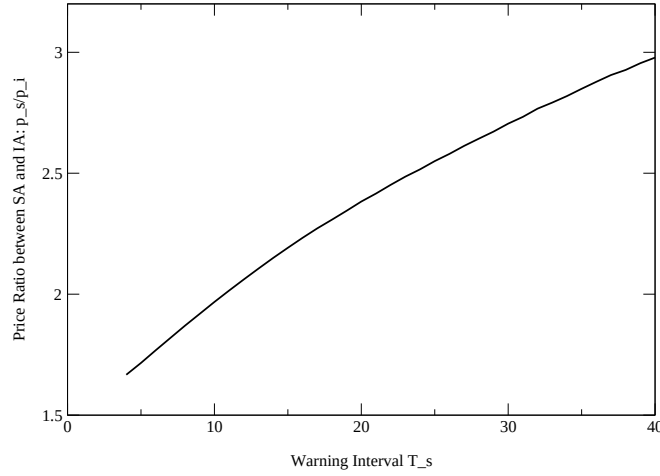


Figure 6-13: Effect of T_S on the SA Price

provide guaranteed allocation stability without distorting the incentive mechanisms. Figure 6-14 shows the amount of service purchasing power needed for one bandwidth unit as a function of the unit valuation of the original session. Because the additional service purchasing power reflects the opportunity cost (or shadow price) of stabilizing a session's allocation, service providers can use the accumulated additional service power to charge for a guaranteed allocation service.

6.8. Summary

The contribution of this chapter is as follows. We provided a solution to the problem of engineering incentives for edge-based wireless access services, which offer both higher throughput (IA) for bursty data applications and more stable allocation (SA) for real-time applications. Our incentive engineering model includes the use of service purchasing power and a price-service menu to effectively constrain the strategy space of users to a set of cooperative behaviors, leading to fair usage of IA services and truthfully self-differentiation in SA service. The algorithm design minimizes the protocol overhead on mobile devices and over-the-air. The rate enforcement

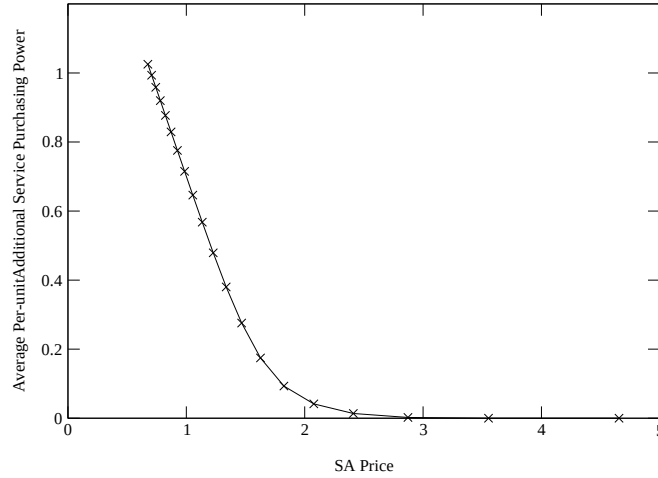


Figure 6-14: Additional Service Purchasing Power for Allocation Guarantee

algorithm controls the downlink traffic tolerating incomplete information caused by the absence of mobile device participation in the control algorithm. The reservation based SA service relieves users from declaring session lifetime, and gives early warning of any pending allocation degradation while keeping potential arbitrage between IA and SA services to zero. Users of higher bid-price may preempt lower-price users given a warning interval. In addition, users may follow their individual utility function to partition the service purchasing power for IA and SA allocation. The design provides differentiated relative/soft QOS among users according to their bid price and assigned service purchasing power.

We should note that even though our scheme is designed for IEEE 802.11b WLAN based access network, it can also benefit cellular data networks such as GPRS [2] and CDMA2000 [3]. Since cellular networks are based on centralized control for both down/uplink at base stations, the IA control approximation discussed in Section 6.4.3 is not necessary. However, the benefit of incentive compatibility for the design of SA and IA classes can assure service providers that users will not abuse bandwidth reservations.

Chapter 7

Conclusion

Realizing flexible and efficient bandwidth service management for Internet and its wireless extensions is a challenging problem. This thesis has presented a number of contributions to this problem. We have proposed a broad set of algorithms for utility-based adaptation, dynamic provisioning and incentive engineering. Collectively these algorithms contribute to a better understanding of this complex problem that encompasses multi-dimensional issues such as application requirements, network mechanisms, service differentiation, and user incentives.

The use of bandwidth utility functions has served as a unifying abstraction in this thesis for efficient network allocation and control, application responsiveness to change such as controlled degradation, and content delivery trade-offs. It is interesting to note that originally bandwidth utility functions were first adopted by the signal processing community based on Shannon's early work [92] on rate distortion. Today, however, utility-based network bandwidth allocation has largely remained an abstraction for economic theory and market mechanisms. Our work has contributed to a broader application of utility functions in networking. In Chapter 2, we presented a new formulation for utility functions that leads to the development of flexible bandwidth management models. We extended utility functions to non-video applications, adding normalization parameters to support service

differentiation, and applying utility prediction to accommodate network adaptation time-scale. Recently, we have noted that utility-based (i.e., rate distortion function based) transcoding and media scaling techniques have begun to be discussed in the literature and are being considered for MPEG-4 over-the-air applications, pushing the abstraction into the packet for possible exploitation by future utility-based network algorithms.

In Chapter 3, we presented a number of link sharing algorithms that could exploit utility function state information. We presented a number of utility-based foundation algorithms that included utility-differentiating and utility-maximizing bandwidth allocation targeted to operate at single bottleneck links. While these allocation schemes proved efficient there is a need to aggregate utility function information when possible to make the solution more scalable to accommodate greater number of flows and more frequent adaptation in utility-based networks. The complexity of state management in these networks where each flow may potentially has its own form of utility function presents a number of significant challenges to scalability that was addressed in Chapter 4. We presented the design and analysis of a utility-based model for an edge-based wireless access networks that efficiently supported application controlled degradation needs, and importantly, delivered this using scalable resource allocation algorithms. The model comprised a distributed architecture and messaging protocol, where distributed adaptation handlers managed application specific adaptation scripts that were capable of renegotiating bandwidth reservations on behalf of applications in an efficient manner.

Next, we studied dynamic bandwidth provisioning for core IP networks in support of quantitative differentiated services. The design space for core networks is considerably more restrictive than edge-based access networks in terms of the granularity of control and state management issue. Typically, core networks are stateless

and in the case of bandwidth service management the time-scale for control is coarse. This requires that the network allocation and provisioning algorithms be effective when there is limited or no global state. In the core network, per-flow states such as utility functions are maintained at the edge of the core network. In this case, the challenge of bandwidth service management is to gaining effective control using coarse-grain control information, resulting from aggregation of both flow states and control mechanisms. One outcome of aggregated edge rate control is the problem of flow control for point-to-multipoint traffic aggregates. Chapter 5 presents a solution to this problem and considers different optimizing goals such as fairness, minimizing the worst case rate reduction, or a combination of both. We demonstrated that our model is capable of delivering capacity provisioning in an efficient manner and providing quantitative delay-bounds with differentiated loss across per-aggregate service classes in core networks. The key enabler to this solution was a measurement-based multi-service virtual queue technique extend from [59], which effectively predicts traffic overload conditions without assuming any traffic model. To our best knowledge this is the first published solution to this problem.

The contributions discussed so far have focused on the design of efficient utility-based bandwidth allocation algorithms. One conjecture we hold is that the correct operation of these network mechanisms also require end-user cooperation (e.g., in choosing the appropriate service class, and truthfully declaring their utility functions). However, such cooperation can conflict with end-users' self-optimizing goals unless the network optimization goal is incentive compatible with end-users' selfish goals. The final contribution of the thesis addressed this problem by creating incentives for differentiated service mechanisms in edge-based wireless networks. An important aspect of bandwidth service management is the elimination of arbitrage inherent in differentiated service models. Previous efforts have considered conges-

tion pricing, which is theoretically sound but practically infeasible to realize because it is entangled with service charging which is the domain of the business stakeholder. As discussed at the beginning of this thesis this entanglement of control and pricing creates great complexity in the mechanism design and deployment, resulting in an unwanted tussle [25] between different stakeholders: network control and business. In Chapter 6 we resolve this tussle by decoupling congestion pricing and monetary concerns. The resulting constraints lead to a new set of incentive engineering designs that reduce the strategy space of users to remove arbitrages in service differentiation model leading to the incentive compatibility of our service differentiation framework. Our design minimizes protocol messaging overhead imposed on wireless subscribers while possessing a number of beneficial properties including Nash bargaining fairness for the instantaneous allocation service, and incentive compatibility for mobile users promoting the truthful declaration of service preferences.

Incentive engineering of network mechanisms is an emerging research direction with great potential and many outstanding challenges. Our attempt in Chapter 6 is only a first attempt applied to wireless access networks. Future work is needed to extend the results reported in this thesis to multi-hop networks including wireless ad-hoc networks, as well as to network peering points between different service providers networks, where the stakeholders of the tussle become more complex involving network control and multiple cooperative/competitive business organizations. This also leads to another research challenge; that is, the modeling and integration of diverse and conflicting business policies using utility-based algorithms over single link (i.e., at the network peering points) and across multi-hop networks (i.e., an ISP providing virtual network to different customers). It would be important to analyze the property of such a network where different bandwidth management policies apply. Chapter 3 presents a good starting point through the use of utility-based

hierarchical scheduling algorithms. It remain to be seen what other properties beyond utility-based max-min fair discussed in Chapter 4 could be derived by applying diverse utility-based policies over networks.

Chapter 8

My Publications as a Ph.D. Candidate

My publications as a Ph.D. candidate (1996-2002) are listed below. This list also includes research papers that are indirectly related to the work presented in this thesis, including, the design and implementation of QOS signalling for programmable mobile networks, deployment trials for wireless broadband networks, and analysis of network bandwidth peering for service differentiation.

8.1. Patents

- R. R-F. Liao and A.T. Campbell. US Patent Application WO0169851. Method and Apparatus for Allocation of Resources, Columbia University, March 2000.

8.2. Journal Papers

- R. R-F. Liao, R. H. Wouhaybi, and A. T. Campbell. WinCent: Wireless Incentive Engineering. to appear, *IEEE J. Select. Areas Commun., Special Issue on Recent Advances in Wireless Multimedia*, 4th Quarter 2003.
- R. R.-F. Liao and A. T. Campbell. A Utility Based Approach for Quantitative

Adaptation in Wireless Packet Networks. *ACM Baltzer J. Wireless Networks (WINET)*, 7(5):541–557, September 2001.

- N. Semret, R. R.-F. Liao, A. T. Campbell, and A. A. Lazar. Pricing, Provisioning and Peering: Dynamic Markets for Differentiated Internet Services and Implications for Network Interconnections. *IEEE J. Select. Areas Commun., Special Issue on QOS in the Internet*, 18(12):2499–2513, December 2000.
- A.T. Campbell, M. Kounavis, and R. R.-F. Liao. On Programmable Mobile Networks. *J. Computer Networks*, 31:741–765, April 1999.

8.3. Journal Papers under Submission

- R. R-F. Liao and A. T. Campbell. Utility-based Network Adaptation for Multimedia Content Delivery. under submission, *IEEE/ACM Trans. on Networking*, August 2001.
- R. R-F. Liao and A. T. Campbell. Dynamic Core Provisioning for Quantitative Differentiated Service. under submission, *IEEE/ACM Trans. on Networking*, July 2001.

8.4. Magazine Papers, Review Articles and Book Chapters

- O. Angin, A. T. Campbell, M. E. Kounavis, and R. R.-F. Liao. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. *IEEE Personal Commun. Mag.*, August 1998. Source code freely available at <http://comet.columbia.edu/mobiware>.
- A. T. Campbell, R. R.-F. Liao, and Y. Shobatake. Supporting QOS Controlled Handoff in Mobiware. (Eds.) J. Holtzman, and M. Zorzi. *Advances in*

Wireless Communications, Kluwer Academic Publishers, ISBN 0-7923-8126-2, Chapter 2, 157–171, 1998.

- O. Angin, A. T. Campbell, L-T. Cheok, R. R-F. Liao, K-S. Lim, and K. Nahrstedt, IWQOS'97 Workshop Summary. (Eds.) A. T. Campbell, and K. Nahrstedt. *Building QOS into Distributed Systems*, Chapman and Hall, ISBN 0-412-80940-0, xiv–xli, 1997.

8.5. Conference Papers

- R. R.-F. Liao and A. T. Campbell. On Programmable Universal Mobile Channels in a Cellular Internet. In *Proc. ACM MOBICOM (Mobicom'98)*, 191–202, Dallas, TX, October 1998.
- N. Semret, R. R.-F. Liao, A. T. Campbell, and A. A. Lazar. Peering and Provisioning of Differentiated Internet Services. In *Proc. IEEE INFOCOM*, 2:414-420, Tel Aviv, Israel, March 2000.
- R. R.-F. Liao, R. H. Wouhaybi, and A. T. Campbell. Incentive Engineering in Wireless LAN Based Access Networks. In *Proc. Int'l Conf. Network Protocols (ICNP 2002)*, Paris, France, November 2002.
- R. R.-F. Liao and A. T. Campbell. Dynamic Core Provisioning for Quantitative Differentiated Service. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service (IWQoS 2001)*, Karlsruhe, Germany, June 2001.
- R. R.-F. Liao and A. T. Campbell. Dynamic Edge Provisioning for Core Networks. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service (IWQoS 2000)*, Pittsburgh, USA, June 2000.

- N. Semret, R. R.-F. Liao, A. T. Campbell, and A. A. Lazar. Market Pricing of Differentiated Internet Services. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service (IWQoS'99)*, London, UK, May 1999.
- G. Bianchi, A. T. Campbell, and R. R.-F. Liao. On Utility-Fair Adaptive Services in Wireless Networks. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service (IWQoS'98)*, Napa Valley, USA, May 18-20 1998.
- R. R.-F. Liao, P. Bocheck, A. T. Campbell, and S.-F. Chang. Utility-based Network Adaptation for MPEG-4 Systems. In *Proc. of Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSS-DAV'99)*, Basking Ridge, New Jersey, USA, June 1999.
- O. Angin, A. T. Campbell, M. E. Kounavis, and R. R.-F. Liao. Open Programmable Mobile Networks. In *Proc. of Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'98)*, Cambridge, England, July 1998.
- R. R.-F. Liao, P. Bouklee, and A. T. Campbell. Online Generation of Bandwidth Utility Function for Digital Video. In *Proc. of PacketVideo'99*, New York City, April 26-27 1999.
- R. R.-F. Liao, M. Brown, G. Mapp, and I. Wassell. The Cambridge Wireless Broadband Trial. In *Proc. of 6th Intl. Workshop on Mobile Multimedia Communications (MoMuC'99)*, San Diego, November 1999.
- R. R.-F. Liao, M. E. Kounavis, and A. T. Campbell. Design, Implementation and Evaluation of Mobeware. In *Proc. of 5th Intl. Workshop on Mobile Multimedia Communications (MoMuC'98)*, Berlin, Germany, October 12-16 1998.

References

- [1] ISO/IEC 14496-1. Information Technology - Coding of Audio-visual Objects, Part 1: Systems, December 1998. ISO/IEC JT1/SC 29/WG 11 Draft International Standard.
- [2] 3GPP TS 03.60 v7.8.0. General Packet Radio Service (GPRS) Service Description, Stage 2, January 2002. <ftp://ftp.3gpp.org/specs/latest/>.
- [3] 3GPP2 P.S0001-A Version 1.0.0. Wireless IP Network Standard, July 2000. http://www.3gpp2.org/Public_html/specs/index.cfm.
- [4] C. Albuquerque, B. J. Vickers, and T. Suda. Network Border Patrol. In *Proc. IEEE INFOCOM*, Tel Aviv, Israel, March 2000.
- [5] J. Altmann and K. Chu. A Proposal for a Flexible Service Plan that is Attractive to Users and Internet Service Providers. In *Proc. IEEE INFOCOM*, Alaska, USA, April 2001.
- [6] O. Angin, A. T. Campbell, M. E. Kounavis, and R. R.-F. Liao. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. *IEEE Pers. Commun.*, August 1998. source code freely available at <http://comet.columbia.edu/mobiware>.
- [7] AF-TM-0121.000 ATM Forum. The ATM Forum Traffic Management Specification Version 4.1, March 1999. <http://www.atmforum.com/standards>.

- [8] M. Barry, A.T. Campbell, and A. Veres. Distributed Control Algorithm for Service Differentiation in Wireless Packet Networks. In *Proc. IEEE INFOCOM*, 2001.
- [9] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [10] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1992.
- [11] V. Bharghavan, K.-W. Lee, S. Lu, S. Ha, J.-R. Li, and D. Dwyer. The TIMELY Adaptive Resource Management Architecture. *IEEE Pers. Commun.*, August 1998.
- [12] G. Bianchi, A. T. Campbell, and R. R.-F. Liao. On Utility-Fair Adaptive Services in Wireless Networks. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Napa Valley, CA, May 18-20 1998.
- [13] F. Black and M. Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.
- [14] P. Bocheck and S.-F. Chang. Content Based Dynamic Resource Allocation for VBR Video in Bandwidth Limited Networks. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Napa Valley, CA, May 18-20 1998.
- [15] P. Bocheck, Y. Nakajima, and S.-F. Chang. Real-time Prediction of Subjective Utility Functions for MPEG-4 Video Objects. In *Proc. of PacketVideo'99*, New York City, April 26-27 1999.
- [16] E. Bouillet, D. Mitra, and K. G. Ramakrishnan. Design-Assisted, Real Time, Measurement-Based Network Controls for Management of Service Level

- Agreements. In *Proceedings of EURANDOM Workshop on Stochastics of Integrated-Services Comm. Networks*, Eindhoven, The Netherlands, November 15-19 1999.
- [17] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint Admission Control: Architectural Issues and Performance. In *Proc. ACM SIGCOMM*, Stockholm, Sweden, September 2000.
 - [18] L. Breslau and S. Shenker. Best-effort versus Reservations: a Simple Comparative Analysis. In *Proc. ACM SIGCOMM*, September 1998.
 - [19] S.L. Campbell and C.D. Meyer, Jr. *Generalized Inverses of Linear Transformations*. Pitman, London, UK, 1979.
 - [20] Z. Cao and E.W. Zegura. Utility max-min: An application-oriented bandwidth allocation scheme. In *Proc. IEEE INFOCOM*, March 1999.
 - [21] C. Cetinkaya and E. Knightly. Egress Admission Control. In *Proc. IEEE INFOCOM*, Tel Aviv, Israel, March 2000.
 - [22] A. Charny, D. D. Clark, and R. Jain. Congestion Control With Explicit Rate Indication. In *Proc. IEEE Int'l Conf. Commun.*, June 1995.
 - [23] A. Charny and K. K. Ramakrishnan. Time Scale Analysis of Explicit Rate Allocation in ATM Networks. In *Proc. IEEE INFOCOM*, April 1996.
 - [24] S. Choi and K. G. Shin. A Comparative Study of Bandwidth Reservation and Admission Control Schemes in QOS-Sensitive Cellular Networks. *ACM Baltzer J. Wireless Networks (WINET)*, 6(4):289–305, 2000.
 - [25] D. D. Clark, J. Wroclawski, K. Sollins, and R. Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. In *Proc. ACM SIGCOMM*, 2002.

- [26] Ron Cocchi, Scott Shenker, Deborah Estrin, and Lixia Zhang. Pricing in Computer Networks: Motivation, Formulation, and Example. *IEEE/ACM Trans. Networking*, 1(6):614–627, 1993.
- [27] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. In *Proc. ACM SIGCOMM*, September 1999.
- [28] K. E. Drexler and M. S. Miller. Incentive engineering for computational resource management. In Bernardo Huberman, editor, *The Ecology of Computation*. Elsevier Science Publishers/North-Holland, 1988.
- [29] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. A Flexible Model for Resource Management in Virtual Private Networks. In *Proc. ACM SIGCOMM*, September 1999.
- [30] N. G. Duffield and M. Grossglauser. Trajectory Sampling for Direct Traffic Observation. In *Proc. ACM SIGCOMM*, September 2000.
- [31] R. Braden, editor. (RSVP) – Version 1 Functional Specification. IETF RFC 2205, September 1997. <http://www.ietf.org/rfc/rfc2205.txt>.
- [32] A. Eleftheriadis and D. Anastassiou. Dynamic rate shaping of compressed digital video. In *Proc. of 2nd IEEE Intl. Conf. on Image Processing*, Arlington, VA, October 1995.
- [33] H. Schulzrinne et. al. Real time streaming protocol (rtsp). IETF RFC 2326, April 1998. <http://www.ietf.org/rfc/rfc2326.txt>.
- [34] J. Rosenberg et. al. Sip: Session initiation protocol. IETF RFC 3261, June 2002. <http://www.ietf.org/rfc/rfc3261.txt>.

- [35] S. Black et. al. An architecture for differentiated services. IETF RFC 2475, December 1998. <http://www.ietf.org/rfc/rfc2475.txt>.
- [36] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving Traffic Demands for Operational IP Networks: Methodology and Expeience. In *Proc. ACM SIGCOMM*, September 2000.
- [37] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. NetScope: Traffic Engineering for IP Networks. *IEEE Network Mag.*, March/April 2000.
- [38] D. Ferrari and L. Delgrossi. Charging for QOS. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Napa Valley, CA, May 18-20 1998. Keynote paper.
- [39] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Networking*, 1(4):397–413, August 1993.
- [40] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Trans. Networking*, 3(4):365–386, August 1995.
- [41] A. Fox, S. D. Gribble, Y. Chawathe, and E. A. Brewer. Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives. *IEEE Pers. Commun.*, August 1998.
- [42] E. Fulp and D. Reeves. Qos rewards and risk: a multi-market approach to resource allocation. In *Proc. IFIP TC6 Networking 2000 Conference*, Paris, France, 2000.
- [43] D. Fundenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, Mass., 1991.

- [44] R. J. Gibbens and F. P. Kelly. Distributed Connection Acceptance Control for a Connectionless Network. In *Proc. IEE Int'l Teletraffic Congress*, Edinburgh, UK, June 1999. Elsevier Science Publishers B.V.
- [45] R. Gibbons. *Game Theory for Applied Economists*. Princeton University Press, Princeton, NJ, 1992.
- [46] G. Hardin. The Tragedy of the Commons. *Science*, 162:1243–1248, 1968.
- [47] Y. T. Hou, H. Tzeng, and S. S. Panwar. A Generalized Max-Min Rate Allocation Policy and Its Distributed Implementation Using the ABR Flow Control Mechanism. In *Proc. IEEE INFOCOM*, San Francisco, CA, March 1998.
- [48] P. Hurley, M. Kara, J.-Y. Le Boudec, and P. Thiran. A Novel Scheduler for a Low Delay Service within Best-Effort. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Karlsruhe, Germany, June 2001.
- [49] P. Hurley, M. Kara, J.-Y. Le Boudec, and P. Thiran. ABE: Providing a Low-Delay Service within Best-Effort. *IEEE Network Mag.*, May/June 2001.
- [50] IEEE P802.11. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, D2.0, November 1997.
- [51] Recommendation ITU-R BT.500-7. Methodology for the Subjective Assessment of the Quality of Television Picture, October 1999. ITU-R Recommendations.
- [52] J. Jaffe. Bottleneck Flow Control. *IEEE Trans. Commun.*, COM-29(7):954–962, July 1981.
- [53] R. Jain. Congestion Control and Traffic Management in ATM Networks:

- Recent Advances and A Survey. *J. Computer Networks and ISDN Systems*, 28(13):1723–1738, November 1996.
- [54] R. Jain, T. Mullen, and R. Hausman. Analysis of Paris Metro Pricing Strategy for QOS with a Single Service Provider. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Karlsruhe, Germany, June 2001.
- [55] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. An Efficient Rate Allocation Algorithm for ATM Networks Providing Max-min Fairness. In *IFIP HPN'95*, Spain, September 11-16 1995.
- [56] R. H. Katz. Adaptation and Mobility in Wireless Information Systems. *IEEE Pers. Commun.*, 1(1), First Quarter 1994.
- [57] F. P. Kelly. Routing in Circuit-switched Networks: Optimization, Shadow Price and Decentralization. *Adv. Appl. Prob.*, 20:112–144, 1988.
- [58] F. P. Kelly. Charging and rate control for elastic traffic. *European Trans. Telecommunications*, 8:33–37, 1997.
- [59] F. P. Kelly, P. B. Key, and S. Zachary. Distributed Admission Control. *IEEE J. Select. Areas Commun.*, 18(12):2617–2628, December 2000. Special Issue on QOS in the Internet.
- [60] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.
- [61] S. Keshav. *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, Reading, Mass., 1997.

- [62] H. W. Kuhn and A. W. Tucker. Non-linear Programming. In *Proc. 2nd Berkeley Symp. on Mathematical Statistics and Probability*, pages 481–492. Univ. Calif. Press, 1951.
- [63] S. Kunniyur and R. Srikant. End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks. In *Proc. IEEE INFOCOM*, Tel Aviv, Israel, March 2000.
- [64] A. Kuznetsov. Linux Traffic Control (TC). <http://www.sparre.dk/pub/linux/tc>.
- [65] C. Lambrecht and O. Verscheure. Perceptual Quality Measure Using a Spatio-Temporal Model of the Human Visual System. In *Proc. of IS&T/SPIE*, San Jose, CA, February 1996.
- [66] C. Lee, J. P. Lehoczky, R. Rajkumar, and D. Siewiorek. On Quality of Service Optimization with Discrete QOS Options. In *Proc. IEEE Real-time Technology and Applications Symposium*, June 1999.
- [67] K. Lee. Adaptive Network Support for Mobile Multimedia. In *Proc. ACM MOBICOM*, Berkeley, CA, November 1995.
- [68] B. Li and K. Nahrstedt. A Control-based Middleware Framework for Quality of Service Adaptations. *IEEE J. Select. Areas Commun.*, 17(9):1632–1650, September 1999.
- [69] R. R.-F. Liao and A. T. Campbell. On Programmable Universal Mobile Channels in a Cellular Internet. In *Proc. ACM MOBICOM*, Dallas, TX, October 1998.

- [70] R. R.-F. Liao and A. T. Campbell. Dynamic Edge Provisioning for Core Networks. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Pittsburgh, USA, June 2000.
- [71] J. Liebeherr and N. Christin. JoBS: Joint Buffer Management and Scheduling for Differentiated Services. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Karlsruhe, Germany, June 2001.
- [72] S. H. Low, F. Paganini, and J. C. Doyle. Internet Congestion Control: An Analytical Perspective. *to appear in IEEE Control Systems Magazine*, December 2001.
- [73] S. Lu and V. Bharghavan. Adaptive Resource Management Algorithms for Indoor Mobile Computing Environments. In *Proc. ACM SIGCOMM*, September 1996.
- [74] J. K. MacKie-Mason and H. R. Varian. Pricing congestible network resources. *IEEE Journal on Selected Areas in Communications*, 13(7):1141–1149, September 1995.
- [75] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM Comput. Commun. Review*, 27, 1997.
- [76] D. Mitra, J.A. Morrison, and K. G. Ramakrishnan. Virtual Private Networks: Joint Resource Allocation and Routing Design. In *Proc. IEEE INFOCOM*, New York City, March 1999.
- [77] M. Naghshineh and M. Willebeek-LeMair. End-to-End QOS Provisioning in Multimedia Wireless/Mobile Networks Using an Adaptive Framework. *IEEE Commun. Mag.*, 35(11):72–81, November 1997.

- [78] J.F. Nash. The Bargaining Problem. *Econometrica*, 18(2):155–162, April 1950.
- [79] B. D. Noble, M. Stayanarayanan, and etc. Agile Application-aware Adaptation for Mobility. In *Proc. ACM Symposium on Operating System Principles*, St. Malo, France, October 1997.
- [80] A. M. Odlyzko. Paris Metro Pricing for the Internet. In *Proc. ACM Conference on Electronic Commerce (EC'99)*, pages 140–147, 1999.
- [81] A. M. Odlyzko. Internet Pricing and the History of Communications. *Computer Networks*, 36:493–517, 2001.
- [82] A. Ortega and K. Ramchandran. Rate-distortion Methods for Image and Video Compression. *IEEE Signal Processing Magazine*, 15(6):23–50, November 1998.
- [83] L. Qiu, Y. Zhang, and S. Keshav. On individual and aggregate tcp performance. In *Proc. Int'l Conf. Network Protocols*, Toronto, Canada, November 1999.
- [84] R. Rajan, D. Verma, S. Kamat, E. Felstaine, and S. Herzog. A Policy Framework for Integrated and Differentiated Services in the Internet. *IEEE Network Mag.*, pages 36–41, September/October 1999.
- [85] R. Rajkumar, C. Lee, J. P. Lehoczký, and D. Siewiorek. Practical Solutions for QOS-based Resource Allocation Problems. In *Proc. 19th IEEE Real-time Systems Symposium*, December 1998.
- [86] S. J. Rassenti, V. L. Smith, and B. J. Wilson. Turning off the Lights. *Regulation: The Cato Review of Business and Govern-*

- ment*, 24(3):70–76, Fall 2001. Special Report: The California Crisis, <http://www.cato.org/pubs/regulation/regv24n3/specialreport2.pdf>.
- [87] D. Reininger. *Dynamic Quality-of-Service Framework for Video in Broadband Networks*. PhD thesis, Rutgers University, New Jersey, January 1998.
- [88] R. Rejaie, M. Handley, and D. Estrin. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proc. ACM SIGCOMM*, September 1999.
- [89] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.
- [90] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching (mpls) architecture. IETF RFC 3031, January 2001.
- [91] N. Semret and A. A. Lazar. Spot and derivative markets in admission control. In *Proc. IEE Int'l Teletraffic Congress*, Edinburgh, UK, June 1999. Elsevier Science Publishers B.V.
- [92] C. E. Shannon. Coding Theorems for a Discrete Source with a Fidelity Criterion. *IRE Nat. Conv. Rec.*, 4:142–163, 1959. Reprinted in D. Slepian (ed.), *Key Papers in the Development of Information Theory*, IEEE Press, 1974.
- [93] S. Shenker. Fundamental Design Issues for the Future Internet. *IEEE J. Select. Areas Commun.*, 13(7):1176–1188, September 1995.
- [94] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in Computer Networks: Reshaping the Research Agenda. *ACM Comput. Commun. Review*, 26(2):19–43, 1996.

- [95] I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks. In *Proc. ACM SIGCOMM*, September 1998.
- [96] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proc. ACM SIGCOMM*, September 1999.
- [97] Elsevier Advanced Technology. Green Energy. *REFOCUS: The International Renewable Energy Magazine*, Sep.-Oct. 2000. http://www.refocus.net/so2000_4.html.
- [98] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, V. Narayan, and F. Reichmeyer. Internet2 QBone: Building a Testbed for Differentiated Services. *IEEE Network Mag.*, pages 8–16, September/October 1999.
- [99] UCB/LBNL/VINT. Network Simulator - ns. www.isi.edu/nsnam/ns/.
- [100] UCB/LBNL/VINT. Network Simulator - ns, DiffServ Module. www.isi.edu/nsnam/ns/ns-contributed.html.
- [101] A. G. Valko. Cellular IP - A New Approach to Internet Host Mobility. *ACM Comput. Commun. Review*, January 1999.
- [102] B. Vandalore, S. Fahmy, R. Jain, R. Goyal, and M. Goyal. A Definition of General Weighted Fairness and its Support in Explicit Rate Switch Algorithms. In *Proc. Int'l Conf. Network Protocols*, Austin, TX, October 1998.
- [103] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, (16):8–37, 1961.

- [104] X. Wang and H. Schulzrinne. Pricing Network Resources for Adaptive Applications in a Differentiated Services Network. In *Proc. IEEE INFOCOM*, Alaska, USA, April 2001.
- [105] Z. Wang. A case for proportional fair sharing. In *Proc. IEEE/IFIP Int'l Workshop on Quality of Service*, Napa Valley, CA, May 18-20 1998.
- [106] A.A. Webster, C.T. Jones, M.H. Pinson, S.D. Voran, and S. Wolf. An Objective Video Quality Assessment System Based on Human Perception. In *SPIE Human Vision, Visual Processing, and Digital Display IV*, volume 1913, February 1993.
- [107] W. E. Willinger, W.E. Leland, M.S. Taqqu, and D.V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Networking*, 2(1):1–15, February 1994.
- [108] R. Wilson. Efficient and competitive rationing. *Econometrica*, 57(1):1–40, January 1989.
- [109] H. Yaïche, R.R. Mazumdar, and C. Rosenberg. A Game Theoretic Framework for Bandwidth Allocation and Pricing in Braodband Networks. *IEEE/ACM Trans. Networking*, 8(5):667–678, October 2000.
- [110] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd. Filters: QOS Support Mechanisms for Multipeer Communications. *IEEE J. Select. Areas Commun.*, 14(7):1245–1262, September 1996. Special Issue on Distributed Multimedia Systems and Technology.
- [111] Z.L. Zhang, Z.H. Duan, L.X. Gao, and Y.W. Hou. Decoupling QOS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services. In *Proc. ACM SIGCOMM*, September 2000.