Wireless Networks 7, 541–557, 2001

A Utility-Based Approach for Quantitative Adaptation in Wireless Packet Networks

RAYMOND R.-F. LIAO and ANDREW T. CAMPBELL
Department of Electrical Engineering, Columbia University, USA

Abstract. This paper assesses the state-of-the-art in Quality-of-Service (QoS) adaptive wireless networks and proposes new adaptation techniques that better suit application specific needs. The contribution of the paper is as follows: we propose an adaptive service comprising (i) *bandwidth utility functions,* which capture the adaptive nature of mobile applications in terms of the range of bandwidth over which they prefer to operate; and (ii) *adaptation scripts,* which enable adaptive mobile applications to program the per-flow adaptation time scale and bandwidth granularity realizing application-specific adaptive services. To maintain adaptive services in wireless packet access networks, we propose a split level adaptation control framework that operates at the network and application levels. Network level control employs a periodic probing mechanism between mobile devices and network gateways in support of *utility based max–min fair* resource allocation. Application level control is managed by a set of distributed adaptation handlers that operate at mobile devices realizing application-specific adaptation strategies.

Keywords: adaptive wireless service, bandwidth utility function, utility-fair resource management

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. First, physical layer impairment (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 Quality-of-Service (QoS) guarantees unlikely. Next, user mobility can trigger rapid degradation in delivered service quality (e.g., during handoff). Third, wireless networks are typically bandwidth constrained in comparison to wireline networks. These system characteristics result in the delivery of time-varying Quality-of-Service to mobile applications. In many cases, mobile applications are not designed to operate successfully under such conditions. We observe that an application's ability to adapt to network changes, while keeping the user's perceptible quality meaningful, is very much application-specific. In this paper, 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 third generation 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 with end systems during adaptation resulting in poor scalability when the number of flows or traffic aggregates[1] grows or adaptation becomes frequent as in the case of wireless/mobile environments.

Recently, a number of adaptive mobile networking proposals have begun to address some of these issues [1–8]. However, 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 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. Second, it is challenging to develop quantitative metrics capable of explicitly representing adaptation needs because the wide variety of application-specific adaptation behavior complicates the definition of a single unified metric. As a result, existing quantitative models (e.g., [9–11]) are coarse and not unified. In some cases, the adaptation metric is directly related to user perception (e.g., by means of subjective testing such as the 5-level Mean Opinion Score measure for video quality [12]). In other cases, objective measurements (e.g., signal-to-quantization noise ratio in adaptive MPEG coding) are more effective.

Finally, network control needs to be extended in an efficient manner to support common adaptation requirements, which can be characterized as having two dimensions; that

[1] The terms flow and traffic aggregate are used synonymously in this paper. Both refer to packets satisfying the same packet header classification rule and sharing the same routing path in a wireless packet access network.

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 [13] 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 [14]. A new challenge is to extend max–min fairness to support adaptation in an efficient and scalable manner.

In this paper, 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 paper is as follows. In section 2 we discuss related work in the area of adaptive resource control in wireless networks. Following this, in section 3, we describe a utility-based adaptation framework for wireless packet access networks. In section 4, we introduce utility functions and their basic operations. In section 5, we present a detailed description of our *utility-based network control* algorithm that realizes utility-based max–min fairness. Following this, in section 6, 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 7, 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 paper in section 8 and present the pseudo-code for the utility-fair max–min algorithm in the appendix.

2. Related work

There has been a number of architectural proposals for adaptive services in mobile networks [1,2,4]. In [3], Lu and Bharghavan present a set of admission control and reservation mechanisms that extend generic resource renegotiation to wireless and mobile environments. In [2], utility functions are proposed for network resource management. However, there is no discussion concerning specific mechanisms. Our previous work on network resource allocation [6–8] differs from the work presented in [2] and [3] because our research is motivated by the need to support multiple bandwidth allocation strategies and utility-based resource allocation in wireless networks. In [6], explicit support for the delivery of multi-resolution flows is built into the network and transport layers of the Mobiware system. In this case, wireless access points support the transport of base and enhancement layers, and media scaling and packet discarding techniques. Mobiware supports per-mobile adaptive resources control that periodically probes the wireless access network for bandwidth availability over slow time scales (i.e., in the order of seconds). In [7], we propose a programmable MAC middleware architecture to support QoS adaptation at the data link layer. The scheme manages service weights of a channel-state dependent scheduler [15,16] for both the uplink and downlink with respect to per-flow utility functions. In this paper, we extend this work to the network and transport layers by taking into consideration wireless access networks.

The use of utility functions has been widely cited in the literature as a means of capturing application-specific behavior in adaptive networking environments (e.g., Internet [17,18], mobile networks [2] and ATM networks [9]). The Q-RAM project [10] proposes a utility-based mechanism that allocates resources to operating system processes with the aim of maximizing the total system utility. Unlike utility-maximizing allocation algorithms, which maximize the global utility, our previous work [7] exploits "utility-fair allocation". This formulation has a simple closed-form solution and is consistent under aggregation, which allows the network bandwidth allocation mechanisms to operate on aggregated traffic promoting a high degree of scalability.

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 [19,20]). In this paper, we formalize the utility-based max–min fair allocation proposed in [8] and investigate system issues associated with protocol design and adaptation policy, including algorithm and protocol scalability to support aggregated flow states.

Our approach to realizing adaptation policies differs from end-system oriented approaches (e.g., the Odyssey Project [5] and the adaptation proxy work discussed in [21]). We introduce a utility-based adaptation framework that provides a generic approach to adaptation allowing a wide range of application specific policies to be implemented. For example, a "smooth" adaptation policy can have the same effect as the end system playout-control mechanism discussed

in [22]. 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.

In [23], 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 realtime 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.

3. Utility-based adaptation framework

Figure 1 illustrates our utility-based adaptation framework for wireless packet 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.

3.1. Customizing adaptive services

3.1.1. Bandwidth utility functions

Utility functions can represent a wide variety of application adaptation behavior, as illustrated in figure 2. 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 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 applications that

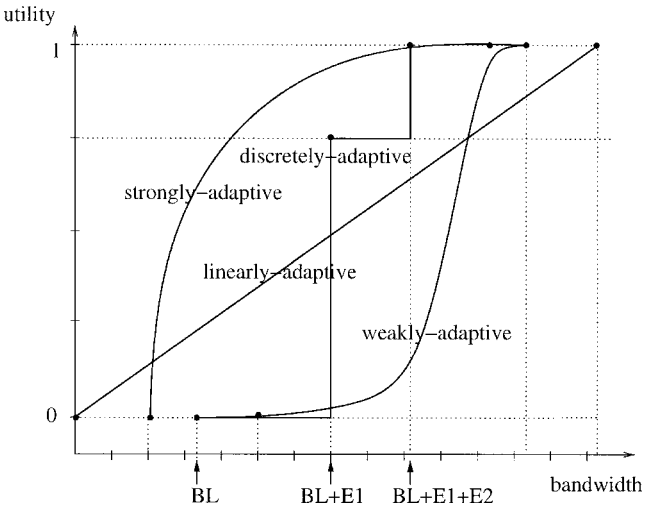

Figure 2. Different styles of utility functions.

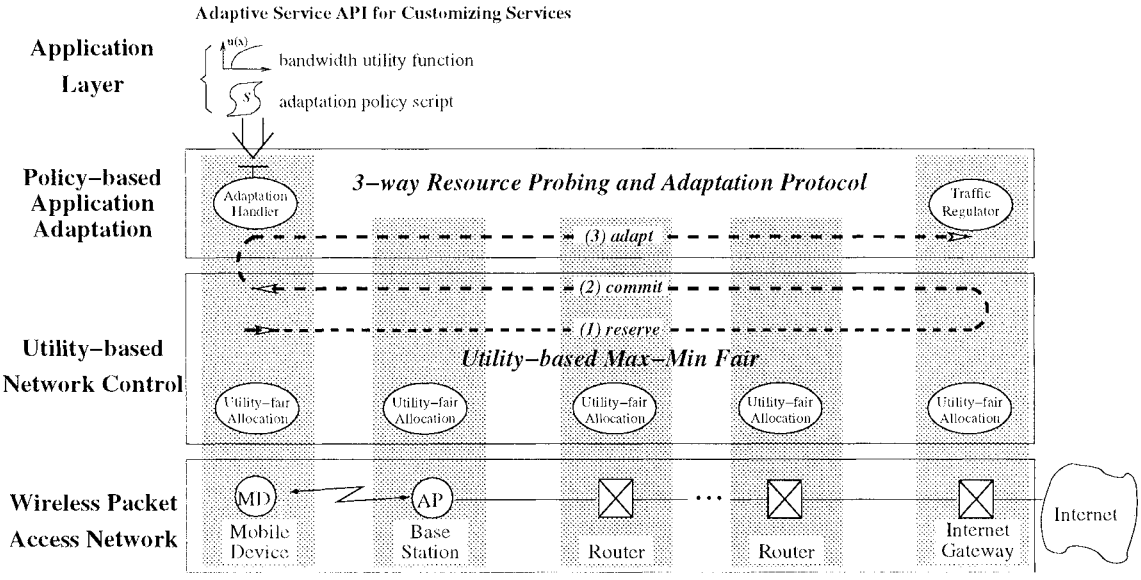

Figure 1. Utility-based adaptation framework.

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). In our framework, utility functions are defined by service providers as part of a customer's service plan. Applications specify their bandwidth needs by choosing a service specific form of utility function and modifying customizable parameters of the utility function (e.g., the maximum bandwidth value, or the bandwidth values at the break points of a discrete utility curve).

3.1.2. Adaptation scripts

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 scales and bandwidth granularities; 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 6.

3.2. Utility-based network control

We assume a two-tier network model, where the global Internet provides interconnectivity to a set of wireless packet access networks through gateways, as shown in figure 1. An example of the wireless packet access network is a Cellular IP [24] 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 1. The first two phases called *reserve* (1) and *commit* (2) are part of the network control scheme that performs utility-based max–min fair bandwidth allocation (discussed in section 5.2) 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 [25] 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 timeout 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 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 [7].

3.3. Policy-based application adaptation

As shown in figure 1, we design an additional control layer on top of the utility-based network control. 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.

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 in figure 3.

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 3, the consumed bandwidth can be less than or equal to the allocated bandwidth

(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 3. Simple adaptation script schema.

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.

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. Bandwidth utility functions

For simplicity and system scalability, we propose to use bandwidth utility functions to support a per-service-class model[2] that quantitatively describes the trajectory of service degradation in relation to time-varying bandwidth conditions experienced by mobile devices in wireless networks.

4.1. Definition

We define a bandwidth utility function $u(x)$ as the mapping of the transmission bit rate, referred to as available network bandwidth x, into a "utility" (i.e., quality) value that represents the level of service quality satisfaction perceived by an application. Typically, an increase in available bandwidth does not decrease the application quality, making the utility function a non-decreasing function of the bandwidth x.

For ease of implementation, it is necessary to quantize utility functions using a small set of parameters. Since the utility value is used to model relative preference, its value has only relative significance. Therefore, without loss of generality, we can normalize the utility values of all curves in the range of $[0, K]$. The quantization levels are the K numbers of *critical utility levels* that divide the utility axis range $[0, K]$ into K equal intervals. Following on from this, for flow i, its *critical bandwidth value,* $b_{i,k}$, can be defined as $u_i(b_{i,k}) = k$, $k \in \mathcal{N}_K$, where $\mathcal{N}_K$ denotes the natural number set $\{1, 2, \ldots, K\}$.

Utility functions are an extremely flexible tool that allow adaptive applications to share the same resources. For this reason, we do not give any specific meaning to the utility value, other than the utility range. The mapping between utility values in the range $[0, K]$ and the perceived level of service quality is a task left to each individual adaptive application. This mapping is dependent on application semantics and on the way in which quality is accounted for (e.g., MOS, SNR, etc.). For an example of this type of mapping for a video application see [11].

[2] This does not, however, preclude the use of bandwidth utility functions on a per-application or per-mobile basis.

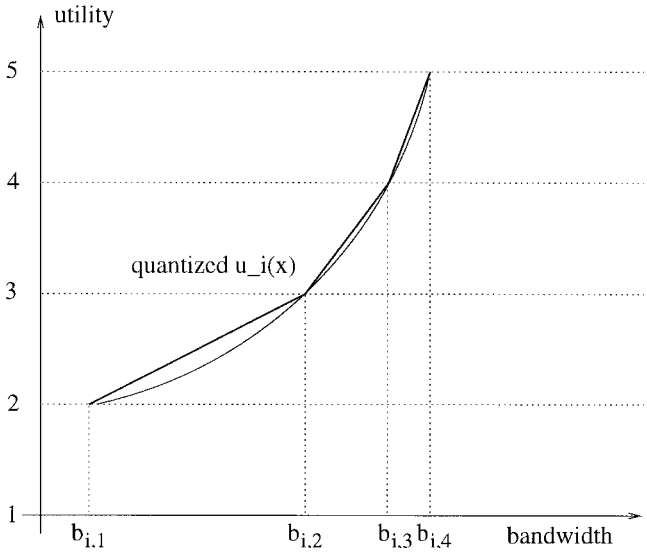

Figure 4. Quantized utility function.

We approximate utility functions after quantization using either a continuous piecewise linear shape, a discrete staircase shape, or a combination of the two. Figure 4 illustrates an example of a quantized piecewise linear utility function. However, in the following derivation of the bandwidth allocation formula, we only assume strictly increasing and continuous piecewise-linear utility functions to avoid the complexity of a "bin-packing problem", which could arise from resource partitioning using discrete granularity and does not have closed-form solution. Later in section 6.2, we extend our solution to support non-strictly-increasing or discrete utility functions by an iterative approach. The support for these type of utility functions is managed by adaptation handlers at mobile devices, where discrete adaptation handlers compete with each others for available residual bandwidth based on their discrete bandwidth requirements. This approximation does not introduce significant error because the error is bounded by the difference between critical bandwidth values, where the actual and approximated piecewise-linear curves coincide. In practice, the original utility function is normally obtained by coarse subjective testing [12] and presented as a piecewise linear shape with a small number of quality levels.

Consider a flow i. The strictly increasing and continuous piecewise linear utility function $u_i(x)$ has the general form of

$$u_i(x) = k + \frac{x - b_{i,k}}{b_{i,k+1} - b_{i,k}} \quad \forall x \in [b_{i,k}, b_{i,k+1}], \quad k \in \mathcal{N}_{K-1}. \tag{4.1}$$

Here $b_{i,k} < b_{i,k+1}$. The inverse of $u_i(x)$ has the form:

$$u_i^{-1}(y) = b_{i,k} + (y-k)(b_{i,k+1} - b_{i,k}) \quad \forall y \in [k, k+1]. \tag{4.2}$$

We observe that the critical utility value 1 is the lowest acceptable operating point, where $b_{i,1}$ ($b_{i,1} \geqslant 0$) is defined as the *minimum sustained rate* corresponding to the minimum bandwidth at which an application is able to successfully operate. In contrast, $b_{i,K}$ denotes the *peak rate* which corresponds to an application's optimal operating rate, and $u_i(b_{i,K}) = K$ represents the maximum satisfaction level. Since $\{b_{i,k} \mid i \in \mathcal{N}_K\}$ completely characterize a piecewise linear utility function with K critical utility levels, we use the critical bandwidth vector $\mathbf{u}_i \triangleq \langle b_{i,1}, b_{i,2}, \ldots, b_{i,K}\rangle$ to represent the piecewise linear utility function $u_i(x)$ in the remaining part of this paper.

Next, we define the aggregation of strictly-increasing and continuous piecewise linear utility functions as follows. A piecewise linear utility function $\mathbf{u}_3$ is the aggregation of piecewise linear utility functions $\mathbf{u}_1$ and $\mathbf{u}_2$, denote $\mathbf{u}_3 \triangleq \mathbf{u}_1 \oplus \mathbf{u}_2$, where $u_3^{-1}(\eta) = u_1^{-1}(\eta) + u_2^{-1}(\eta)\ \forall \eta \in [1, K]$. Because of the linearity of $u_i^{-1}(y)$ when $y \in [k, k+1)$, $\mathbf{u}_1 \oplus \mathbf{u}_2$ can be equivalently represented in vector form as

$$\mathbf{u}_1 \oplus \mathbf{u}_2 \Longleftrightarrow \langle b_{1,1} + b_{2,1}, \ldots, b_{1,K} + b_{2,K}\rangle.$$

Similarly, a piecewise linear utility function $\mathbf{u}_3$ is the segregation of $\mathbf{u}_2$ from $\mathbf{u}_1$ denote $\mathbf{u}_3 \triangleq \mathbf{u}_1 \oplus \mathbf{u}_2$, where $u_3^{-1}(\eta) = u_1^{-1}(\eta) - u_2^{-1}(\eta)\ \forall \eta \in [1, K]$ and $u_1^{-1}(\eta) \geqslant u_2^{-1}(\eta)$. In vector form, this is

$$\mathbf{u}_1 \ominus \mathbf{u}_2 \Longleftrightarrow \langle b_{1,1} - b_{2,1}, \ldots, b_{1,K} - b_{2,K}\rangle.$$

From the vector form, the utility function aggregation and segregation can be calculated with $O(K)$ complexity.

4.2. Utility-based fairness

Typically, bandwidth allocation mechanisms treat fairness with respect to bandwidth. In what follows, we define a new fairness criterion with respect to the utility value that corresponds to a flow's allocated bandwidth.

Definition 4.1. A bandwidth allocation rule for n flows characterized by utility functions $u_i(x)$, $i = 1, \ldots, n$, is utility-fair when the bandwidth allocation vector $\beta \triangleq \langle \beta_1, \ldots, \beta_i, \ldots, \beta_n\rangle$, where β_i the allocation to flow i, satisfies the following:

$$u_i(\beta_i) = u_j(\beta_j) \quad \forall i, j \in \mathcal{N},$$

and

$$\sum_{i=1}^{n} \beta_i = \min\left\{\mathcal{B}, \sum_{i=1}^{n} b_{i,K}\right\},$$

where $b_{i,K}$ is the peak rate requirement of flow i, and $\mathcal{B}$ is the total bandwidth available for all flows on the link.

Consider n adaptive flows described by strictly increasing piecewise linear utility functions $\mathbf{u}_j = \langle b_{j,1}, \ldots, b_{j,K}\rangle$, $j \in \mathcal{N}$. The aggregated utility function $\mathbf{u}_\oplus = \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \cdots \oplus \mathbf{u}_n = \langle B_1, \ldots, B_K\rangle$, where $B_k = \sum_{j=1}^{n} b_{j,k}$ results from the aggregation operation. Because of the admission control on the minimum sustained rate $b_{j,1}$, we may assume $\mathcal{B} > B_1$.

In what follows, we show that for strictly increasing piecewise linear utility functions, the utility-fair allocation has a closed-form solution.

Proposition 4.2. For strictly increasing piecewise linear utility functions, the *utility-fair* allocation rule is given by

$$\beta_i = \mathcal{F}(\mathbf{u}_i, \mathbf{u}_\oplus, \mathcal{B}) = \begin{cases} b_{i,K} & \text{if } \mathcal{B} \geqslant B_K, \\ b_{i,k} + \dfrac{\mathcal{B} - B_k}{B_{k+1} - B_k}(b_{i,k+1} - b_{i,k}) & \text{otherwise,} \end{cases} \tag{4.3}$$

where $k = \lfloor u_\oplus(\mathcal{B})\rfloor$ is the index such that $B_k \leqslant \mathcal{B} < B_{k+1}$, and

$$u_i(\beta_i) = u_\oplus(\mathcal{B}) = k + \frac{\mathcal{B} - B_k}{B_{k+1} - B_k} \quad \forall i \in \mathcal{N}_n, \tag{4.4}$$

namely, all flows are allocated the same utility value $\eta \triangleq u_\oplus(\mathcal{B})$.

Proof. We simply need to verify that the allocation given by equation (4.3) satisfies definition 4.1. Since $B_k \leqslant \mathcal{B} < B_{k+1}$, from equation (4.3) we have $b_{i,k} \leqslant \beta_i < b_{i,k+1}$. Then we can write down the utility value $u_i(\beta_i)$ from equation (4.1) as

$$u_i(\beta_i) = k + \frac{\beta_i - b_{i,k}}{b_{i,k+1} - b_{i,k}}.$$

After substituting equation (4.3) in, we have:

$$u_i(\beta_i) = k + \frac{\mathcal{B} - B_k}{B_{k+1} - B_k}\,\frac{b_{i,k+1} - b_{i,k}}{b_{i,k+1} - b_{i,k}} = k + \frac{\mathcal{B} - B_k}{B_{k+1} - B_k} = \eta.$$

Since the right-hand side of the above equation is not a function of i, it indicates that the same equation holds for all the values of i. Therefore, we have

$$u_i(\beta_i) = u_j(\beta_j) = \eta \quad \forall i, j \in \mathcal{N}_n,$$

which agrees with the definition of the utility-fair rule. □

In fact, from equation (4.4), another form of $\mathcal{F}(\mathbf{u}_i, \mathbf{u}_\oplus, \mathcal{B})$ is given by

$$\mathcal{F}(\mathbf{u}_i, \mathbf{u}_\oplus, \mathcal{B}) = u_i^{-1}(\eta) = u_i^{-1}\big(u_\oplus(\mathcal{B})\big), \quad i \in \mathcal{N}_n. \tag{4.5}$$

An example of the bandwidth allocation algorithm that follows this rule is illustrated in figure 5. Our implementation of the allocation algorithm consists of three steps:

(1) update the aggregated utility function $\mathbf{u}_\oplus$ with respect to the change for flow i (with complexity $O(K)$); then,

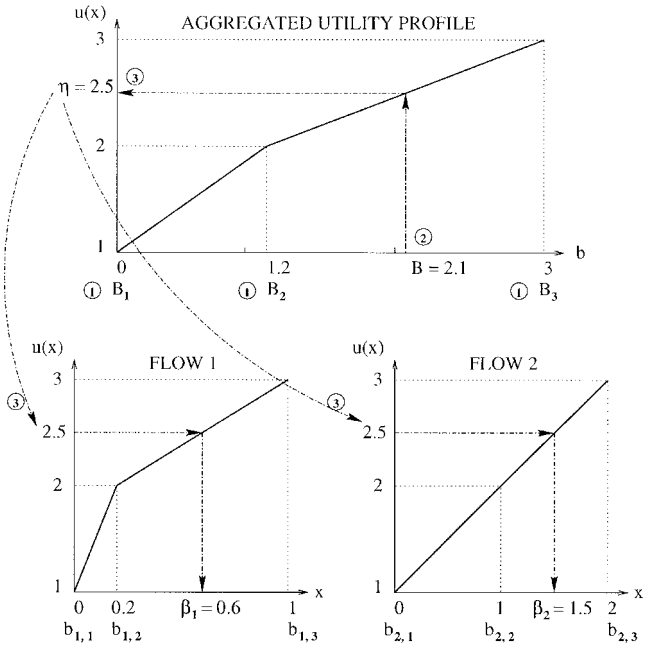

Figure 5. Graphical example of the utility-fair allocation rule. In the example, $K = 3$ and $\mathcal{B} = 2.1$. Results: $\eta = 2.5$, $\beta_1 = 0.6$, $\beta_2 = 1.5$.

(2) locate the critical utility level k so that the total available bandwidth $\mathcal{B}$ fits in (this constitutes a binary search with complexity $O(\log K)$); then,

(3) apply equation (4.3) to calculate allocation β_i (with complexity $O(1)$).

The total complexity to update a flow's bandwidth allocation is of the order of $O(K)$. The simplicity of the utility-based fair allocation algorithm is a consequence of our optimization goal; that is, to maximize bandwidth utilization under the constraint of utility-based fairness rather than to maximize global utility as applied by existing network economic literature (e.g., see [9]). An important property of the utility-based fair allocation rule is the allocation consistency under flow aggregations; that is:

Proposition 4.3. For any set of piecewise linear utility functions $\{\mathbf{u}_j\}$, under the same available bandwidth $\mathcal{B}$, the utility-fair allocation for each flow is not affected by any order and combination of utility function aggregations over $\{\mathbf{u}_j\}$.

Proof. We start with a set of three piecewise linear utility functions $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$. For flow #1, the property indicates that first, flow #1's allocation is not affected by aggregation of other flows, which is to prove equation (4.6):

$$\mathcal{F}(\mathbf{u}_1, \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3, \mathcal{B}) = \mathcal{F}\big(\mathbf{u}_1, \mathbf{u}_1 \oplus (\mathbf{u}_2 \oplus \mathbf{u}_3), \mathcal{B}\big), \tag{4.6}$$

and second, flow #1's allocation is not affected by aggregation involving itself, which is to prove equation (4.7):

$$\mathcal{F}(\mathbf{u}_1, \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3, \mathcal{B}) = \mathcal{F}(\mathbf{u}_1, \mathbf{u}_1 \oplus \mathbf{u}_2, X), \tag{4.7}$$

where

$$X \triangleq \mathcal{F}\big((\mathbf{u}_1 \oplus \mathbf{u}_2), (\mathbf{u}_1 \oplus \mathbf{u}_2) \oplus \mathbf{u}_3, \mathcal{B}\big).$$

The proof of equation (4.6) directly follows from the associativity of operator $\oplus$, namely $\mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3 = \mathbf{u}_1 \oplus (\mathbf{u}_2 \oplus \mathbf{u}_3)$. To prove equation (4.7), we use equation (4.5) to rewrite the right-hand side of (4.7) as

$$\mathcal{F}(\mathbf{u}_1, \mathbf{u}_1 \oplus \mathbf{u}_2, X) = u_1^{-1}\big(\eta^*\big),$$

and from equation (4.4),

$$\eta^* = (\mathbf{u}_1 \oplus \mathbf{u}_2)(X). \tag{4.8}$$

Similarly, with equations (4.5) and (4.4), X can be simplified as

$$\begin{aligned} X &= \mathcal{F}\big((\mathbf{u}_1 \oplus \mathbf{u}_2), (\mathbf{u}_1 \oplus \mathbf{u}_2) \oplus \mathbf{u}_3, \mathcal{B}\big) \\ &= (u_1 \oplus u_2)^{-1}\big(\eta^{**}\big), \qquad (4.9) \\ \eta^{**} &= \big((u_1 \oplus u_2) \oplus u_3\big)(\mathcal{B}) \\ &= (u_1 \oplus u_2 \oplus u_3)(\mathcal{B}) \triangleq \eta. \qquad (4.10) \end{aligned}$$

Combining (4.9) and (4.10) into (4.8), we have

$$\eta^* = (u_1 \oplus u_2)\big((u_1 \oplus u_2)^{-1}(\eta)\big) = \eta.$$

Substituting the result into (4.8), we have

$$\mathcal{F}(\mathbf{u}_1, \mathbf{u}_1 \oplus \mathbf{u}_2, X) = u_1^{-1}(\eta) = \mathcal{F}(\mathbf{u}_1, \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3, \mathcal{B}).$$

This proves equation (4.7). The extension of the proof to more than three flows can be achieved by induction. □

This property implies that the allocation formula can be recursively applied to different levels of aggregated utility functions without distortion.

5. 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 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 3 and detailed in this section.

5.1. 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 5.1. A bandwidth allocation vector $\beta = \langle \beta_1, \ldots, \beta_n \rangle$ is *feasible* if for each flow $i \in \mathcal{N}_n$, $\beta_i \geqslant b_{i,1}$ and for each link l, $\sum_{\forall i \text{ passing link } l} \beta_i \leqslant \mathcal{B}^l$.

Definition 5.2. 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 $u_j(\beta_j) \leqslant u_i(\beta_i)$.

Definition 5.3. 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., $\sum_{\forall i \text{ passing link } l} \beta_i = \mathcal{B}^l$, and $u_i(\beta_i) \geqslant u_j(\beta_j)$ for all the flows j crossing l.

These definitions are similar to the max–min fairness definition in [14]. The main change is to substitute the comparison in bandwidth values (e.g., $\beta_i \geqslant \beta_j$) to a comparison in corresponding utility values (e.g., $u_i(\beta_i) \geqslant u_j(\beta_j)$). The utility-based max–min fairness also has the same properties as conventional max–min fairness.

Proposition 5.4. 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 [14] 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 5.5. There exists a unique allocation vector that satisfies utility-based max–min fair rate allocation.

The proof of proposition 5.5 is to first construct one allocation vector satisfying utility-based max–min fairness. One may use the centralized allocation algorithm in [14] 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 5.2.

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.

5.2. 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 [26,27]. In [26], 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

$$\beta_i = \frac{\mathcal{B}^l - \mathcal{B}^l_{\mathcal{L}}}{\sum_{j \in \mathcal{U}^l} 1} = \frac{\mathcal{B}^l - \mathcal{B}^l_{\mathcal{L}}}{|\mathcal{U}^l|}, \tag{5.1}$$

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

We propose a simple extension that replaces equation (5.1) with the following derived from (4.5):

$$\beta_i = \mathcal{F}\big(\mathbf{u}_i, \mathbf{u}^l_{\mathcal{U}}, \mathcal{B}^l - \mathcal{B}^l_{\mathcal{L}}\big) = u_i^{-1}\big(u^l_{\mathcal{U}}\big(\mathcal{B}^l - \mathcal{B}^l_{\mathcal{L}}\big)\big), \tag{5.2}$$

where $\mathbf{u}^l_{\mathcal{U}} \triangleq \bigoplus_{\forall i \in \mathcal{U}^l} \mathbf{u}_i$ is the aggregated utility function of $\mathcal{U}^l$, the set of bottlenecked flows. The composite function from the inverse utility function $u_i^{-1}(\cdot)$ and the aggregated utility function $u^l_{\mathcal{U}}(\cdot)$ captures the notion of weighted fairness, while the utility function has built-in support for the minimum sustained rate.

In [27], 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 $\mathbf{u}^l_{\mathcal{U}}$, i.e., $\mathbf{u}^l_{\mathcal{U}} = \mathbf{u}^l_{\mathcal{U}} \oplus \mathbf{u}_i$. Subsequently, an abstract view of the bandwidth assignment rule is given as follows:

$$\begin{aligned} \beta_i = \max\big\{&\mathcal{F}\big(\mathbf{u}_i, \mathbf{u}^l_{\mathcal{U}}, \mathcal{B}^l - \mathcal{B}^l_{\mathcal{L}}\big), \\ &\mathcal{F}\big(\mathbf{u}_i, \mathbf{u}^l_{\mathcal{U}} \oplus \mathbf{u}_k, \mathcal{B}^l - \mathcal{B}^l_{\mathcal{L}} + \beta_k\big)\big\}, \\ \text{where } k = \arg&\max_{\forall j \in \mathcal{L}^l}\big\{u_j(\beta_j)\big\}. \end{aligned} \tag{5.3}$$

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 [27]. 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.

5.3. 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 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 (4.5).

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 delay[3].

To reduce the convergence time of the probing scheme by half, we exploit the backward signalling 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 5.1), two parallel sets of reservation state $(\rho_i^{\text{ideal}}, r_{i,l}^{\text{ideal}})$ and $(\rho_i^{\text{actual}}, r_{i,l}^{\text{actual}})$ 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. $(\rho_i^{\text{actual}}, r_{i,l}^{\text{actual}})$ the actual allocation under the feasibility constraint, and $(\rho_i^{\text{ideal}}, r_{i,l}^{\text{ideal}})$ tracks the ideal allocation without the feasibility constraint[4] and ensures convergence to utility-based max–min allocation.

For a mobile traffic aggregate i, the states stored at each link l include $\mathbf{u}_i$, $r_{i,l}^{\text{ideal}}$, $r_{i,l}^{\text{actual}}$, and a flag $S_{i,l} \in$ {bottlenecked,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 $\mathcal{L}^l$: $\mathcal{B}^l_{\mathcal{L}}$, and the unused bandwidth $\mathcal{B}^l_{\text{free}}$. The algorithm also maintains state for an aggregated utility function $\mathbf{u}^l_{\mathcal{U}}$.

The allocation rule follows from equation (5.3). To reflect the two allocation algorithms operating in parallel, the ideal and actual allocations of a traffic aggregate are calculated based on the following:

$$\begin{aligned} &\text{ideal:} \quad r_{i,l}^{\text{ideal}} = \min\bigl\{\beta^l_{\text{alloc}}, \rho_i^{\text{ideal}}\bigr\}, \\ &\text{actual:} \quad r_{i,l}^{\text{actual}} = \min\bigl\{\beta^l_{\text{alloc}}, \rho_i^{\text{actual}}, \mathcal{B}^l_{\text{free}}\bigr\}. \end{aligned} \tag{5.4}$$

After the allocation steps given by (5.3) and (5.4) 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 [27] 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.

5.4. Convergence property

The analysis of our algorithm's convergence property follows closely from [28]. It is shown in [28] 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 signalling 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 5.6. 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 GSM 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 signalling traffic load on the system. A compromise scheme could use variable probing intervals on different portions of the network

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

[4] The 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.

(e.g., a smaller value across the wireless hop and larger one in the wireline access network) with the result of improving bandwidth usage.

6. 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 predefined, 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**$(j, \rho_j^{\text{ideal}}, \rho_j^{\text{actual}})$ function, where p_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 (4.5). Based on the adaptation script, the **commit**$(j, \rho_j^{\text{ideal}}, \rho_j^{\text{actual}})$ function determines the consumed bandwidth $\sum_{j \in \text{traffic aggregate}} \rho_j^{\text{ideal}}$ and $\sum_{j \in \text{traffic aggregate}} \rho_j^{\text{actual}}$ 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 quality of service 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.

```
commit(j, ρ_j^ideal, ρ_j^actual) {
   adapt(ρ_j^ideal, ρ_j^actual);
}
```

Figure 6. Simple greedy adaptation script.

6.1. 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**$(j, \rho_j^{\text{ideal}}, \rho_j^{\text{actual}})$ 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.

The choice of probing interval needs to balance the trade-off between a desired fairness behavior and the increase in signalling traffic that a short probing interval would bring. In section 7, we study the effect of the probing interval on allocation accuracy.

6.2. 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 7, 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). 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 7 can cause allocation disparity, as we will show in section 7. 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 better treatment than others.

```
Strategy A
commit(j, ρ_j^ideal, ρ_j^actual) {
   // locate discrete utility value
   m = ⌊u_j(ρ_j^ideal)⌋, n = ⌊u_j(ρ_j^actual)⌋;
   // assign discrete bandwidth
   ρ_j^ideal = b_{j,m}, ρ_j^actual = b_{j,n};
   adapt(ρ_j^ideal, ρ_j^actual);
}

Strategy B
commit(j, ρ_j^ideal, ρ_j^actual) {
   if ((ρ_j^actual == ρ_j^ideal) or (ρ_j^actual ⩾ prev_ρ_j^actual)) {
      // for ρ_j^ideal: discrete script
      m = ⌊u_j(ρ_j^ideal)⌋;
      ρ_j^ideal = b_{j,n};
   }
   prev_ρ_j^actual = ρ_j^actual;
   n = ⌊u_j(ρ_j^actual)⌋;
   ρ_j^actual = b_{j,n};
   adapt(ρ_j^ideal, ρ_j^actual);
}
```

Figure 7. Two discrete adaptation scripts.

```
commit(j, ρ_j^ideal, ρ_j^actual) {
   ρ_j^ideal = calc_bw(ρ_j^ideal, prev_β_j^ideal);
   ρ_j^actual = calc_bw(ρ_j^actual, prev_β_j^actual);
   if ((no increment) or
      (time_since_last_increment ⩾ τ)) {
      prev_β_j^ideal = ρ_j^ideal, prev_β_j^actual = ρ_j^actual;
      reset timer;
   } else // do not change
      ρ_j^ideal = prev_β_j^ideal, ρ_j^actual = prev_β_j^actual;
   adapt(ρ_j^ideal, ρ_j^actual);
}

calc_bw(ρ, prev_β) {
   if (prev_β == 0) { // first time
      return 0.5ρ;
   } else if (κρ ⩾ prev_β − δ) {
      // limit increment by δ and κ
      return min{κρ, prev_β + δ};
   } else {
      return min{ρ, prev_β − δ};
   }
}
```

Figure 8. Smooth adaptation script.

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 7 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.

6.3. 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 8.

6.4. 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 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 9 shows a number of possible outcomes for handoff adaptation illustrated in the experimental trace obtained from Mobiware [29], a programmable mobile networking platform that operates over an experimental indoor picocellular 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 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.

7. Simulation

We have presented four common adaptation scripts in section 6. 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 10. 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.

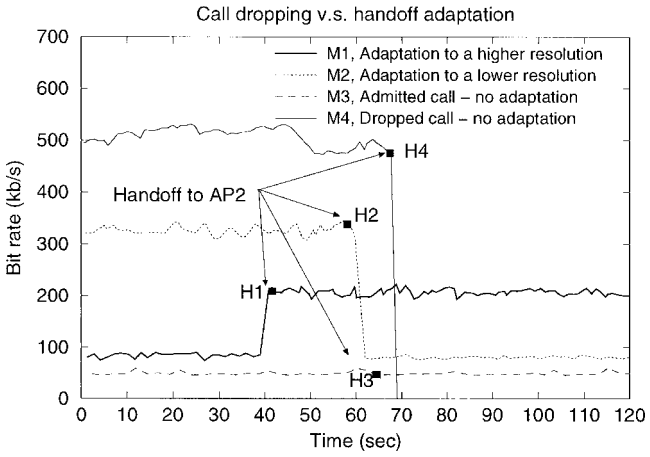

Figure 9. Handoff adaptation script results.

7.1. Simulation environment

In figure 10, 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.

7.1.1. Flow parameters

A total of ten flows with various adaptation scripts are simulated. For clarity, we assume one flow per traffic aggregate. Table 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 Mbps maximum bandwidth requirement ($B_{.,K}$), the others simulate the flows across wireless links with $B_{.,K}$ varying from 2 to 6 and 10 Mbps. Two types of utility functions are used during simulations, as shown in figure 11. 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 dif-

Table 1
Flow utility curve parameters.

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

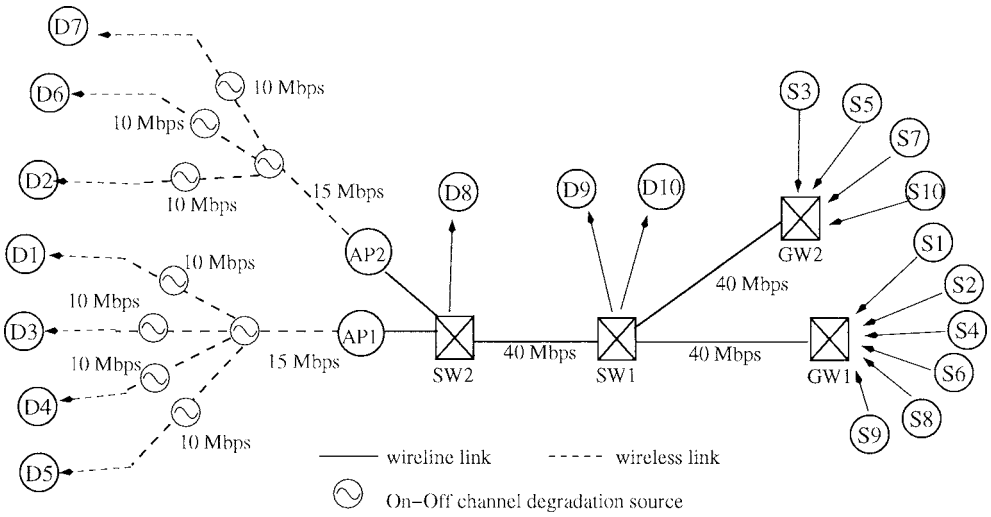

Figure 10. Simulated mobile access network topology.

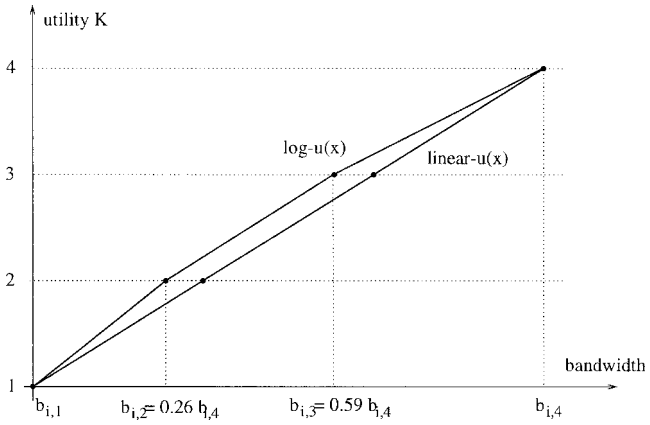

Figure 11. Utility curves used in simulations.

ferent $B_{\cdot,K}$. 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.

7.1.2. 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 s, one flow is established every second, followed by flow 6 terminating at 85 s, flow 4 terminating at 127 s, flow 1 handing off at 171 s and returning again at 204 s into the scenario. Under noise level 0, the network topology shown in figure 10 has three bottleneck links. The most congested bottleneck in this scenario is the link SW2 → AP1, followed by the link SW2 → AP2, and finally link GW1 → 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 10 as links SW2–AP1 and SW2–AP2) have a capacity of 15 Mbps, 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 Mbps 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 s, 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 Mbps channel capacity under ideal channel conditions. This capacity is large enough to prevent the links from becoming bottlenecks because 10 Mbps 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.

7.2. 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 5.3.

The fairness metric of the distributed scheme is calculated using the "fairness index" proposed in [30]. 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

$$FI(t_i) = \frac{(\sum_{j=1}^{N} \gamma_j(t_i))^2}{N \sum_{j=1}^{N} \gamma_j(t_i)^2} \quad \text{and} \quad \gamma_j(t_i) = \frac{\beta_j(t_i)}{\widetilde{\beta}_j(t_i)}, \tag{7.1}$$

where N is the total number of flows, $\beta_j(t_i)$ is the allocation for flow j at time t_i under our distributed scheme, and $\widetilde{\beta}_j(t_i)$ 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:

$$\overline{FI} = \frac{\sum_{i=1}^{L} FI(t_{i-1})(t_i - t_{i-1})}{t_L - t_0}. \tag{7.2}$$

7.3. 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.

7.3.1. Greedy adaptation

Figure 12 shows the simulation results under two noise levels. The fairness index under noise level 0 shows that the dis-

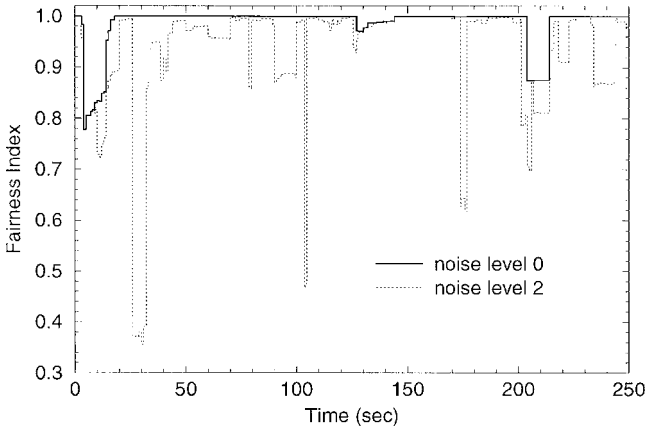

Figure 12. Greedy adaptation results: fairness index.

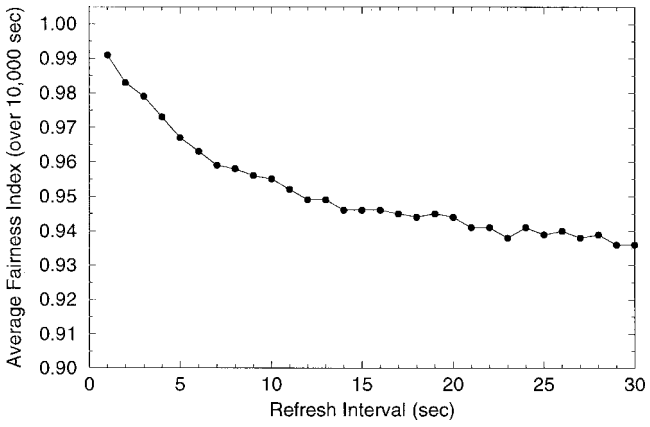

Figure 13. Greedy adaptation results: $\overline{FI}$ versus probing cycle.

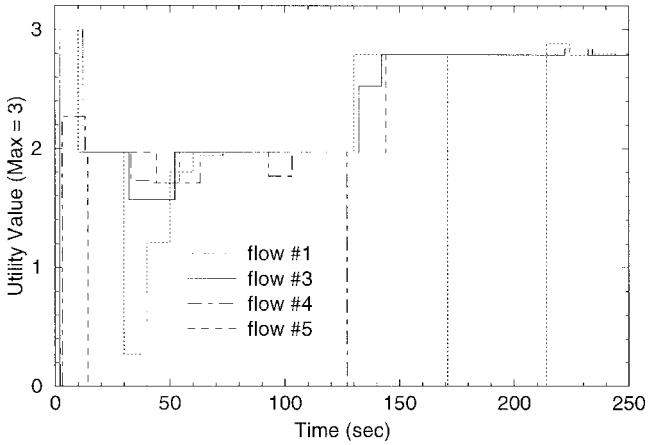

Figure 14. Greedy adaptation results: utility value.

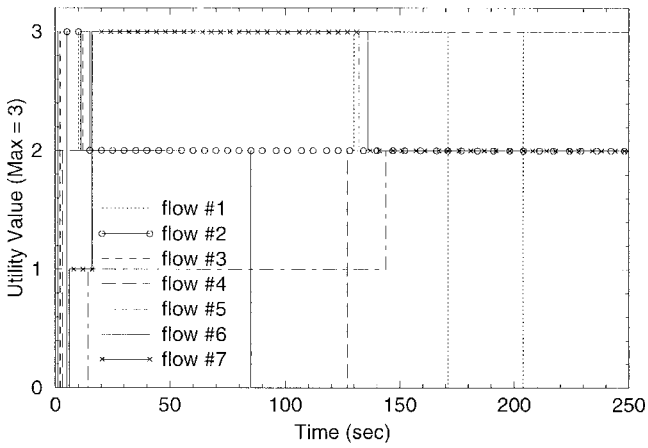

Figure 15. Discrete adaptation script results: utility value.

tributed 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 5.2.

The fairness index under noise level 2 illustrates the system performance under 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 s in duration indicates that the system rectifies the allocation inaccuracy within its probing interval of 10 s. 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 13. This figure shows that under noise level 2 conditions, the average fairness index ($\overline{FI}$ averaged over 100000 s) decreases as the probing interval increases. A good choice of the probing interval needs to balance the tradeoff between a desired fairness index value and increased signalling generated by a short probing interval.

To visualize the effect of utility-based fair allocation, we use figure 14 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 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.

7.3.2. Discrete adaptation

Figure 15 presents results from a simulation (see section 7) 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, and 3}.

We observe that strategy A can cause allocation disparity, as shown in figure 16 (which is under the same scenario as in figure 15 but for flow 3 and 5 only). Because flow 3 and 5 have the same utility function parameters and

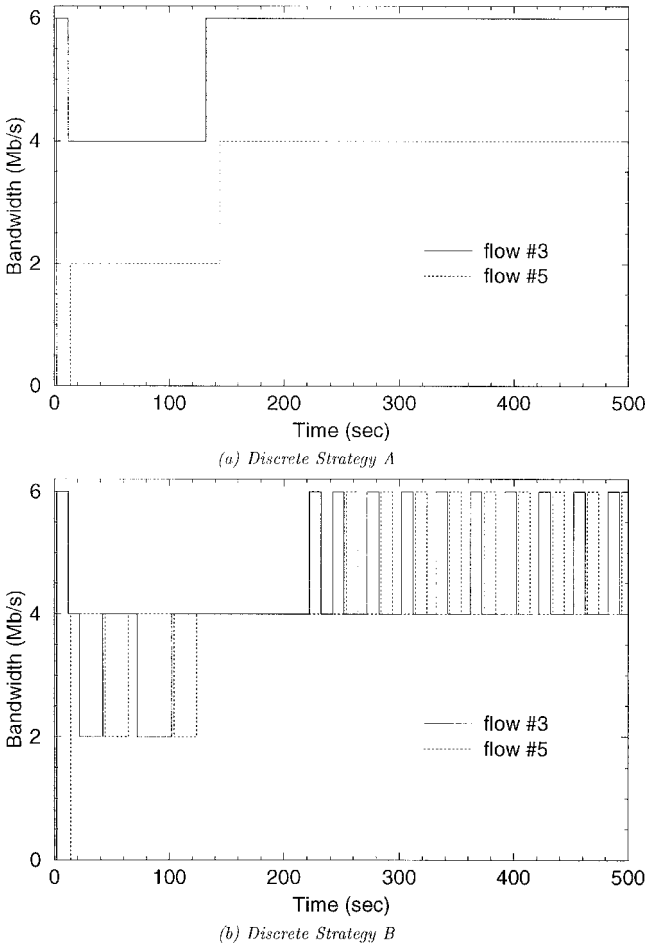

Figure 16. Comparison of discrete adaptation strategies.

the same route, they should receive the same bandwidth allocation. However, as shown in figure 16(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 16(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 fairness problem experienced by strategy A, which otherwise would be difficult to resolve in a deterministic manner.

7.3.3. Smooth adaptation

Figure 17 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 17 (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 $\tau = 20$ s and $\delta = 0.66$ Mbps, one third of its maximum bandwidth requirement. Flow 2 is not

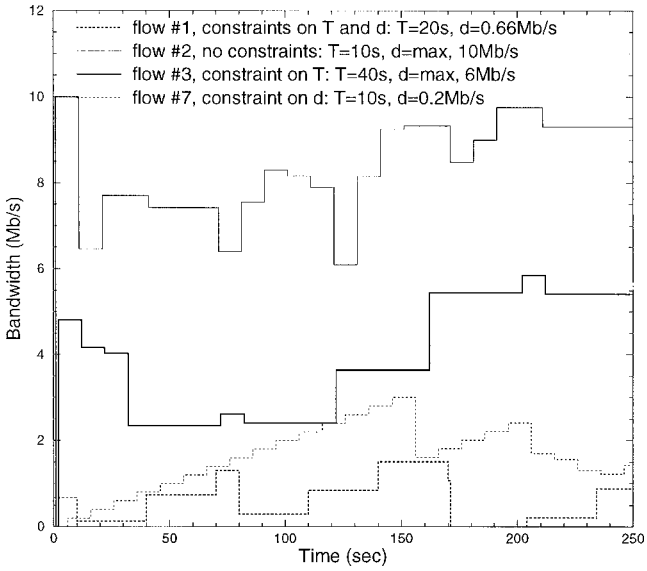

Figure 17. Smooth adaptation script results.

constrained by either τ or δ, so its $\tau = 10$ s, the default probing cycle, and $\delta = 10$ Mbps, its maximum bandwidth requirement. Flow 3 is constrained only by τ, where $\tau = 40$ s and $\delta = 6$ Mbps, its maximum bandwidth requirement. Finally, flow 7 is constrained only by δ, where $\delta = 0.2$ Mbps and $\tau = 10$ s, the default probing cycle. In comparison with the greedy adaptation flow 2, smooth adaptation flows experience less bandwidth oscillations (i.e., "QoS flapping"), as shown in figure 17. 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.

8. Conclusion

In this paper, we have 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 mo-

bile 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.

Acknowledgements

This work is supported in part by the Intel Corporation and by the Army Research Office (ARO) under ARO Advanced DAAD 19-99-10287. We would like to thank Giuseppe Bianchi and Mischa Schwartz for their valuable suggestions on improving the quality of this paper.

Appendix. 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
    β_id^ideal = 0; β_id^actual = 0;
    u_id = u;
    state_id = SATISFIED;
    update u into u_ALL;
    add u into u_U;
  } else {
    update u_ALL
    if (state_id == SATISFIED) {
      // prepare for allocation
      add u into u_U
      B_U+ = β_id^ideal;
    }
    B_free+ = β_id^actual;
  }
  alloc_bw(id, req^ideal, req^actual);
}

// utility-fair algorithm: equation (4.3)
alloc_bw(id, req^ideal, req^actual) {
  β_alloc = util_fair_alloc(id);
  // without feasibility constraint
  β_id^ideal = min{β_alloc, rec^ideal};
  // with feasibility constraint
  β_id^actual = min{β_alloc, req^actual, B_free};
  // adjust flow state
  if (β_id^ideal < req^ideal) state_id = BTLNECKED;
  else state_id = SATISFIED;
  if (state_id == SATISFIED) {
    remove u from u_U
    B_U− = β_id^ideal;
  }
  B_free− = β_id^actual;
  req_id^ideal = β_id^ideal;
  req_id^actual = β_id^actual;
}

// a commit message arrives at a network
// node
commit(id, ack^ideal, ack^actual) {
  B_free+ = (β_id^actual − ack^actual);
  β_id^actual = ack^actual;
  if (ack^ideal ⩾ β_id^ideal) return;
  // ack^ideal < β_id^ideal, adjust flow states
  if (state_id == SATISFIED) B_U+ = β_id^ideal;
  β_id^ideal = ack^ideal;
  // update index for u_max
  if (β_id^ideal > β_max) {
    β_max = β_id^ideal;
    max_id = id;
  } else if (β_id^ideal < β_max and id == max_id) {
    β_max = β_id^ideal;// reduce β_max
  }
  if (id == max_id) {
    if (state_id == SATISFIED) {
      state_id = BTLNECKED;
      add u_id into u_U;
    }
  } else {
    if (state_id == BTLNECKED) {
      state_id = SATISFIED;
      remove u_id from u_U;
    }
    B_U− = β_id^ideal;
  }
}

// flow release or timeout
release(id) {
  remove u_id from u_ALL;
  if (state_id == BTLNECKED)
    remove u_id from u_U;
  else
    B_U+ = β_id^ideal;
  // return BW to free pool
  B_free+ = β_id^actual;
  delete flow states;
}
```

References

[1] R.H. Katz, Adaptation and mobility in wireless information systems, IEEE Personal Communications 1(1) (First Quarter 1994).

[2] K. Lee, Adaptive network support for mobile multimedia, in: *Proc. ACM MOBICOM,* Berkeley, CA (November 1995).

[3] S. Lu and V. Bharghavan, Adaptive resource management algorithms for indoor mobile computing environments, in: *Proc. ACM SIGCOMM* (September 1996).

[4] M. Naghshineh and M. Willebeek-LeMair, End-to-end QoS provisioning in multimedia wireless/mobile networks using an adaptive framework, IEEE Communications Magazine 35(11) (November 1997).

[5] B.D. Noble, M. Stayanarayanan et al., Agile application-aware adaptation for mobility, in: *Proc. ACM Symposium on Operating System Principles,* St. Malo, France (October 1997).

[6] O. Angin, A.T. Campbell, M.E. Kounavis and R.R.-F. Liao, The mobiware toolkit: Programmable support for adaptive mobile networking, IEEE Personal Communications (August 1998).

[7] G. Bianchi, A.T. Campbell and R.R.-F. Liao, On utility-fair adaptive services in wireless networks, in: *Proc. IEEE/IF1P Int. Workshop on Quality of Service,* Napa Valley, CA (May 1998).
[8] 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).
[9] D. Reininger, Dynamic Quality-of-Service framework for video in broadband networks, Ph.D. thesis, Rutgers University, New Jersey (January 1998).
[10] 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).
[11] R.R.-F. Liao, P. Bocheck, A.T. Campbell and S.-F. Chang, Utility-based network adaptation for MPEG-4 systems, in: *Proc. of NOSS-DAV'99,* Basking Ridge, NJ (June 1999).
[12] Recommendation ITU-R BT.500-7, Methodology for the subjective assessment of the quality of television picture (October 1999).
[13] J. Jaffe, Bottleneck flow control, IEEE Transactions on Communications 29(7) (July 1981).
[14] D. Bertsekas and R. Gallager, *Data Networks* (Prentice-Hall, Englewood Cliffs, NJ, 1992).
[15] S. Lu, V. Bharghavan and R. Srikant, Fair scheduling in wireless packet networks, in: *Proc. ACM SIGCOMM* (September 1997).
[16] T.S.E. Ng, I. Stoica and H. Zhang, Packet fair queueing algorithms for wireless networks with location-dependent errors, in: *Proc. IEEE INFOCOM* (March 1998).
[17] S. Shenker, Some fundamental design decisions for the future Internet, IEEE Journal on Selected Areas in Communications 13 (1995) 1176–1188.
[18] L. Breslau and S. Shenker, Best-effort versus reservations: a simple comparative analysis, in: *Proc. ACM SIGCOMM* (September 1998).
[19] 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).
[20] 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. Conf. Network Protocols,* Austin, TX (October 1998).
[21] A. Fox, S.D. Gribble, Y. Chawathe and E.A. Brewer, Adapting to network and client variation using active proxies: lessons and perspectives, IEEE Personal Communications (August 1998).
[22] R. Rejaie, M. Handley and D. Estrin, Quality adaptation for congestion controlled video playback over the Internet, in: *Proc. ACM SIGCOMM* (September 1999).
[23] B. Li and K. Nahrstedt, A control-based middleware framework for quality of service adaptations, IEEE Journal on Selected Areas in Communications 17(9) (September 1999).
[24] A.G. Valko, Cellular IP – A new approach to internet host mobility, ACM Comput. Commun. Review (January 1999).
[25] R. Braden, ed., (RSVP) – Version 1 functional specification, IETF RFC 2205 (September 1997), `ftp://ftp.isi.edu/in-notes/rfc2205.txt`.
[26] A. Charny, D.D. Clark and R. Jain, Congestion control with explicit rate indication, in: *Proc. IEEE Int. Conf. Commun.* (June 1995).
[27] 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).
[28] A. Charny and K.K. Ramakrishnan, Time scale analysis of explicit rate allocation in ATM networks, in: *Proc. IEEE INFOCOM* (April 1996).
[29] The Mobiware Project: Building open programmable mobile networks (1997), source code freely available at `http://comet.columbia.edu/mobiware`.
[30] R. Jain, Congestion control and traffic management in ATM networks: Recent advances and a survey, Journal of Computer Networks and ISDN Systems 28(13) (November 1996).

Raymond R.-F. Liao is a Ph.D. candidate and Graduate Research Assistant of the COMET Group in the Department of Electrical Engineering, Columbia University. From 1993 to 1996, he worked full-time at Newbridge Networks Corporation on ATM network performance analysis and traffic management, and co-authored three patent applications. He holds an M.A.Sc. in electrical and computer engineering from University of Toronto, Canada, and a Bachelor degree from Huazhong University of Science and Technology, China. His current research focuses on dynamic provisioning for service differentiation over packet networks, and realizing utility function based adaptive service in wireless packet networks with methodologies including programmable networks and network economics.
E-mail: liao@comet.columbia.edu

Andrew T. Campbell is an Associate Professor in the Department of Electrical Engineering and member of the COMET Group, Columbia University. His areas of interest encompass mobile computing and networking, open programmable networks, and Quality-of-Service research. Dr. Campbell is active in the IETF and the international working group on Open Signaling (OPENSIG). He is member of a number of editorial boards including Computer Networks, IEEE/ACM Transactions on Networking, ACM Wireless Networks, and ACM SIGCOMM Computer Communication Review. Dr. Campbell received his Ph.D. in computer science in 1996 and the NSF CAREER Award for his research in programmable mobile networking in 1999.
E-mail: campbell@comet.columbia.edu