Abstract

Existing mobile systems (e.g., mobile IP, mobile ATM, and third-generation cellular systems) lack the intrinsic architectural flexibility to deal with the complexity of supporting adaptive mobile applications in wireless and mobile environments. We believe that there is a need to develop alternative network architectures from the existing ones to deal with the tremendous demands placed on underlying mobile signaling, adaptation management, and wireless transport systems in support of new mobile services (e.g., interactive multimedia and Web access). In this article we present the design, implementation, and evaluation of mobiware, a mobile middleware toolkit that enables adaptive mobile services to dynamically exploit the intrinsic scalable properties of mobile multimedia applications in response to time-varying mobile network conditions. The mobiware toolkit is software-intensive (comet.columbia.edu/mobiware) and is built on CORBA and Java distributed object technology. Based on an open programmable paradigm developed by the COMET Group, mobiware runs on mobile devices, wireless access points, and mobile-capable switch/routers providing a set of open programmable interfaces and algorithms for adaptive mobile networking.

The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking

Oguz Angin, Andrew T. Campbell, Michael E. Kounavis, and Raymond R.-F. Liao
Columbia University

The phenomenal growth in cellular telephony over the past several years has demonstrated the value people place on mobile voice communications. The goal of next-generation wireless systems is to enable mobile users to access, manipulate, and distribute voice, video, and data anywhere, anytime. As the demand for mobile multimedia services grows, high-speed wireless extensions to existing broadband and Internet technologies will be required to support the seamless delivery of voice, video, and data to mobile devices with sustained high quality. New wireless services will include Internet access to interactive multimedia, video conferencing, and real-time data, as well as traditional services such as voice, e-mail, and Web access.

The wireless and mobile environment presents a number of technical challenges to this vision. First, physical-layer impairments contribute toward time-varying error characteristics and time-varying channel capacity as observed by mobile devices. We describe the quality index maintained across the wireless channel as wireless quality of service (QoS). Second, user mobility can trigger rapid degradation in the quality of the delivered signal. This can lead to transient service outages resulting in handoff dropping in broadband cellular networks when a new access point is unable to accommodate a new mobile device at its current level of service. As a result, mobile applications can experience unwarranted delays, packet losses, or loss of service. We describe the quality index maintained during handoff between access points as mobile QoS.

There is growing consensus that adaptive techniques [1] present a viable approach to countering time-varying QoS impairments found in wireless and mobile networking environments. However, providing systemwide (i.e., end-system and network) adaptive QoS support for mobile multimedia communications is complex to realize in practice and not well understood by the community [2]. Recently, a number of adaptive mobile systems [3–8] have been proposed in the literature; however, few experimental systems exist today to assess the viability of the adaptive approach. We believe that there is a need to build adaptive mobile networking testbeds, study their behavior, and learn from these experiments in building more scalable adaptive mobile systems. We believe that there is a need to take a hands-on systems approach coupled with the analysis of well-founded adaptive QoS models to investigate the viability of the approach and utility of adaptive mobile networking to mobile users.

To address these challenges, we have built an open [9] and active [10] programmable mobile network [11, 12] that is controlled by a software middleware toolkit called *mobiware* [13]. Mobiware extends earlier work by the COMET Group on programmable broadband networks [14] to the mobile and wireless domain. By *open*, we mean that there is a need to "open up" hardware devices (e.g., mobile devices, access points, and mobile-capable switches and routers) for implementation of new mobile signaling, transport,and adaptive QoS management algorithms. At the lowest level of programmability, mobiware abstracts hardware devices and represents them as distributed computing objects based on Common Object Request Broker Architecture (CORBA) technology [15]. These objects (e.g., an access point object) can be programmed via a set of open programmable network interfaces to support adaptive QoS assurances. By *programmable*, we mean that these programmable network interfaces are high-level enough to allow new adaptive services to be built using distributed object computing technology. By *active*, we mean that adaptive QoS algorithms can be represented as active transport objects based on Java objects and injected on the fly into mobile devices, access points, and mobile-capable network switches/routers to provide value-added QoS support when and where needed.

In this article, we present an overview of mobiware followed by a detailed discussion of the design, implementation and evaluation of the mobiware programmable mobile network layer. The source code distribution for mobiware v. 1.0 can be freely downloaded from [13] for experimentation. The structure of the article is as follows. The next section describes an adaptive-QoS application programming interface (API) and service model, the mobiware architecture, and the network model. Following this, we present the design and implementation details of the mobiware programmable mobile network layer. This is followed by an evaluation of the system in an experimental setting and a discussion of our results. Finally, we present some concluding remarks.

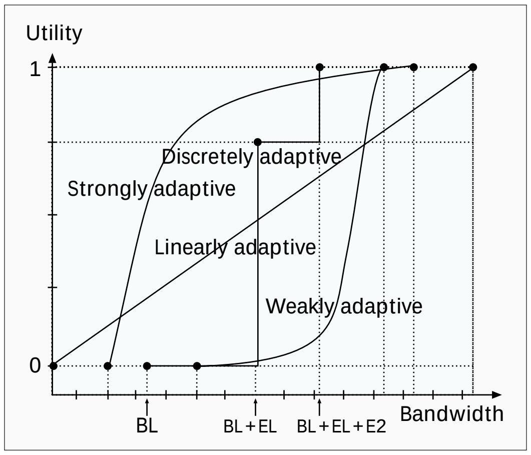


Figure 1. *Utility functions.*

Mobiware

Mobile applications need to be capable of responding to time-varying wireless QoS and mobile QoS conditions. To address this, wireless transport and adaptation management systems should be capable of transporting and manipulating content in response to changing mobile network QoS conditions. Mobile signaling should be capable of establishing suitable network support for adaptive mobile services (e.g., the delivery of scalable flows or packet services with drop preferences). Medium access controllers must be capable of sharing the wireless link capacity among mobile devices supporting adaptive QoS assurances when possible.

Mobiware is based on a methodology of open programmability [9] for the introduction, control, and management of new adaptive mobile services. It provides a set of open programmable CORBA interfaces and objects that abstract and represent network devices and resources, providing a toolkit for programmable signaling, adaptation management, and wireless transport services. Mobiware aims to provide a foundation for open programmable mobile networking that is suited to managing the evolving service demands of adaptive mobile applications and dealing with the inherent complexity of delivering scalable audio and video and real-time services to mobile devices. Built on an adaptive QoS API, mobiware consists of a set of controllers that interact with transport, network, and medium access control distributed objects that maintain application-specific adaptive QoS needs.

The Adaptive QoS API and Service Model

By trading off temporal and spatial quality with available bandwidth, mobile applications can be made to adapt to time-varying conditions with minimal perceptual distortion. In [16], we introduced an adaptive QoS API and service model specifically designed to quantitatively address the wireless QoS and mobile QoS needs of adaptive mobile applications. Mobile applications use this API at the transport layer, specifying:

- A *utility function*, which captures the adaptive nature over which an application can successfully adapt to available bandwidth in terms of a utility curve that represents the range of observed quality to bandwidth. The observed quality index refers to the level of satisfaction perceived by an application at any moment, as illustrated in Fig. 1.
- An *adaptation policy*, which captures the adaptive nature of mobile applications in terms of a set of adaptation strategies (fast, smooth, after handoff, never). These policies allow the application to control how it moves along its utility curve as resource availability fluctuates, for example, sale up only after handoff, fast adaptation, and smooth adaptation.

The utility function allows *utility-fair* bandwidth allocation algorithms to derive explicit optimization rules under heterogeneous application adaptation behavior. Here bandwidth is allocated fairly to all the flows so that the same utility value is achieved at an access point. For full details of the utility-fair bandwidth allocation algorithm, see [16]. The utility function alone, however, is not capable of capturing application-specific adaptation dynamics. Rather, a simple set of adaptation policies is used to capture how an application wishes to respond to instantaneous bandwidth availability.

A mobile multimedia application's range of perceptible quality is strongly related to how and when it responds to resource changes. Frequent oscillation between what may be considered optimal and minimum utility or even the frequent small change around an average application quality may be annoying to many applications. Some applications may wish to limit the frequency of adaptation to change (e.g., multiresolution applications). In contrast, others may wish to exploit any opportunity for adaptation (e.g., real-time data applications). By limiting or dampening the response to change an application attempts to follow trends in resource availability rather than fluctuations to instantaneous changes. Such a conservative adaptation policy may lead to a more stable operating point on an application's utility curve. This is in contrast to a policy that responds to instantaneous fast-moving points, which may suit other styles of mobile application.

The adaptive QoS API is supported by mobiware at the transport level and realized at the mobile device and in the network. Mobile applications use this API to specify flow utility functions and adaptation policy. The adaptive QoS API allows applications to associate temporal or event-based dimensions with their utility functions.

The mobiware service model supports the following adaptation "menu" policy options:[1]

- *Fast*, to instantly move up the utility curve, responding instantly to any resources changes.
- *Smooth*, to move up the utility curve only after a suitable damping period has passed.
- *Handoff*, to move up or down the utility curve only after handoff.
- *Never*, to never move up the utility curve after the initial bandwidth allocation has been made.

The Architecture

Mobiware is a software-intensive adaptive mobile networking environment based on distributed object technology. As illustrated in Fig. 2, mobiware promotes the separation of mobile signaling and adaptation management, and the transport of media. At the transport layer, an *active wireless transport* supports the end-to-end transmission of audio, video and real-time data services based on an adaptive QoS paradigm. The active wireless transport is an object-based transport that blurs the region over which traditional transports (e.g., TCP and RTP) typically operate to include access

[1] A generalization of this approach is detailed in [16]. Adaptive mobile applications supply adaptation handlers, which implement application-specific adaptation policies supporting more sophisticated levels of adaptation than the current menu options (e.g., fast, handoff) offered in the existing system. Mobiware exposes a set of low-level APIs to allow the application to control its adaptation strategy.

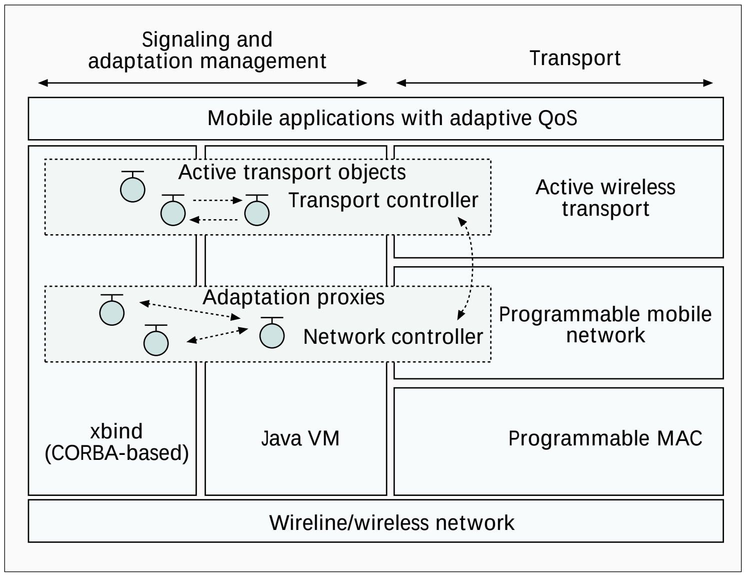

Signaling and adaptation management
Transport
Mobile applications with adaptive QoS
Active transport objects
Transport controller
Active wireless transport
Adaptation proxies
Network controller
Programmable mobile network
xbind (CORBA-based)
Java VM
Programmable MAC
Wireline/wireless network

Figure 2. *The mobiware architecture.*

points and mobile devices. Built on a set of Java classes, the transport system binds active and static transport objects at mobile devices and access points to provide end-to-end transport adaptation services. Static transport objects include segmentation and reassembly, rate control, flow control, playout control, resource control, and buffer management objects. These objects are loaded into the mobile device as part of the transport service creation process. Active transport objects can be dynamically dispatched to mobile devices and access points to support valued-added QoS. Currently, two styles of active transport objects have been implemented: active media filters [17], which perform temporal and spatial scaling for multiresolution video and audio flows, and adaptive forward error correction (FEC) filters [13], which protect content against physical radio link impairments by matching the level of Reed Solomon channel coding to time-varying error characteristics.

At the network layer, a *programmable mobile network* supports the introduction of new mobile adaptive QoS services based on the xbind broadband kernel [14]. The network layer supports switched IP flows over ATM native transport services. Architecturally, the network comprises a set of CORBA network objects and adaptation proxies that operate at the mobile device, access points and at mobile-capable switch/routers. Currently, an adaptive QoS network service supports the delivery of multiresolution flows having a base layer and one or more enhancement layers. The base layer provides a foundation for better resolutions to be delivered through the reception of enhancement layers based on the availability of resources in the wireless environment. Three key mobiware network algorithms include:

- *QoS-controlled handoff*, which gracefully scales flows (up and down) based on the semantics of the adaptive QoS service during handoff when bandwidth availability may vary
- *Mobile soft-state*, which provides mobile devices with the capability to respond to changes in wireless and mobile QoS.
- *Flow bundling*, which exploits a common routing representation for all the flows to and from a mobile device to speed up handoff.

The focus of this article is the design and evaluation of the programmable mobile network layer described in the next two sections.

At the data link layer, a *programmable MAC* [18] combines a set of foundation services to support more sophisticated adaptive wireless QoS services. Foundation services provide sustained rate services used to support minimum wireless QoS assurances, and active and passive adaptive services to support application-specific adaptation policy, as discussed earlier. The "programmable" nature of the data link service provisioning over wireless networks provides an alternative approach to that found in the literature. Rather than supporting a specified set of "hardwired" MAC services (e.g., constant bit rate) by means of centralized control schemes, it provides a programmable air interface that allows new services to be dynamically created and installed on the fly. This programmable MAC service support relies on a simple core architecture that pushes complexity and application-specific adaptation decision making to the mobile device. For full details on the programmable MAC see [18].

The Network Model

Mobiware provides a middleware toolkit that controls *mobinet,* an experimental programmable broadband cellular access

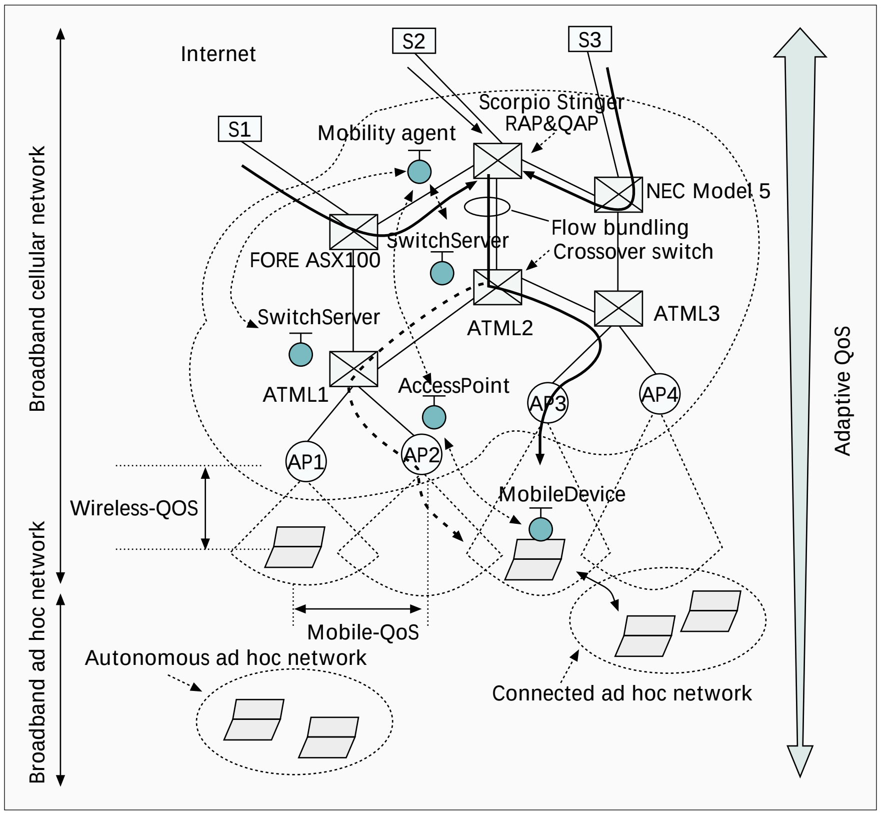


Figure 3. *Mobinet.*

network. The network model[2] comprises a set of mobile devices, wireless access points, and mobile-capable switches/routers providing broadband cellular and ad hoc communication services to mobile users. Mobinet is based on asynchronous transfer mode (ATM) switching technology that supports IP-switched flows in the access network. Mobile devices can be connected to mobinet via broadband cellular or ad hoc wireless access modes. In broadband cellular mode, mobile devices receive core network services via a set of wireless access points. Ad hoc devices may operate autonomously without the aid of any fixed infrastructure and core network services or can connect to the broadband cellular network via multiple ad hoc hops as illustrated in Fig. 3.

Providing QoS assurances in broadband cellular networks is difficult. However, providing QoS assurances without the aid of any fixed infrastructure, as in the case of mobile ad hoc networking, is more challenging [19]. We believe there is a need to understand the level of QoS that can be supported at different points of attachment in mobinet (e.g., at the access point or multiple hops away from the access point). We observe that QoS assurances are likely to diminish as a mobile device moves away from the core network. Providing seamless QoS support to mobile devices on the move (e.g., switching between broadband cellular and ad hoc modes) underpins mobiware's adaptive QoS design approach.[3]

A Programmable Mobile Network

Programmable Objects

The mobile network comprises a set of programmable distributed CORBA objects[4] that support the delivery of adaptive QoS flows to mobile devices over mobinet. The use of distributed object technology also provides support for interoperability between mobile devices utilizing different operating systems and protocol support. Mobiware objects execute on mobile devices, access points, and mobile-capable switch/routers supporting a set of mobile signaling and QoS adaptation algorithms (QoS-controlled handoff, flow bundling, and mobile soft-state), as illustrated in Fig. 3. Objects combine data structure (defining the object's state) with a set of methods (defining the object's behavior). Methods are executable programs associated with objects that operate on information in an object's data structure.

Two per-mobile proxy objects support the adaptation and handoff of flows in mobiware:

- *QoS adaptation proxy (QAP)* objects play an integral role in allowing mobile devices to probe and adapt to changing resource availability over the wireless link.
- *Routing anchor proxy (RAP)* objects support the "bundling"(i.e., aggregation) of flows to and from mobile devices for fast, efficient, and scalable handoff.

```
interface AccessPoint : NodeServer {
  // flow setup from the current access point to
  // the network, called by the mobile device object
  void setupFlow (in long fbi,
            inout QOSSpecification qosSpec,
            inout FlowInfo flowinfo, in string<40> scrname,
            inout EndPoint peerEp,
            inout EndPoint RAP_fix, inout EndPoint RAP_mobile,
            inout EndPoint AP_fix, inout EndPoint AP_mobile,
            inout End PointId airIP, inout double msr_time)
        raises (Reject);

  // handoff flow bundle for a specific mobile device
  void handoffFlowBundle (in long fbi, inout QOSSpecList
            qosSpec [2], inout FlowInfoList flowinfo [2],
            inout SourceList scrname [2], inout EndPointList
            RAP_fix [2], inout EndPointList RAP_mobile [2],
            inout EndPointList AP_fix [2], inout EndPointList
            AP_mobile [2], inout EndPointList airIP [2],
            inout double msr_time)
        raises (Reject);

  // refresh mobile soft-state for flow bundle
  // through the current access point to the network
  void refreshFlowBundle (in long fbi,
            inout EndPointList RAP_mobile [2],
            inout EndPointList AP_fix [2],
            inout EndPointList AP_mobile [2],
            inout double ntw_msr_time)
    raises (Reject);

  // initialize mobile related states during registration
  void mobileRegistration (in long fbi,
            const char *rapName, const char *cmName,
            const char *msName)
    raises (Reject);
};
```

■ **Figure 4.** *CORBA IDL for an access point object.*

To manage the network state introduced by flow-oriented communications and, importantly, to gain efficiency across a wireless link, mobiware deploys a number of network objects that can execute on network nodes or on servers at the edge of the network. In the following we outline the function of some of the mobiware objects and their interface definitions. For full details on the objects and interfaces see [13].

A *mobile device object* abstracts the operation of mobile devices and provides APIs for querying beaconing information, registering with new access points, establishing flows, renegotiating QoS and handing off flows. It also includes the functionality to dynamically control the transport system at the access points (e.g., to set the media scaling or error control level for video flows). The mobile device object state mainly consists of QoS specification (viz. utility function and adaptation policy) for all the flows transported to/from the mobile device, and routing information including the source/destination address and current RAP and access point addresses.

An *access point object* supports APIs for binding to wireline network objects (e.g., mobility agent) on behalf of mobile stations, propagating CORBA calls, and establishing and periodically refreshing local wireless flow state, as illustrated in Fig. 4. This object plays an important role in QoS-controlled handoff and interacts with the transport system for the injection of active transport objects.

A *mobility agent object* provides flow, adaptation, and mobility management services. It interacts with per-mobile RAP and QAP state in the switch servers and supports APIs

[2] *It is envisioned that the Internet will provide interconnectivity between a set of these access networks providing a programmable cellular internetworking environment. In this case the Internet will support macro-level mobility using Mobile IP and the programmable access networks will support QOS controlled handoff, flow bundling, mobile soft-state and active wireless transport services to mobile devices.*

[3] *In this article we focus on adaptive quality of service support for programmable broadband cellular communications. Currently, mobile ad hoc support is under investigation by the COMET group [19, 20].*

[4] *Mobiware programmable objects also include active transport objects based on Java classes that "plugin" to the wireless transport data path at access points to provide value-added quality of service adaptation support. Currently, the active wireless transport supports active media scaling and adaptive FEC support.*

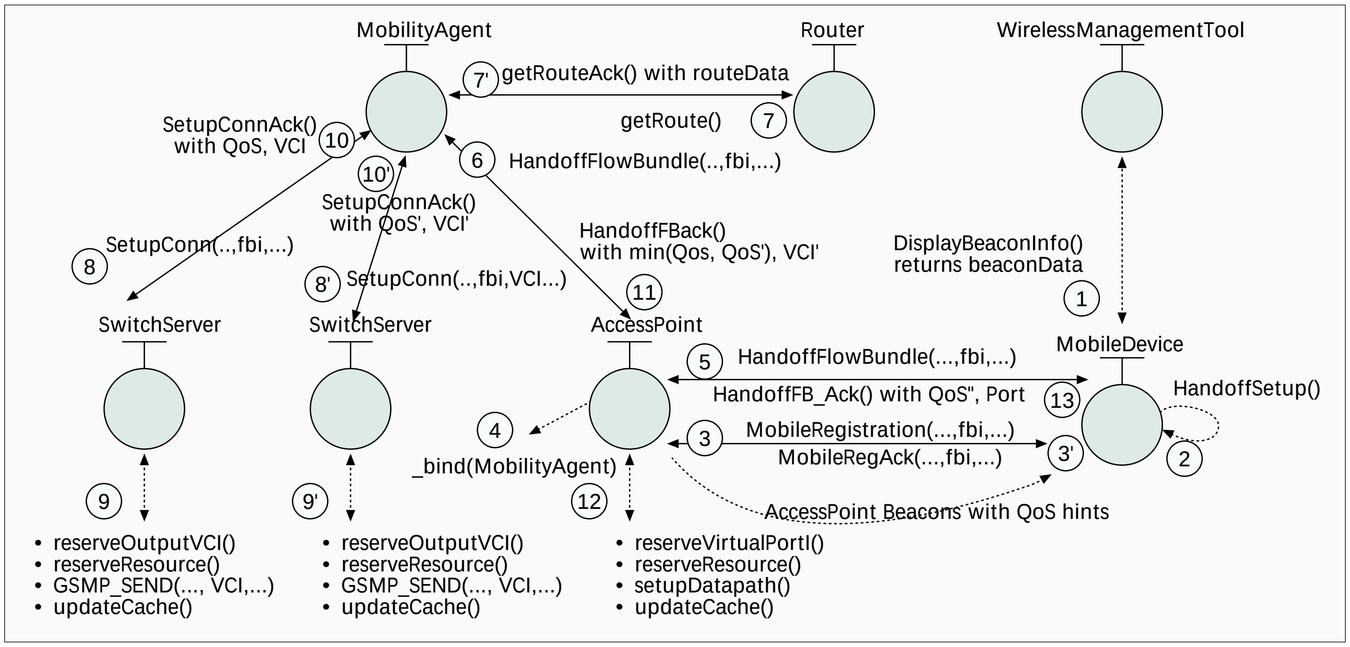

MobilityAgent
Router
WirelessManagementTool
7' getRouteAck() with routeData
getRoute() 7
SetupConnAck() with QoS, VCI 10
10'
6 HandoffFlowBundle(..,fbi,...)
SetupConnAck() with QoS', VCI'
HandoffFBack() with min(Qos, QoS'), VCI'
DisplayBeaconInfo() returns beaconData
8 SetupConn(..,fbi,...)
8' SetupConn(..,fbi,VCI...)
11
1
SwitchServer
SwitchServer
AccessPoint
MobileDevice
5 HandoffFlowBundle(...,fbi,...)
HandoffFB_Ack() with QoS'', Port 13
HandoffSetup()
4
_bind(MobilityAgent)
3 MobileRegistration(...,fbi,...)
MobileRegAck(...,fbi,...) 3'
2
9
9'
12
AccessPoint Beacons with QoS hints
• reserveOutputVCI()
• reserveResource()
• GSMP_SEND(..., VCI,...)
• updateCache()
• reserveOutputVCI()
• reserveResource()
• GSMP_SEND(..., VCI,...)
• updateCache()
• reserveVirtualPortI()
• reserveResource()
• setupDatapath()
• updateCache()

Figure 5. *QoS-controlled handoff object interaction.*

for retrieving network topology information from a router object (e.g., location of the crossover switch) and for interacting with the switch servers (see below) to establish, maintain, and hand off flows in the cellular access network.

A set of *switch server objects* [14] abstract and represent physical ATM switch/routers and are QoS programmable. These objects support APIs for the reservation and release of namespace, such as virtual channel identifier/virtual path identifier (VCI/VPI) pairs, and the allocation of network resources (e.g., bandwidth). State mainly consists of per-flow connection information, stored in local hash tables called *switch caches*. Switch server objects have been extended to be mobile-capable (i.e., support RAP and QAP functionality). The General Switch Management Protocol (GSMP [21]) is used at the access points and switch/routers for accessing the switch tables.

QoS-Controlled Handoff

QoS controlled handoff gracefully scales flows during handoff based on the semantics of the adaptive QoS API described earlier. By scaling flows during periods of resource contention (e.g., during handoff), mobiware improves the wireless resource utilization and helps reduce the handoff dropping probability. While the style of handoff is entirely programmable [22], the current implementation style is mobile-initiated, forward handoff with soft handoff on the downlink and hard handoff on the uplink. By mobile-initiated, we mean that after a suitable dwell time a mobile device initiates a handoff by first registering with the forward/new access point. By soft handoff, we mean that during handoff the mobile device simultaneously receives flows from the old and new access points on the downlink. In contrast, uplink flows use a “break and make” approach between the old and new access points. During handoff, registration to the new access point, rerouting of flows, and QoS adaptation are accomplished by signaling objects and associated APIs outlined in the previous section. Signaling APIs are programmable,[5] allowing various styles of handoff to be tailored toward particular radio environments.

The QoS-controlled handoff object interactions are illustrated in Fig. 5. Mobile device objects periodically “hunt” for beacon signals from neighboring access points. Beacons are made programmable by the mobiware toolkit and carry low-level signal information in addition to the current bandwidth availability at the sending access point. The mobile devices’ hunting algorithm periodically compares all beacons received over the current hunt period and cumulatively over multiple hunt periods. If the wireless QoS[6] indicated in the beacon from the current access point falls below a predetermined threshold, the hunt algorithm selects a new access point for handoff. Handoff is initiated after a suitable dwell period, when the mobile device registers with the new access point starting the handoff process as indicated by 2 and 3 in Fig. 5.

The device registration procedure triggers the new access point object to bind to a mobility agent object (4). The mobility agent caches bindings to the per-mobile adaptation proxies that are implemented as part of switch servers. Mobility agents and proxies can run anywhere in the mobinet, that is, mobility agents can operate at fixed edge devices or mobile devices, or on the switches. Mobility agents are the main controllers for managing handoff in mobiware. Mobility management is a fully distributed algorithm that includes one or more mobility agents for scalability. Currently, we allocate a single mobility agent to manage handoff in the experimental mobinet. When the mobile device initiates a handoff (5), it passes a unique

[5] *The programmable object-oriented nature of the mobiware signaling system makes it easy to “program” different styles of handoff algorithm (e.g., network-initiated handoff that is hard by nature) that can operate in parallel to other styles of handoff over mobinet. For full details of the programming interfaces and results from the different styles of programmable handoff, see [22].*

[6] *In addition to monitoring delay, loss, and bandwidth characteristics of flows, a mobile device object receives beacons that inform it of the state of the wireless link. The beacon informs the mobile device of the channel conditions upon reception, reporting the signal level, silence level, signal quality, and antenna selected. The signal and silence levels are derived from the receiver’s automatic gain control settings. Beacon messages are augmented with a 16-bit field that indicates the available bandwidth at the access point. The mobile device can use this to scale down its request for bandwidth resources during handoff, given that the bottleneck node is typically the access point. Our radios are based on WaveLAN operating in the 2.4–2.8 GHz industrial, scientific, and medical (ISM) band, to which we have low-level access for programming the beacon.*

mobile device identifier called the flow bundle identifier (FBI) to the access point that allows mobiware to identify the mobile device's flow bundle in the wireless access network.

Mobility agents are responsible for rerouting a mobile device's flow bundle from an old access point to a new one, as illustrated in Fig. 5. This entails the mobility agent invoking the route object (7) to determine the location of the crossover switch. Switch server objects are used to reestablish new flow state at all switches between the crossover switch and the new access point. The rerouting phase includes name space reservation (outgoing VCI/VPI) and bandwidth value at each network switch and the new access point. The final process of rerouting a flow bundle through a switch includes the use of GSMP (9) (9') (12) to set up the switch table and reserve resources. However, GSMP does not support the concept of flow bundling. While the mobile agent informs the switch server objects to establish state for a complete flow bundle, the switch server interacts with GSMP on a flow-by-flow basis. In the "Evaluation" section we describe enhancements to GSMP to support flow bundling. The mobility agents interact with mobile-capable switch servers and the new access point in parallel (8) (8') (11), resulting in a speedup of the rerouting phase of the handoff algorithm over conventional hop-by-hop signaling. After the rerouting of the flow bundle is complete, the mobile agent informs the new access point of the negotiated QoS and flow bundle VCI/VPI mappings (11). The new access point interacts with the active wireless transport to provide active media filters based on the available bandwidth at the air interface [17].

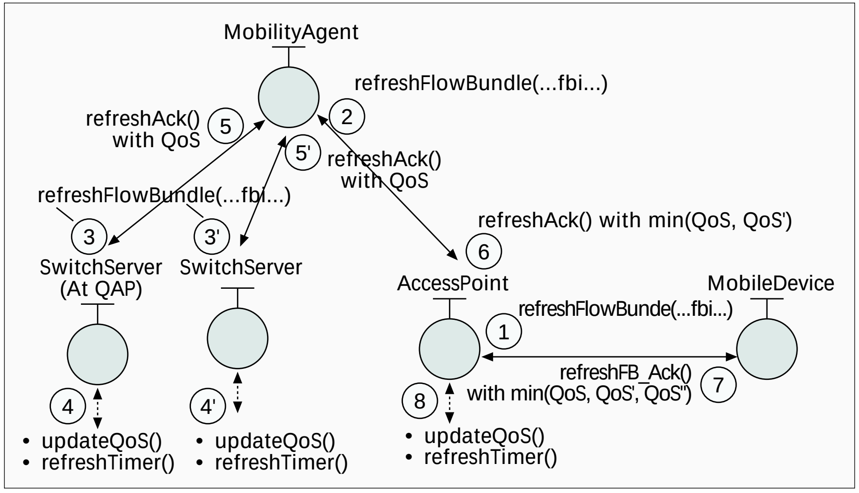


Figure 6. *Mobile soft-state object interaction.*

To keep the name space binding between the mobile device and access points constant with mobility, we have implemented the notion of *virtual wireless ports*. As mobile devices connect to different access points, their VPI/VCIs mapping remain constant. The flow bundle to VPI/VCI bundle is resolved by a virtual wireless port, which is dynamically allocated by the new access point during handoff. This approach minimizes the impact of renegotiation in comparison to a full name space renegotiation, which would be disruptive during handoff.

Flow Bundling

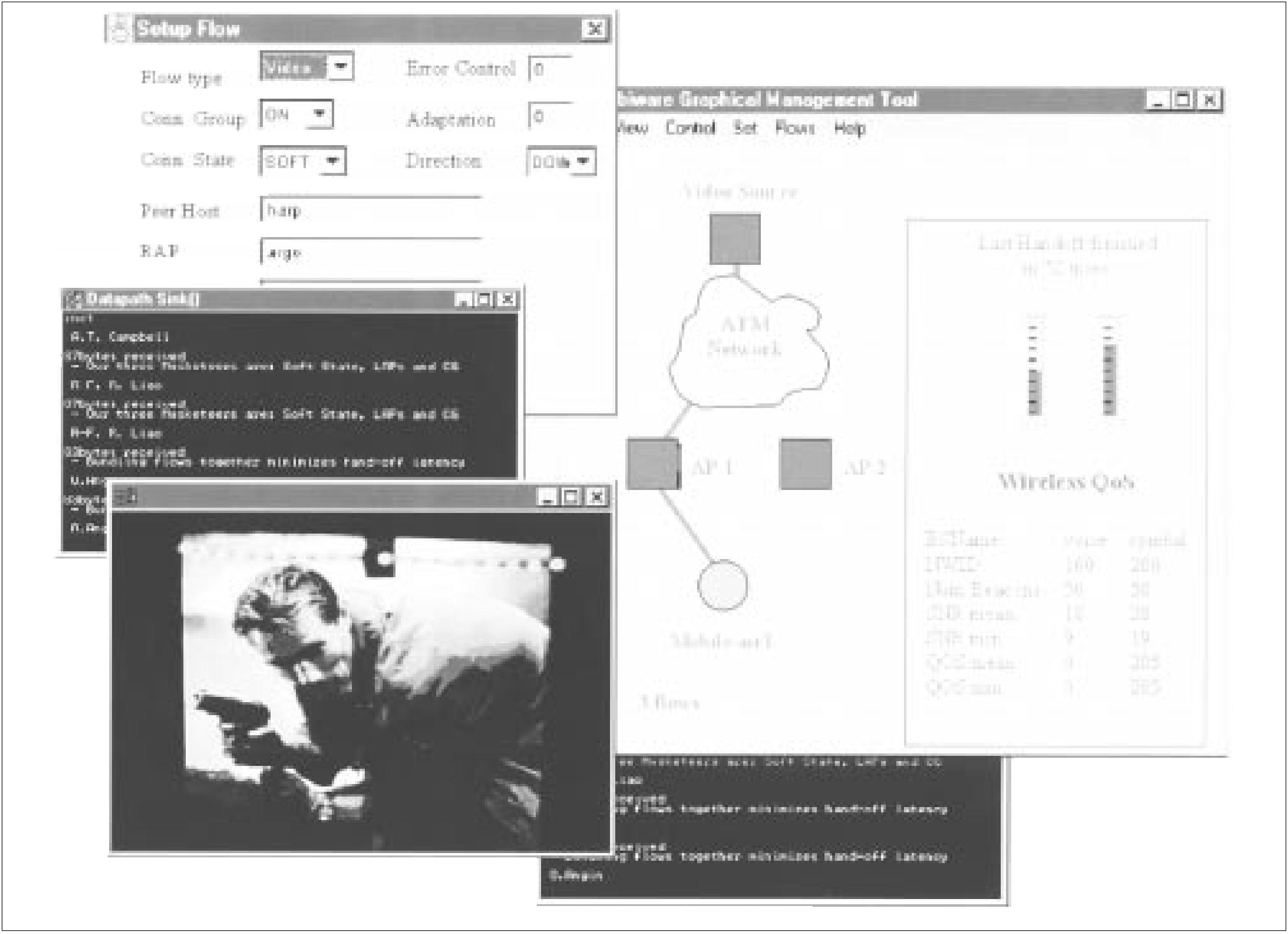
QoS-controlled handoff and mobile soft-state exploit flow bundling to speed up handoff and minimize the signaling overhead associated with maintaining the network state. Flow bundling [23] provides a common routing representation for all the flows to and from a mobile device, as illustrated in Fig. 3. This is similar to the virtual path concept in ATM networks or tunneling in IP networks. Flow bundling is a general method for encapsulating and routing. Flow bundling is motivated by the need to reduce the complexity of rerouting multiple independent flows to and from mobile devices during handoff. By aggregating flows in this manner we can speed up handoff, simplify mobile soft-state probing and minimize signaling overhead. Using flow bundling, QoS-controlled handoff simply discovers a single crossover switch, then reroutes all flows to the new access point in a single object-level operation.

During handoff the flow bundle object interaction is as follows. The mobile device object invokes the access point's HandoffFlowBundle() method once for the flow bundle, minimizing the signaling overhead at the air interface, as illustrated in Fig. 5. Mobiware supports the option of enabling or disabling bundling when flows are established. Note that when flow bundling is disabled the mobile device, access point, and mobility agent objects treat each flow independently during handoff. As discussed in the evaluation, this increases the signaling overhead and the handoff latency. During each invocation a separate crossover switch needs to be located, using a shortest path algorithm and individual flows need to be rerouted and signaled independently.

The mobile agent interacts with the switch servers to reestablish flows and update switch tables for all switches between the crossover switch and the mobile device. GSMP is used to update the switch table at each traversed switch. In order to support the atomic rerouting of flow bundles we have enhanced the GSMP protocol. Flow aggregation at the switch control level has been implemented as a modification to the GSMP invocation mechanism. Two enhancements to GSMP to support flow bundling are considered later. The first enhancement has no impact on the current GSMP client–server interaction. The mobile agent simply invokes the GSMP client for each flow in a flow bundle without waiting for acknowledgment from each GSMP command. This results in the switch server sending a burst of GSMP setup messages to the switch, then waiting for acknowledgments. The second enhancement augments the GSMP setup message to allow up to 256 VCI pairs to be passed across the interface in one client–server interaction. This allows the switch server to set up the switch table for a flow bundle in one operation. We discuss the performance benefits of flow bundling later.

Mobile Soft-State

During handoff a flow bundle must be rerouted to a new access point, resources need to be reserved, and the old flow bundle state between the old access point and the crossover switch must be removed. Mobile devices resident in cells also need to scale flows in accordance with channel conditions, whether new flows are established or released, and when new mobile devices enter and leave cells. Mobile soft-state provides QoS adaptation support to cater to a number of these conditions. Mobile soft-state results in the periodic establishment of bandwidth and name space resources for flow bundles between a mobile device and a per-mobile QAP. Mobiware supports the idea of soft-state [24] in the mobinet to refresh the network state. The periodic refresh messages sent by a mobile device as part of a soft-state probing mechanism are sent on a per-flow-bundle basis, not a per-flow basis. This means a mobile device sends a single probe message for all flows supported at the mobile device rather than one


Setup Flow
Mobiware Graphical Management Tool
View Control Set Flows Help
Video Source
ATM Network
AP 1
AP 2
Mobile ari1
3 flows
Last Handoff finished in 52 msec
Wireless QoS

Flow type	Video
Error Control	0
Conn. Group	ON
Adaptation	0
Conn. State	SOFT
Direction	DOW
Peer Host	harp
RAP	argo

BSName	voice	spatial
[illegible]	160	200
[illegible]	50	50
[illegible]	18	28
[illegible]	9	19
QOS mean	0	205
QOS max	0	205

```
Datapath Sink()
start
 A.T. Campbell
0Bytes received
 - Our three Musketeers are: Soft State, LMPs and CG
 A.F. R. Liao
0Bytes received
 - Our three Musketeers are: Soft State, LMPs and CG
 A-F. R. Liao
0Bytes received
 - Bundling flows together minimizes hand-off latency
 U.Ha
```

```
es Musketeers are: Soft State, LMPs and CG
Liao
eived
g flows together minimizes handoff latency

eived
ling flows together minimizes hand-off latency
 O.Angin
```

■ **Figure 7.** *Mobiman: a Java-based management tool.*

probe per flow. During the refresh phase mobile devices respond to any changes in allocated bandwidth (based on utility functions) to a flow bundle.

In [25] we argue that a soft-state approach is well suited to supporting QoS adaptation in mobile networks. Mobile soft-state supports a distributed probing mechanism based on flow bundles, allowing mobile devices to compete fairly for bandwidth in a completely decentralized, competetive, and scalable manner. During handoff mobile devices do not explicitly remove the old flow bundle state between the old access point and the crossover switch. In this case, mobile soft-state *timers* located at the switches and old access point timeout and release resources automatically. Mobile devices resident at the old access point compete for these available resources, thereby potentially improving their utility.

Mobile devices periodically probe the path between the mobile device and the per-mobile QAP and contend for resources. Note that per-mobile QAPs can be located at an access point or any mobile switch/router between the mobile and its corresponding RAP. If the QAP is located at the access point, mobile soft-state is only active over the air interface; that is, between the mobile device and access point. The position and configuration of where these proxies reside is programmable. In many cases, the access point is the most suitable location because radio resources are generally the bottleneck in broadband wireless or wireless LAN systems.

Mobile devices independently probe the wireless access network. The probe includes a list of flow requirements for the complete flow bundle, which includes the utility function for each flow. In the current system, we have implemented discretely adaptive flow support (Fig. 1) where a base layer (BL) and one or more enhancement layers (E1, E2) would be signaled in the probe message. The probe interacts with the access point and all mobile-capable switches/routers between the mobile device and its QAP. The response is returned to the mobile device indicating the available bandwidth reserved for each flow in the bundle over the next time interval. This time interval is called the *refresh interval*. If a probe message is received along the path before the refresh interval expires, the reservation is "refreshed" based on available resources. In contrast, if no refresh message is received and the timer expires, the resources are deallocated and states are removed. This reservation-adaptation style of probing and response is implemented at the object level as a set of refreshFlowBundle() method invocations (2) (3) (3'), as illustrated in Fig. 6. The mobile device issues a refresh to the mobility agent object, which then refreshes all switches and the current access point on the path between the mobile and QAP.

Evaluation

To evaluate the programmable mobile network performance, we have built the mobinet testbed, developed a wireless network management tool, and designed a set of experiments to analyze QoS-controlled handoff, flow bundling, and mobile soft-state algorithms. Through the course of measurement, evaluation, and redesign we managed to substantially improve upon our initial baseline design.

The Experimental Environment

At the network level, mobinet provides wireless access to the Internet and comprises four ATM switches[7] (ATML Virata, Fore ASX/100, NEC model 5, Scorpio Stinger switches) and four wireless access points, as illustrated in Fig. 3. The access points run the mobiware code and are based on a set of multihomed 200 MHz Pentium PCs that provide radio access to a wireline IP/ATM switched access network. High-performance notebooks (IBM, Gateway, and NEC) provide support to mobile applications and mobile access to the access network. Wireline ATM links operate at OC-3 rates between the switches and fixed end systems, and at 25 Mb/s between the switches and access points. Currently, the air interface is based on WaveLAN operating at 2 Mb/s in the 2.5 GHz band. A future version of the mobinet will operate at 25 Mb/s spectrum in the 5 GHz NII/SUPERNet unlicensed band. Mobile-capable switches, access points and mobile devices are abstracted as programmable CORBA objects. Mobiware requires Internet Inter-ORB Protocol (IIOP) CORBA for mobile signaling and adaptation management. For the results provided in this article the mobile devices and access points use IONA's Orbix CORBA running under Windows NT, and the mobile-capable switches/routers use IONA's Orbix running under UNIX or proprietary operating systems.

We have designed and implemented *mobiman*, a Java-based management tool for wireless networks, as illustrated in Fig. Mobiman drives experiments, displays recorded statistics, and provides network management capability to view the network and inspect the state of any mobile device. To measure the performance of the programmable mobile network, we inserted measurement checkpoints throughout the code and recorded performance statistics during the experiments. Mobiman, which runs on any fixed or mobile device, can remotely target any mobile device operating in mobinet displaying mobile device object statistics. It can set up flows, turn flow bundling and mobile soft-state on/off, interact with media scaling and adaptive FEC control at the transport level, and force handoff operations to occur. Measured information displayed by the mobiman wireless management tool comprises flow information, QoS measurements (e.g., signal level and access point bandwidth availability), and experimental checkpoint measurements. Mobiman displays measured information, wireless network topology, and mobile device location in a control window, as illustrated in Fig. 7. When a mobile device is selected by mobiman a control window indicates the state of the mobile; for example, three flows are delivered to "mobile-air#1" in Fig. 7. In this example, mobile-air#1 is running mobiman and the three flows correspond to the playout of the "True Lies" video and two low-resolution text windows. A flow setup panel appears in the top left corner of Fig. 7.

Microsoft's Active Movie is used for the reception, decoding, and rendering of digital video. It provides a software tool capable of controlling and processing streams of multimedia data. Active Movie uses modular components called filters and filter graphs. Typically, a filter graph consists of a source filter that provides the system with multimedia data, a transform filter that performs data decompression, and a rendering filter. Active Movie's filter graph has been enhanced with an

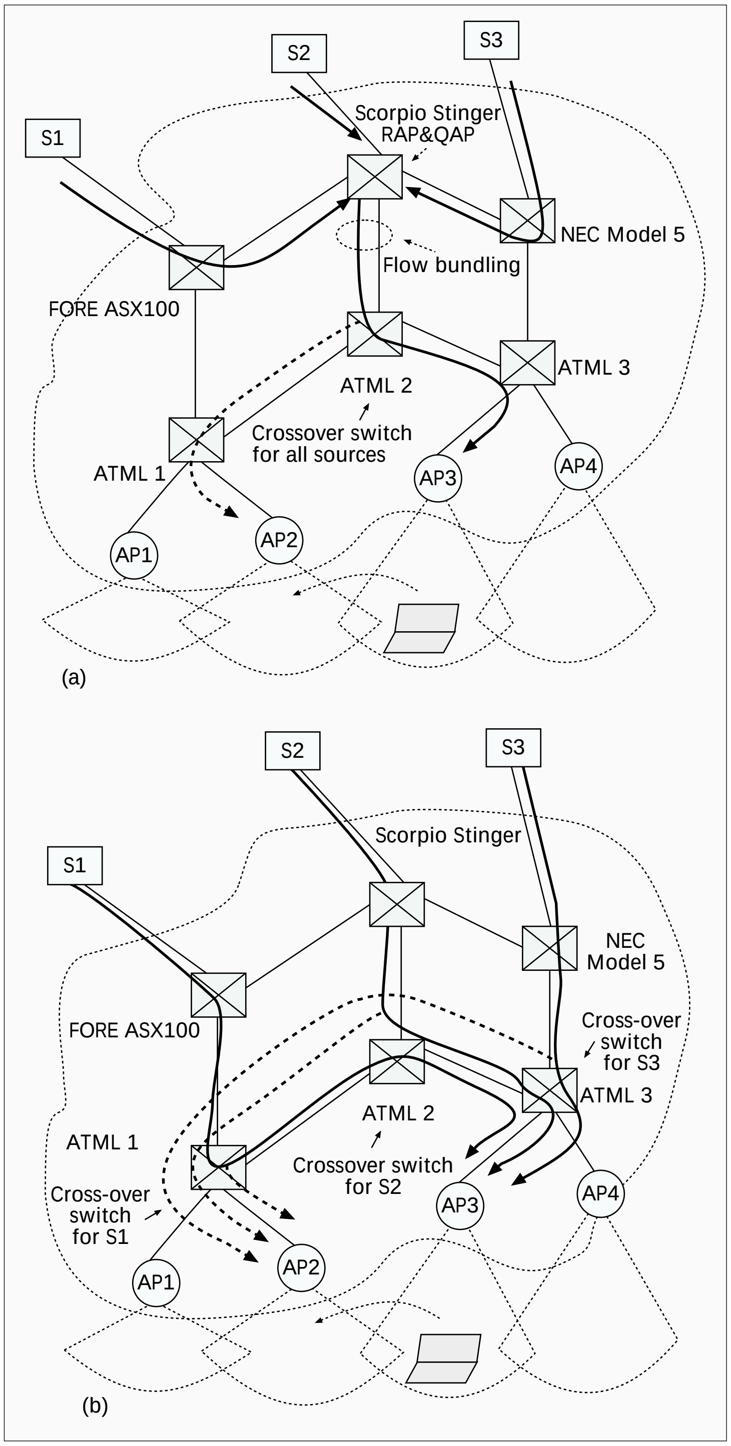


Figure 8. *Handoff (a) with flow bundling on; (b) with flow bundling off.*

[7] *We modify the concept of switchlets [26] to provide an extended network for mobiware evaluation. Switchlets allow multiple virtual network elements to be operational within the same physical nodes. Three ATM switches (ATML 1, 2, and 3, as shown in Fig. 3) in our network are switchlets physically collocated at the same physical switch. For example, packets traversing three switchets located at the physical switch travel across the physical switch three times via cables that directly connect one port to another in the same switch. Each switchlet corresponds to a different CORBA switch server object with a different name space, and manages its own resources and controls connections independently from others.*

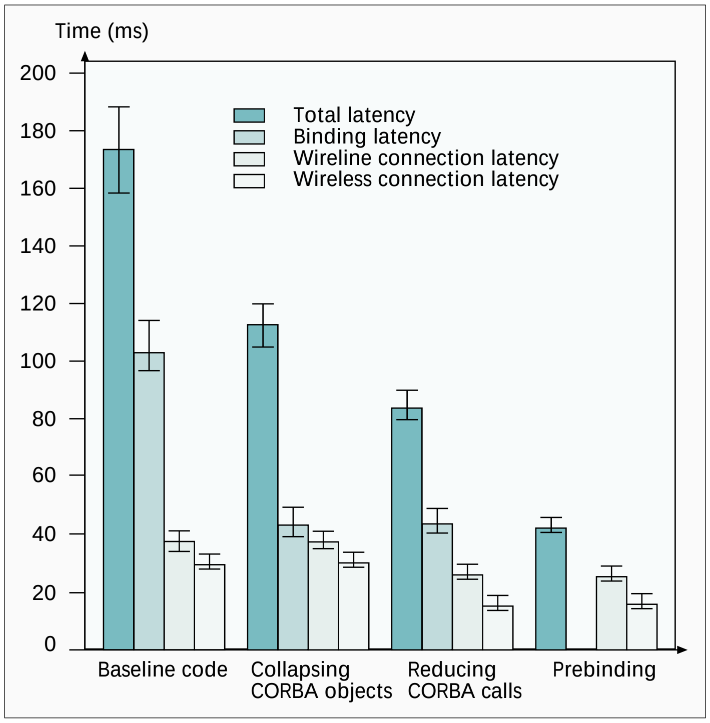


Figure 9. *Handoff latency measurement results.*

appropriate mobiware static transport object to perform synchronization of flows during handoff, delay-jitter control, and rate control.

The Experiments

Our evaluation methodology is based on a set of experiments designed to investigate the performance of the mobiware programmable mobile network in supporting mobile multimedia communications. The use of CORBA for mobile signaling, wireless adaptation and mobile network programmability is a novel aspect of our implementation. CORBA objects run at the edges and in the mobile network to support wireless and mobile QoS. An important aspect of our evaluation was to determine if such distributed object technology is viable in supporting mobile signaling and adaptation management.

Handoff Analysis — An important objective of this experiment is to measure the handoff latency and understand how the signaling system delays breakdown. In this experiment we investigated the handoff of a single flow. Handoff with flow bundling is described in the next section. For this experiment we streamed a single video flow from a fixed network server (S1) to a mobile device (M1) as illustrated in Fig. 3. The mobile device moved repeatedly between access points AP2 and AP3 with the crossover switch located at the ATML2 switch. The average handoff latency for the baseline code was measured to be 171 ms. This measurement broke down into 102 ms for mobile registration and object binding, 30 ms for wireless ATM connection setup, and 39 ms for wireline connection setup. The greatest portion of the total latency time was absorbed by the binding process between objects during handoff. As described earlier, the mobile device object remotely binds to the access point object at the forward access point. Following this, the access point locally binds to a QoS mapper object and remotely binds to a mobility agent object for handoff control.

The following enhancements were made to the baseline code to reduce binding and remote procedure call (RPC) overhead. First, by collapsing unnecessarily independent CORBA objects into a single object, the binding overhead was reduced. To reduce binding over the air interface, the mobile proxy and QoS mapper located at the access point object were collapsed into a single CORBA object. This reduced the number of CORBA requests across the air interface, reducing the binding time from 102 ms to 42 ms. Collapsing objects in this manner reduced the handoff latency to 111 ms, as illustrated in Fig. 9. Next, by reducing the number of CORBA RPCs between objects during handoff, the overhead was further reduced. An RPC across the air interface between the mobile and access point took an average of 15 ms to complete. The number of RPCs between the mobile and access point was reduced from four to two (registration, handoff request). Reducing the number of CORBA RPCs during handoff reduced the handoff latency by 28 ms to 83 ms (Fig. 9).

The final enhancement to the baseline handoff code exploited the concept of caching object bindings. In order to eliminate the binding latency, we set up and cached bindings between remote CORBA objects prior to handoff being initiated; we call this *prebinding*. All access points periodically broadcast their beacons, which include address information, signal strength, and available bandwidth resources at the access point. When mobile devices receive these beacons in promiscuous mode they register the signal quality in lieu of a possible handoff to a new access point. A prebind capability was added to the programmable mobile network to allow mobile devices to prebind to neighboring access points in advance of handoff. The prebinding criterion is based on the signal strength and available resources. The prebinding algorithm issues a prebind to an access point object on the fly, establishing TCP connections for the CORBA IIOP between the mobile device and the access points. Another enhancement establishes bindings between all access point objects in a domain and its associated mobile agent object. This final enhancement reduced the average handoff latency from 83 ms to 41 ms.

Flow Bundling Analysis — This experiment evaluates the performance gains using flow bundling during handoff. We observe the performance of handing off multiple flows with flow bundling disabled and then enabled. In the experiment video is streamed from three independent sources (S1, S2, S3) across the network to a single mobile device, which repeatedly moves between access points AP2 and AP3. When flow bundling is disabled (Fig. 8b) each flow S1, S2, and S3 is independently rerouted during handoff via the ATML 1, ATML 2, and ATML 3 crossover switches, respectively. When flow bundling is enabled all flows are bundled at a per-mobile

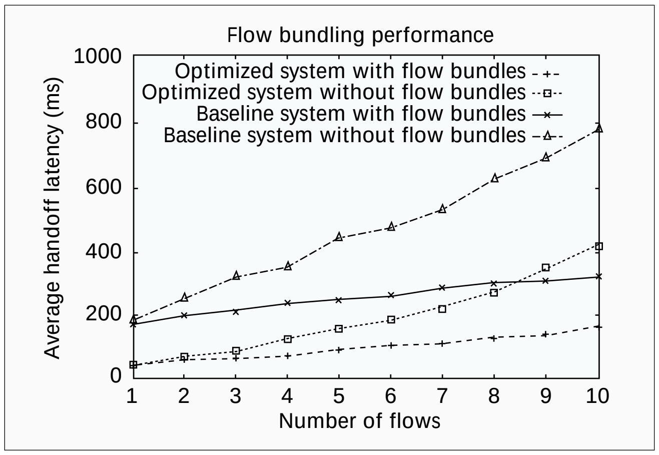


Figure 10. *Performance gain with flow bundling.*

QAP located at the Scorpio switch and rerouted during handoff via a single crossover switch located at the ATML 2 switch (Fig. 8a).

In this experiment, we vary the number of video streams transported to/from a mobile device from one to ten flows with bundling enabled and disabled, and measure the handoff latency. We observe that the results from the baseline measurement highlight the performance improvement (i.e., speedup in handoff) gained using flow bundling techniques in the access network (Fig. 10). As indicated in the figure, the benefit of flow bundling becomes more pronounced as the number of flows increases. For example, the handoff latency for two flows is 200 ms with flow bundling enabled and 250 ms when bundling is disabled. For ten flows with and without flow bundling enabled the handoff latency is 320 ms and 780 ms, respectively. The benefit of flow bundling reduces the handoff latency and, importantly, simplifies the state management of flows in the cellular access network. The adoption of flow bundling provides an improvement of 20 percent for two flows and 59 percent for ten flows. With flow bundling enabled (Fig. 8a) the handoff latency converges, whereas with flow bundling disabled the latency increases almost linearly as the number of flows increases. These results indicate the benefit of using flow bundling to reduce handoff latency and signaling overhead. This is mainly due to the fact that all interactions between objects during handoff deal with aggregated signaling rather than per-flow signaling.

The baseline code only provides flow bundling support between the mobile device, access point, and mobility agent objects. The interface between the mobility agent and the switch server is per-flow. Another observation is that the GSMP interface between the switch server object and switch does not provide any support for aggregation (i.e., a GSMP client cannot update the switch table for more than one VCI pair). To address this we enhanced the GSMP interfaces used by the switch server object. This resulted in seamless support for flow bundling aggregation from the mobile device to the switch tables, providing some level of speedup. The GSMP enhancements include a "parallel" enhancement, which did not require any changes to the GSMP code. In this case, for two flows the latency for total GSMP messages is 849 μs without aggregation and 511 μs with aggregation, showing a 40 percent improvement. With increasing number of flows, the total gain obtained by aggregation increases to 70 percent for ten flows (3907 μs vs. 1184 μs).

The baseline code was enhanced to support the optimization discussed earlier. In addition, the GSMP messaging between the switch server and GSMP provided some incremental improvements, as discussed above. Considering the enhanced code, the handoff latency was reduced to 56 ms with bundling and 67 ms without bundling for two flows, showing a 16 percent improvement. In contrast, the handoff latency for ten flows was 155 ms and 420 ms with and without bundling, respectively, showing a 63 percent improvement.

Mobile Soft-State Analysis — This experiment demonstrates the ability of mobile devices to adapt their bandwidth needs to changes in wireless and mobile QoS based on mobile soft-

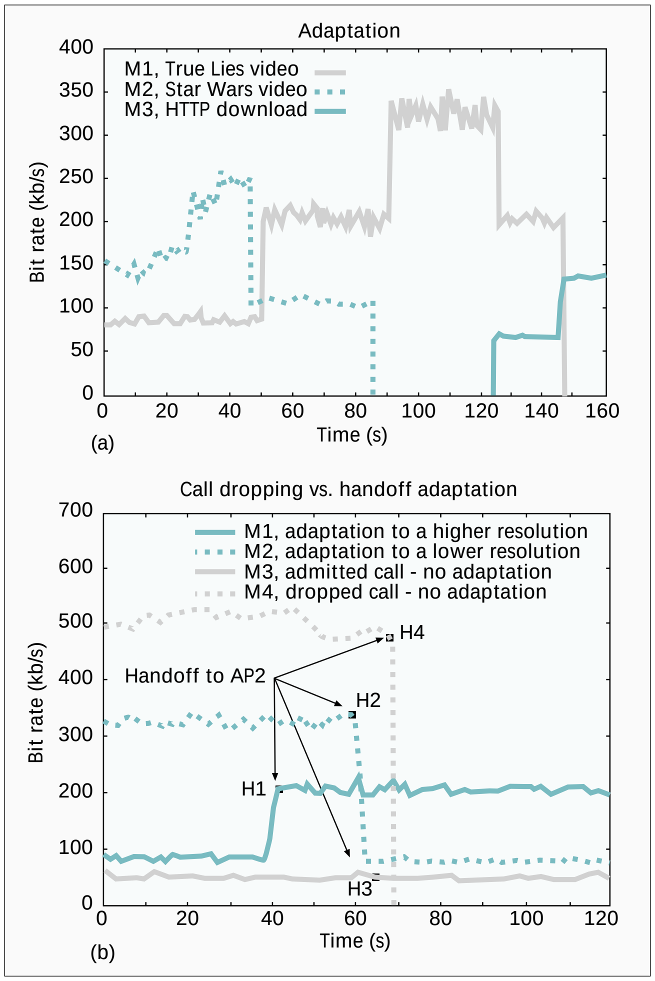


Figure 11. *(a) QoS adaptation within a single cell; (b) handoff-driven QoS adaptation.*

state. Mobile devices periodically probe and adapt to changes in available resources in wireless access networks. Users characterize flows using an adaptive QoS API (described earlier) that includes a utility function and an adaptation policy. In this experiment we present two scenarios that illustrate the benefit of mobile soft-state and QoS adaptation management in wireless and mobile environments.

The first scenario, illustrated in Fig. 11a, shows the QoS adaptive behavior of two mobile devices, M1 and M2, operating within a single wireless cell. Mobile devices M1 and M2 receive the "True Lies" and "Star Wars" video streams, respectively. Both video flows are based on discretely adaptive utility functions (i.e., multiresolution flows). Initially, M1 receives a base layer (BL) at 80 kb/s, and M2 a base and enhancement layer (E1) at 150 kb/s. Currently, the adaptive QoS service gives priority to support the minimum bandwidth requirements of multiresolution flows [27]. During the scenario, M2 registers an increase in bit error rate as it moves away from its current access point. Adaptive FEC is applied to the video between the access point and M2 based on the observed signal-to-noise ratio (SNR) and the measured bit error rate. An adaptive FEC object selectively codes the base and enhancement layers of the "Star Wars" video, increasing the bandwidth consumed by M2 from 150 kb/s to 250 kb/s. For the experiment, the maximum capacity of the air interface is set to 330 kb/s, and approximately 45 s into the scenario the M2 video is adapted back to the base layer with FEC only. Resources released by M2 are consumed by M1 increasing its utility at 50 s[8] into the scenario. This situation remains con-

[8] *Mobile device M1 probes and adapts to the available bandwidth within a single refresh interval that is currently set to 10 s. There are a number of trade-offs in setting the probe interval. A smaller duration allows mobile devices to aggressively grab resources on a fast timescale. However, this increases the signaling load overhead. Currently, we are investigating the coupling of the probing and adaptation timescales to the application-level adaptation policy [16].*

stant until M2 handoffs to a new access point after 80 s, allowing the access point to deliver another enhancement layer to M1. Note that mobile-initiated adaptation to released resources (i.e., scaling up) is somewhat dependent on the refresh/probe interval. When a new mobile device, M3, enters the cell after around 120 s, mobile device M1 explicitly scales back by dropping an enhancement layer. Toward the end of the scenario M3 probes and scales up to a better perceptible quality as M1 hands off to a new access point. At 140 s into the scenario, mobile device M3 sets up a new flow to access Web services, downloading a GIF file at a rate of 70 kb/s scaling up to 135 kb/s. In related work [16] we are investigating a generalized adaptation policy mechanism where applications can specify application-specific adaptation semantics. For example, some applications would not wish to experience the adaptation observed by M1, while others may be as aggressive as M3 in exploiting any available resources.

The second experiment highlights a number of different QoS adaptation scenarios that can take place during handoff. In this experiment, mobile devices hand off to access point AP2 from AP1 and AP3. Here, however, QoS adaptation is not based on the mobile soft-state refresh mechanism described and evaluated in the previous section. Rather, as part of the QoS renegotiation phase during handoff, mobile devices scale their QoS needs to match the available resources. The handoff point at which each of the four mobile devices (M1, M2, M3, M4) enter the new access point, AP2, is illustrated in the trace shown in Fig. 11b. The type of adaptation that takes place after handoff points, marked H1 through H4, is illustrated. During handoff a number of adaptation scenarios may occur, depending on the available resources and the ability of existing mobile devices to adapt. For example, the new access point may force existing mobile devices to drop enhancement layers to allow a new mobile to enter the cell with minimum QoS assurances. In this experiment, mobile device M1 enters the new cell at H1 and scales up its utility to take advantage of available resources. M1's adaptation policy is to only scale after handoff. At point H2 mobile device M2 hands off to access point AP2 and is forced to scale down to its base layer. Mobile device M3 has an adaptation policy of never adapting. At H3 the mobile 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 this instance, insufficient resources are available to support the base layers of M1, M2, M3, and M4, forcing the access point to deny the handoff.

Discussion

We have analyzed the performance of mobiware's QoS-controlled handoff, flow bundling, and mobile soft-state algorithms. While the baseline code raised some initial performance concerns about the viability of using distributed CORBA object technology for controlling mobile networks, the enhanced software is extremely competitive in relation to existing work. The latency measured for QoS-controlled handoff was reduced from 171 ms to 41 ms for the handoff of a single flow through two ATM switches making handoff through a single switch in the order of 20 ms.

While it is difficult to compare results from different testbeds running different signaling systems, we highlight some measurements from comparable systems found in the literature for the purpose of qualitative comparison only. In [28] and [29] handoff latencies were measured to be 10 and 30 ms for a single flow through a single crossover switch. The use of flow bundling techniques in mobile networks shows great performance benefit as the number of flows increase with handoff. The handoff latency for ten flows is 155 ms when flow bundling is enabled and 420 ms when disabled. This clearly shows the advantage of such aggregation techniques. Mobiware's flow bundling compares favorably to the literature. In [29] Mishra reported a handoff latency of 125 ms for ten flows using native ATM signaling code. Mobile soft-state also exploits aggregation techniques provided by flow bundling. This allows resource probing to be based on flow bundles rather than per-flow. In this article QoS adaptation techniques clearly demonstrate the benefit of mobile soft-state in sharing resources among competing mobiles in a cell.

Conclusion

In this article we discuss the design, implementation, and evaluation of an open programmable mobile network based on distributed object technology called mobiware that dynamically exploits the intrinsic scalable properties of adaptive mobile applications. A number of researchers have applied distributed object technology to mobile systems. Our work, however, differs from these efforts. First as part of the open signaling community [30] we are deeply interested in identifying open programming interfaces for mobile and wireless networking. In this work we have identified a number of objects, APIs, and algorithms that provide QoS support for adaptive mobile networking. The mobiware technology we have developed over the last two years marks a considerable software effort. To our knowledge we are one of the first groups to apply distributed object technology as a mobile middleware solution for adaptive mobile networking. Mobiware objects execute on the mobile devices, at the access points, and on switch/routers, exposing open APIs that can be programmed to support mobile signaling, QoS adaptation management, and wireless transport. We observe that once the wireless and mobile APIs have been designed, the programming of new network algorithms (e.g., QoS-controlled handoff, flow bundling, and mobile soft-state) is straightforward engineering. The source code distribution for mobiware v. 1.0 can be freely downloaded from [13] for experimentation.

Acknowledgments

First, we would like to thank the Comet Group's industrial participants for their kind support. We would also like to thank Aurel A. Lazar and the xbind team for providing the xbind broadband kernel on which mobiware is built. Also, we would like to thank the following colleagues for their major contributions toward the implementation of mobiware: Oguz Angin implemented flow bundles, Anand Balachandran implemented the active media filters, Javier G. Castellanos retooled the beacon to include QoS hints, Michael E. Kounavis implemented the transport system, Raymond Liao implemented the signaling system, Chien-Ming Yu implemented the active error control, and finally, Yasuro Shobatake (Toshiba Corp. Japan) implemented the wireless transport management. We would also like to thank Laura Zhou for implementing mobiman and Mun Choon Chan (Lucent Technologies), who implemented the xbind connection management system from which the mobiware mobility agent grew. Finally, we would like to thank Lucent Technologies for providing a beacon API.

References

[1] R. H. Katz, "Adaptation and Mobility in Wireless Information Systems," *IEEE Pers. Commun.*, vol. 1, no. 1, 1st qtr. 1994.
[2] Panel, "QoS in the Next Generation Mobile Internet: What Is Feasible?," chaired by A. T. Campbell, 3rd Annual ACM/IEEE Int'l. Conf. Mobile Comp. and Networking (MobiCom '97), Budapest, Hungary, Oct. 1997.
[3] K. Lee, "Adaptive Network Support for Mobile Multimedia," *Proc. Mobicom '95*, Berkeley, CA, Nov. 1995.

[4] Daedalus/BARWAN project at UC Berkeley; http://daedalus.cs.berkeley.edu/index.html
[5] M. Satyanarayanan, "Mobile Information Access," *IEEE Pers. Commun.*, vol. 3, no. 1, Feb. 1996.
[6] B. Zenel and D. Duchamp, "A General Proxy Filtering Mechanism Applied to the Mobile Environment," *Proc. Mobicom '97*, Budapest, Hungary, Sept. 1997.
[7] S. Lu, K.-W. Lee, and V. Bharghavan, "Adaptive Service in Mobile Computing Environment," *Proc. 5th IFIP Int'l. Wksp. QoS* (IWQoS '97), New York, May 1997, pp. 25–36.
[8] M. Naghshineh and M. Willebeek-LeMair, "End-to-End QoS Provisioning in Multimedia Wireless/Mobile Networks," *IEEE Network*, Mar. 1997.
[9] A. A. Lazar, "Programming Telecommunication Networks," *IEEE Network*, Oct. 1997.
[10] D. L. Tennenhouse *et al.*, "A Survey of Active Network Research," *IEEE Commun. Mag.*, vol. 35, no. 1, Jan. 1997, pp. 80–86.
[11] A. T. Campbell, "Mobiware: QoS-Aware Middleware for Mible Multimedia Communications," 7th IFIP Int'l. Conf. High-Perf. Networking, White Plains, NY, Apr. 1997.
[12] O. Angin *et al.*, "Open Programmable Mobile Networks," *Proc. 8th Int'l. Wksp. Network and Op. Sys. Support for Digital Audio and Video* (NOSSDAV), Cambridge, U.K., July 1998.
[13] Mobiware v. 1.0 Source Code Distribution: http://comet.columbia.edu/mobiware
[14] xbind Broadband Kernel code: http://comet.columbia.edu/xbind
[15] OMG, The Common Object Request Broker: Architecture and Specification, Rev. 1.2, Dec. 1993.
[16] G. Bianchi, A. T. Campbell, and R. R.-F. Liao, "Supporting Utility-Fair Adaptive Services in Wireless Networks," *Proc. 6th Int'l. Wksp. QoS* (IWQoS '98), Napa Valley, CA, May 1998.
[17] A. Balachandran, A. T. Campbell, and M. E. Kounavis, "Active Filters: Delivering Scalable Media to Mobile Devices," *Proc. S7th Int'l. Wksp. Network and Op. Sys. Support for Digital Audio and Video* (NOSSDAV), St. Louis, MO, May 1997.
[18] G. Bianchi and A. T. Campbell, "A Programmable MAC," to appear, *Proc. ICUPC '98*, Florence, Italy, Oct. 1998.
[19] M. S. Corson and A. T. Campbell, "Toward Supporting QoS in Mobile Ad hoc Networks," work in progress session, 1st IEEE OPENARCH '98, San Francisco, CA, Apr. 1998.
[20] S-B Lee and A. T. Campbell, "INSIGNIA: Inband Signaling for Mobile Ad Hoc Networking," *Proc. Int'l Workship Mobile Multimedia Commun. (MOMUC '98),* Berlin, Germany, Oct. 1998.
[21] P. Newman *et al.*, "Ipsilon's General Switch Management Protocol Specification," IETF RFC 1987, Aug. 1996.
[22] O. Angin *et al.*, "Enabling the Creation, Control and Management of Adaptive Wireless Services in Programmable Mobile Networks," Ctr. for Telecommun. Res. tech. rep. submitted for publication, June 1998.
[23] J. Porter *et al.*, "The ORL Radio ATM System, Architecture and Implementation," ORL tech. rep., 1995.
[24] D. Clark and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," *Proc. Sigcomm '90*, Sept. 1990.
[25] A. T. Campbell and G. Coulson, "QoS Adaptive Transports: Delivering Scalable Media to the Desktop," *IEEE Network*, Mar. 1997.
[26] J. E. Van der Merwe and I. Leslie, "Switchlets and Dynamic Virtual ATM Networks," *Integrated Network Management V*, A. A. Lazar, R. Saracco, and R. Stadler, Eds., New York: Chapman and Hall, 1997.
[27] A. T. Campbell, D. Hutchison, and C. Aurrecoechea, "Dynamic QoS Management for Scalable Video Flows," *Proc. 5th Int'l. Wksp. Network and Op. Sys. Support for Digital Audio and Video* (NOSSDAV), Durham, NH, 1995.
[28] J. Naylon, "Radio Handover Measurements," OPENSIG Spring '97 Wksp. Open Signaling for ATM, Internet and Mobile Networks, Cambridge, U.K., Apr. 1997.
[29] P. Mishra, "Implementation and Experimental Evaluation of Mobility-enhanced ATM Signaling," OPENSIG Fall '97 Wksp. Open Signaling for ATM, Internet and Mobile Networks, New York, Oct. 1997.
[30] http://comet.columbia.edu/opensig

Biographies

OGUZ ANGIN (comet.columbia.edu/~angin) is currently working at COMSAT Laboratories, Maryland. He received his M.S. degree in electrical engineering from Columbia University in 1998, and his B.S. degree in electrical and electronics engineering at Bosphorus (Bogazici) University, Turkey, in 1996. His research interest is in wireless and mobile networking.

ANDREW T. CAMPBELL (comet.columbia.edu/~campbell) joined the E.E. faculty at Columbia as an assistant professor in January 1996 from Lancaster University, where he conducted research in multimedia communications as a British Telecom Research Lecturer. Before joining Lancaster University, he worked for 10 years in industry focusing on the design and development of network operating systems, communication protocols for packet-switched and local area networks, and tactical wireless communication systems. He is a member of the COMET Group at Columbia's Center for Telecommunications Research, where he is conducting research in wireless media systems (comet.columbia.edu/wireless). His current research interests include the development of programmable mobile networks, cellular IP networks, and programmable router technology.

MICHAEL E. KOUNAVIS (comet.columbia.edu/~mk) is a Ph.D. student at COMET Group, Columbia University. He received his Diploma in electrical and computer engineering from the National Technical University of Athens (NTUA), Greece, in 1996, and the MSc. degree from Columbia in 1998. Until now he has been actively involved in the areas of network programmability and multimedia transport over wireless/mobile networks.

RAYMOND R.-F. LIAO (comet.columbia.edu/~liao) joined the COMET Group, Columbia University in 1996 as a Ph.D. student and graduate research assistant. Before that, he worked at Newbridge Networks Corporation, Canada, for three years on ATM product performance analysis and traffic management. He received the M.A.Sc. degree in fault-tolerant ATM switch design and queueing analysis from the Dept. of Electrical and Computer Engineering, University of Toronto, Canada, in 1993, and the Bachelor degree from Huazhong University of Science and Technology, China, in 1990. His current research focuses on realizing adaptive QoS in wireless/mobile multimedia networks with middleware methodologies, including distributed computing and network economics.