A Software Support Infrastructure for Wireless Access Routers

Petros Zerfos, *Student Member, IEEE*, Gary Zhong, and Songwu Lu, *Member, IEEE*

***Abstract*—Routers are expected to play an important role in the Internet protocol-based wireless data network. Although a substantial number of adaptive and intercell coordination techniques have been proposed to improve wireless network performance under dynamic wireless channel conditions and host mobility, a system support framework is still missing. In this paper, we describe DIRAC, a software-based router system that is designed for wireless networks to facilitate the implementation and evaluation of various channel-adaptive and mobility-aware protocols. DIRAC adopts a distributed architecture that is composed of two parts: a router core (RC) shared by the wireless subnets, and a router agent (RA) at each access point/base station. RAs expose wireless link-layer information to the RC and enforce the control commands issued by the RC. This approach allows the router to make adaptive decisions based on link-layer information feedback on both data and control planes. It also permits the router to enforce its policies (e.g., policing) more effectively through underlying link-layer mechanisms. It further enables interaccess-point coordination at the RC. As showcases, we implement under DIRAC the prototypes of three wireless network services: link-layer assisted fast handover, channel-adaptive scheduling, and link-layer enforced policing. Our implementation and experiments show that our distributed wireless router provides a flexible framework, which enables advanced network-layer wireless services that are adaptive to channel conditions and host mobility.**

***Index Terms*—Communication system operations and management, communication systems, software, wireless LAN.**

I. Introduction

ROUTERS have been a key architectural component in the Internet protocol (IP)-based wired Internet. As wireless data services become increasingly popular and wireless networks move toward an IP-based paradigm, routers are expected to play an equally important role in the emerging wireless Internet era.

The main functionality of a router is data-plane packet forwarding and control-plane routing and management [1], [2]. The evolution of the Internet has also made a case for new services such as packet filtering, intrusion detection, level-n switching, and packet tagging [2]. Despite all such advances, current router systems do not work well in wireless networks because they do not consider the issues of wireless links and host mobility.

In recent years, a large number of protocols [3]–[6] have been proposed to achieve the above goal. At the core of such protocols link-layer information feedback is employed, e.g., channel errors and link handoff events, to make channel-adaptive and mobility-aware decisions. They also enable intercell coordinations, e.g., in the context of multimedia support and fast handover management. These solutions have been carefully analyzed and evaluated through simulations. However, they have not been able to materialize in practice. The key missing piece is a system framework that facilitates the real implementation, experimental evaluation, and field deployment of such new solutions. Developing a router system that provides such a system framework is the subject of this work.

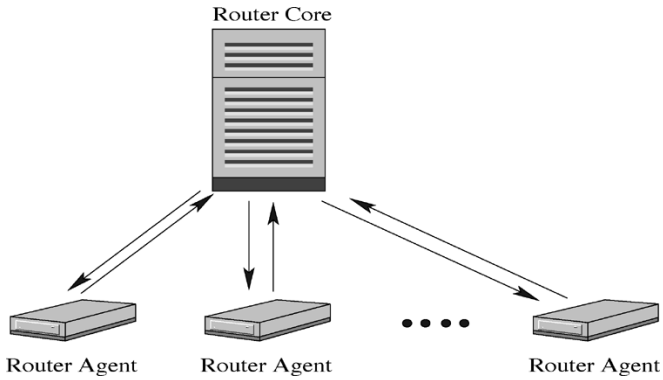

Fig. 1. Distributed wireless router architecture.

This paper describes the design and implementation of DIRAC,[1] a software-based wireless router system. DIRAC currently works with IEEE 802.11b, but its design principles apply to other wireless technologies such as 802.11a, 802.11g, and wide-area third-generation (3G) and fourth-generation (4G). The key technical innovation of DIRAC is a distributed router architecture (shown in Fig. 1), consisting of a router core (RC), and several router agents (RAs), each of which runs on an 802.11 access point (AP). The RC is shared by the wireless subnets and carries the main functionality of a router. It interacts with each RA through a lightweight communication protocol. Each RA collects and reports link-layer information back to the RC, and accepts and enforces router policies such as policing via link-layer mechanisms (e.g., authentication/reauthentication of 802.11 standard). The resulting benefits of such a design are twofold.

1) It enables the router to make *informed, adaptive* packet forwarding decisions on the data plane and improved mobility management on the control plane, based on link-layer information feedback. It also enables the router to perform intercell coordinations between APs.

Manuscript received February 2, 2004; revised January 30, 2005. An earlier version of this paper appeared in ACM MOBICOM'03, San Diego, CA.

The authors are with the Computer Science Department, University of California, Los Angeles, CA 90095-1596 USA (e-mail: pzerfos@cs.ucla.edu; gzhong@cs.ucla.edu; slu@cs.ucla.edu).

Digital Object Identifier 10.1109/JSAC.2005.845633

[1]DIRAC denotes distributed router architecture for wireless networks.

2) It allows the router to directly execute its network-level policies via the exposed link-layer mechanisms.

As a result, DIRAC gains from the enhanced interactions with the link layer and facilitates the implementation, evaluation and deployment of wireless-specific protocols and services.

The distributed architecture of DIRAC is motivated by the fact that link-layer information, on per host basis, is location dependent and only accessible at each AP. Moreover, DIRAC stays in the middle ground of two extreme architectural alternatives: the intelligent AP approach that provides adaptive link-layer operations at each AP but remains network-layer oblivious, and the ubiquitous router approach, which converts every AP into a full-blown router, and enables all network-layer functions at the cost of increased complexity in management and administration. The DIRAC design allows for cost-effective and rapid development of wireless services by upgrading software only at the RC, without modifying the hardware and/or software of each AP. It facilitates monitoring and configuration of the network, since most complexity is placed on the RC, and enhances robustness by keeping each AP simple. Last, DIRAC not only simplifies the implementation and evaluation of adaptive, wireless protocols designed for a single cell, but also permits the deployment of intercell coordinated resource management protocols.

With the architecture in place, we have implemented several prototype services to demonstrate its viability and practicability. The three services implemented are wireless and mobility specific, and include a link-layer informed, fast handover protocol on the control plane that minimizes transient packet losses during handoffs, a channel-adaptive, FEC-based, packet forwarding solution that solves the head-of-line (HoL) blocking issue over wireless links, and a link-layer assisted policing that penalizes overly aggressive, wireless clients. The list of these three example protocols demonstrates the wide spectrum of wireless router services enabled by DIRAC. Other services, such as inter-AP coordinated resource management, mobility-aware firewall, adaptive wireless scheduling, and cross-cell quality-of-service (QoS) support for voice over IP (VoIP), can also be implemented.

The implementation of DIRAC architecture and services is developed using off-the-shelf hardware and open-source software. The RC is written in Click [7], an extensible and modular software router framework, and runs on a commodity PC. The RAs are implemented on the instant AP platform [8]. Our experiments show that the system is able to support a large number of APs, as well as clients per AP using commodity hardware: processing statistics reports from 50 APs requires only 140 μs, and it takes less than 1 μs to process channel states for 50 clients per AP.

The rest of this paper is organized as follows. Background introduction is provided in Section II. Motivation for DIRAC is given in Section III. The design and implementation of the basic DIRAC framework are given in Sections IV and V, respectively. Section VI describes three prototype services for wireless routers. Section VII evaluates the system performance of DIRAC. Discussions and lessons learned are provided in Section VIII. Section IX compares with related work and this paper is concluded in Section X.

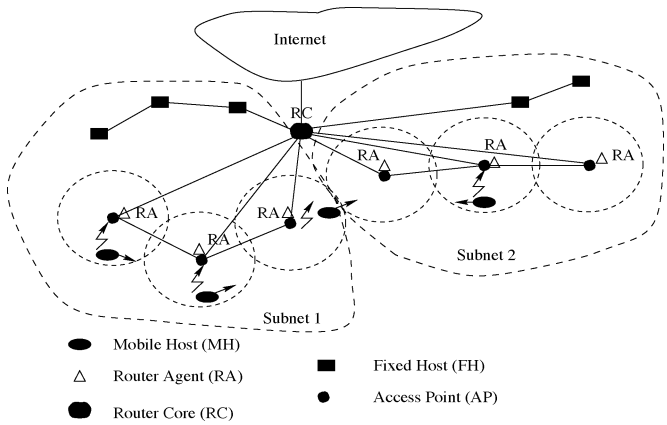

Fig. 2. Packet-switched 802.11b subnets.

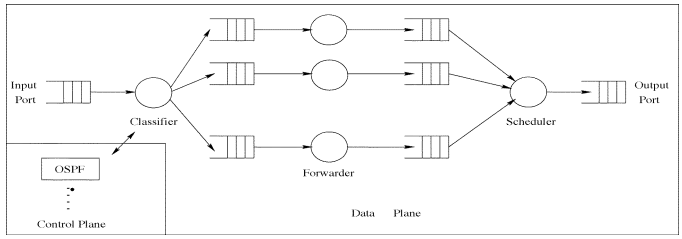

Fig. 3. Wired router.

II. Background

The target scenario is a packet-switched wireless data network, based on the 802.11b wireless access technology (Fig. 2). In a typical enterprise/campus environment, mobile hosts access the Internet via the AP in each cell over the wireless channel, and APs are interconnected via the wired backbone. Multiple cells form a subnet, several subnets together cover the entire enterprise/campus and are interconnected via an access router. Mobile hosts may roam between adjacent cells, possibly across different subnets. Packet transmissions can be downlink (from the network to a mobile host), or uplink (from a mobile host to the network). The scenario we focus on—a typical one in practice—assumes that the wired network has sufficient bandwidth, and the wireless link poses as the bottleneck.

A typical wired router (shown in Fig. 3) consists of a data-plane forwarding engine that forward packets, and a control plane where signaling protocols run. In the forwarding engine [1], a classifier first reads packets from an input port, and based on certain fields in the packet header, selects a forwarder to process the packet and send it to the output queue. Finally, an output scheduler selects one of its output queues and transmits the packet to the output port. The control plane of a wired router executes signaling protocols like OSPF and LDP and assists the packet forwarding engine in the process of packet delivery.

The wired router system does not work well in a wireless network. The fundamental problem is that it does not consider the issues of dynamic wireless link quality and host mobility. The resulting performance degradation is well documented in the literature, in the context of wireless scheduling, adaptive error control [6], micromobility management [9], and admission control, to name a few.

III. Design Rationale

A. Motivation

In recent years, applications such as carrier-grade data delivery and VoIP have called for new wireless protocols for their effective operations. These networking solutions include mobility-aware admission control, wireless packet scheduling, and micromobility management protocols, to name a few. The fundamental principle behind these wireless solutions is to adapt to wireless channel dynamics and also be mobility aware. This can be achieved through interactions between the edge router and the link-layer at the APs. A system support framework that allows for this interaction enables the implementation and deployment of these wireless protocols.

The aforementioned interaction also facilitates intercell coordination, since information on each AP is available at a centralized edge router. Intercell coordination is important for two reasons. First, it is used to manage resource dynamics in both the time and the spatial domains, incurred by roaming users. For example, bandwidth reservations for video transmissions should be made by the new AP right after a handoff. Second, intercell coordination facilitates the management of large wireless networks, in areas such as load balancing, transmission power control for interference minimization, and dead cell-detection [11].

DIRAC also assists on security. By enhancing interactions between each AP and the access router, the router can readily enforce policies and correlate usage patterns between APs to detect attacks across cells. Policies can also be applied using mechanisms available both at the medium access control (MAC) layer of the APs and the network layer of the router, thus avoiding to configure of every individual device.

Finally, recent trend in Internet edge router design advocates the use of software-based frameworks, which provide extensibility and flexibility. The software-based router accelerates the implementation, experimentation and deployment of new network protocols and algorithms. The potential performance penalty can be offset by combining this approach with network processor cards. DIRAC also follows this trend and takes the software-based router design approach. In some sense, flexibility of the access router system is more important than the mere data-forwarding speed in the wireless domain because of the fundamental limit of wireless channel capacity.

B. Architecture Alternatives

Fig. 4 shows the spectrum of architecture alternatives for enterprise wireless routers [11], along with the positioning of DIRAC. The x axis of the diagram represents the level of integration among the devices that implement the various layers of the network protocol stack, while the y axis shows the control plane information. Each rectangle corresponds to one architecture instantiation.

From the above diagram, "Autonomous AP" is a layer-2 device that manages control plane information only from the link-layer (layer-2). Similarly, the "Traditional router" is a layer-3 entity that handles layer-3 control plane information, and "Base Station" is essentially the antenna with the wireless PHY. These designs operate obliviously of one another, in the sense that information at one layer (e.g., link-layer management

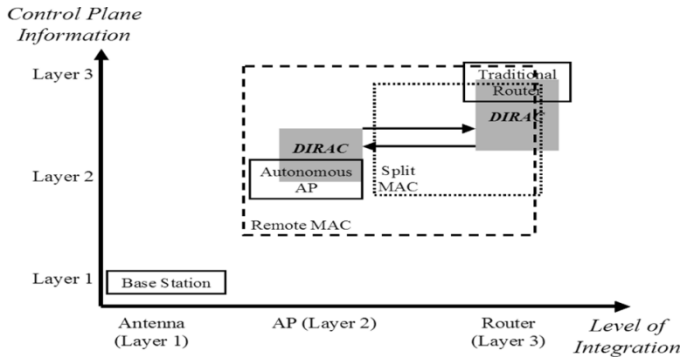

Fig. 4. Design spectrum of wireless router architectures.

frames) is not made available to another (e.g., the router). Besides suffering from severe performance degradation, as exemplified by the HoL blocking problem over the wireless channel (Section VI-B), this approach does not allow the deployment of new wireless services that require aggregate information collected from multiple cells. In contrast to the oblivious approaches, the "split MAC" architecture is a layer-2/3 device, since it also handles frames of the non-real-time management functionality of 802.11, implementing layer-2 and layer-3 functionality on one device, while the real-time aspects of 802.11 are left to the AP. "Remote MAC" takes one further step by providing even the real-time functionality of 802.11 MAC, deteriorating the AP to being just an antenna. These design alternatives are tightly integrated, and interoperability issues among different network access technologies such as 802.11, 802.16, and 3G, also arise, making the integration of these technologies more complex.

One the other hand, DIRAC is implemented as two distinct components that interact with each other. One component is responsible for managing the link-layer frames specific to the network access technology used, while the other is relatively network-access agnostic, and deals with the network layer services. From the above, it is clear that DIRAC arbitrates the architectural options previously discussed. It seeks to preserve the best features and avoid the drawbacks of each approach, and balance between performance, complexity, cost, and manageability.

IV. Design

A. Overview

DIRAC adopts a distributed router architecture that consists of two main software components: a generic RC, and multiple lightweight, network-specific RAs. The architecture is distributed in the sense that each RA runs at an AP, while the RC runs on a commodity PC. They collaborate to provide router functions and services—data-plane packet forwarding and control-plane management—to each mobile host. The RC interacts with the wireless link layer through each RA, without changing the functionality of the IP layer. Communication between the RC and the RAs is carried out via standard user datagram protocol (UDP) sockets over the wired backbone, with each RA serving as a messenger between the RC and the link-layer device driver. The interaction between RC and RA is

TABLE I
EVENTS, ACTIONS, AND STATISTICS

Events	*Actions*	*Statistics*
New host (Association)	Accept/Reject Host Association	Signal-Noise Ratio (dB)
Host Leaves (Disassociation)		Channel Quality Variation
Security Binding (Authentication)	Authentication Deauthentication	Frame Retxmits (frame loss %)
Roaming host (Reassociation)	Set Maximum Retransmit Count	CommTallies (detailed statistics)
Power Saving mode (PS)	Channel Assignment	
	Transmit Power Control	Latency
Dead Cell Detection (Heartbeat)	Set Transmission Rate	Per-host Average Throughput
	Image Download	

constrained; it takes three forms: *events*, *statistics*, and *actions*, to be elaborated in the next section.

B. Forms of Interaction

Table I illustrates the typical events, statistics and actions that are provisioned by the DIRAC router for the 802.11b technology. It shows a representative set of primitives supported by most wireless access technologies, even though the current implementation is 802.11-specific, as well as functionality required for the management task of the multiple AP devices from a centralized point (i.e., the access router).

Events denote occurrences of asynchronous link-layer activity in the cell, detected by the AP and reported back to the RC control engine. For example, a reassociation event informs on a layer-2 handoff by a roaming host. DIRAC acts upon events to make mobility-aware decisions. Moreover, it gets informed of the status of each AP device, a piece of information that can be used for management purposes, such as replacing malfunctioning hardware.

Statistics report the latest information on channel quality that each host perceives, which is location-dependent. Such information can be expressed in link-layer frame loss percentage, or even signal-to-noise ratio (SNR) in decibels. DIRAC uses such statistics to make channel-adaptive packet delivery via its renovated scheduler and forwarders. Furthermore, detailed information for each mobile node (MN) may be reported in the form of *CommTallies*, in order for example to investigate difficulties in communication with a particular host.

Actions are requested by RC in order to enforce its policies. RAs, which receive these requests, execute them at each AP's link-layer, either by tuning the parameters of the 802.11b MAC protocol, or by adjusting the configuration settings of the driver. For example, *Deauthentication* action will trigger the AP to deny the channel access by the target host; *Set_Retransmissions* action will ask the AP to reset the maximum retransmission count. Requests for management reasons sent to the APs from the RC can also be considered as actions. For example, the wireless network operator may need to refresh a new image to every AP, in order to fix bugs in the software, or add extra features. This can be considered as an action to be carried out by the RA. Another example is setting the transmission level for each AP, with respect to transmission power in neighboring cells, in order to minimize intercell interference.

C. Router Core (RC)

The RC renovates its data-plane forwarding engine and control-plane management protocols to make them wireless adaptive and mobility aware.

1) Control Plane: The control plane of DIRAC consists of two types of components: routing and management protocols, and a *control engine*. DIRAC does not mandate a specific choice of the former and leaves it to the users. The control engine in DIRAC is the OS support to enable the cross-layer interactions via *events*, *statistics*, and *actions*. It consists of four components: *EventProcessor*, *StatisticsMonitor*, *ActionProcessor*, and *RegistrationDB*.

EventProcessor accepts messages carrying events reported by each RA and notifies each interested party of such events. Any component on the data plane (e.g., the forwarder) or on the control plane (e.g., the fast handover module) can request and will be informed of a specific event upon its occurrence. This functionality is provided by a *callback* mechanism. Each module interested in an event provides a callback function handler to the EventProcessor, one for each event type. Upon arrival of an event notification, its registered callbacks are executed.

StatisticsMonitor provides a centralized repository to maintain the latest channel quality information for each host. Other elements of the router can query it, retrieve the updated link-layer statistics, and adapt their operations accordingly. The statistics information is updated periodically via messages reported by RAs to the RC over the wired connection. To minimize the memory overhead, we store only the latest received report, along with a timestamp.

ActionProcessor sends an *action_request* message to the appropriate RA, upon request from any module within the RC. It serves as an interface between the RC and the link-layer AP. It encapsulates each action request into a UDP message, and sends it to the corresponding RA.

Finally, *RegistrationDB* offers a registry to store information regarding each mobile host and its associated AP. Each host is assigned a unique ID throughout the router system. This way, even though different underlying network technologies may use different link-layer IDs, the router still treats them in a coherent way.

2) Data Plane: The data-plane forwarding engine allows components such as schedulers and forwarders to implement channel-adaptive protocols, based on link-layer feedback. DIRAC does not stipulate the specific choice of such protocols; many solutions in the literature may serve this purpose.

A distinctive feature of the forwarding engine is that it provides *asymmetric* operations for *uplink* and *downlink* flows (shown in Fig. 5). The downlink service is proactive and the service provided via the wireless scheduler or forwarder is fine grained with a time granularity of tens to hundreds of milliseconds. The uplink service is reactive and the service provided via policing is coarse grained with a time granularity of seconds.

The above design choice is motivated by the inherent wireless network constraint: the RC has complete control and accurate flow information (e.g., the arrival time and length of each

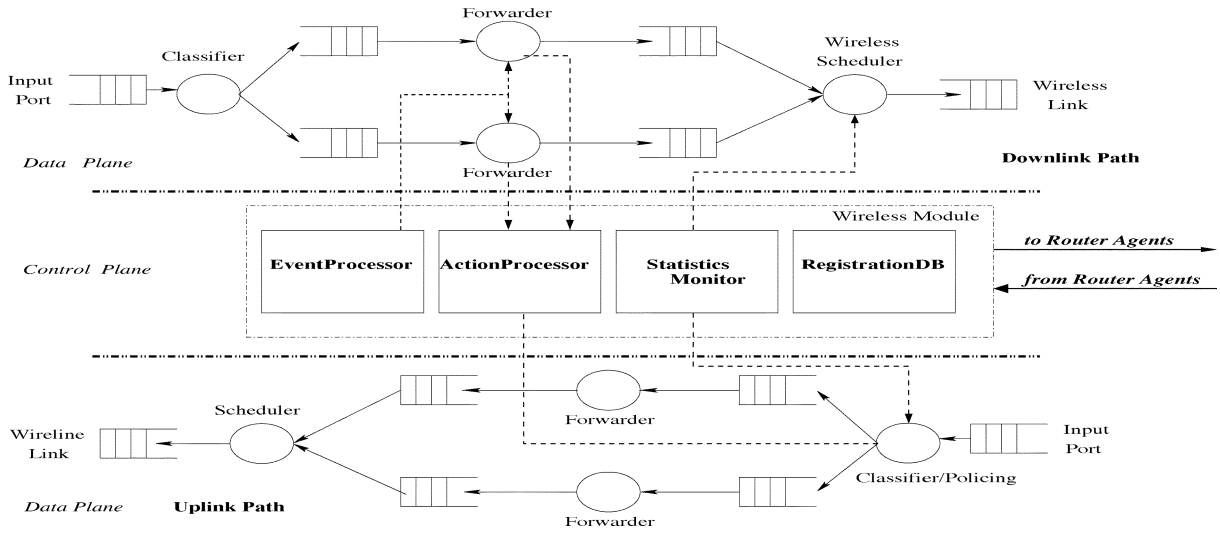

Fig. 5. Conceptual model of the RC.

packet) for downlink flows. However, flow information for uplink flows is spread at each mobile host. The RC does not have full control and accurate information for uplink flows. This approach also fits well with the wireless traffic characteristics, where 66% was shown to flow in the downlink direction.

D. Router Agent (RA)

The RA at each AP bridges the interaction between the RC and the wireless link layer. The goal is not to facilitate generic programmability of the AP, but to run a lightweight messenger between layer-2 of the AP and layer-3 of the access router. Thus, its tasks include the following.

1) Monitoring the state and channel quality for each of its associated hosts, as reported by the device driver.
2) Sending appropriate messages to the RC, when events occur in the cell.
3) Intercepting messages sent by the RC, requesting *action* messages from the AP device driver, and delivering other messages to the mobile host in the form of link-layer frames.

Link-layer statistics are collected for each mobile host by the RA periodically. The collection frequency reflects a tradeoff between state accuracy and communication overhead. A rough calculation shows that the frequency need not be high, since the channel coherence time (i.e., the period during which channel conditions does not vary much) is equal to the time needed for several packet transmissions [10].

E. Decomposition of Functionality and Architectural Features

DIRAC takes an asymmetric approach to functional partition between the RC and each RA. RAs are controlled by the RC to certain extent, while wireless network services are centrally implemented at the RC. Maximum programmability is allowed at the RC, but not at RAs. In addition, the RC is rather link-layer technology independent; when applied to a new link-layer technology, it can be kept intact and only updating the lightweight RAs is needed. This improves portability of the system to other wireless technologies.

The above approach paved the way to the design of the various forms of interaction that was presented in the previous sections, which was based on the *qualitative characteristics* of the communication that takes place between the RC and the RAs. More specifically, statistics and events represent periodic and asynchronous information respectively, which is collected at the RA and reported back to the RC. Actions are essentially the mechanisms that are made available by the RAs for use by the RC.

On the other hand, recent efforts within the IETF aiming to standardize forwarding and control devices and elements in the wireless and wired domain (work-in-progress drafts of control and provisioning of wireless access points (CAPWAP) architecture and ForCES framework [12]) take on slightly different approaches. While both efforts describe possible approaches for splitting the functionality among the multiple physical devices that form the infrastructure of a network, the former protocol is mainly focused on resolving interoperability issues among these devices, which arise due to the different ways of performing such separation of functionality.

The second proposal, the generic ForCES protocol, while it does not explicitly consider routers and base-stations in an 802.11 network setting, it specifies a master–slave relation between a control element (CE) and multiple forwarding elements (FEs), with the latter "...performing all packet processing functions as directed by CEs, with no initiative of their own." This essentially leaves all management and control decisions to control elements that could also perform 802.11-related management functionality, similar to the wireless module of the RC in DIRAC. Likewise, the forwarding elements correspond to the RAs in our design, and have no notion of redundancy, load sharing, distributed control, or other, more advanced services.

The cost saving provided by DIRAC comes from "economies of scale" in two aspects: hardware investment, and "soft-cost" associated with management and administration. The former is relatively easy to assess, while the latter is harder to gauge quantitatively. For the hardware, only upgrading the access router is needed in DIRAC each time a new service requires more computational power. This can be done by using network processor

TABLE II
ELEMENT GLOSSARY

Class	*Element*	*Description*
Control Engine	EventProcessor	Processes event-messages from RAs; invokes callback functions of other elements
	ActionProcessor	Encapsulates actions requested by other elements into UDP messages sent to RA
	IP6L2Statistics	Stores statistics about each mobile host. Statistics are sent as UDP messages by RA
	RegistrationDB	Stores matchings between hosts and RAs; stores addresses, and assigns IDs
Mobility Management	IP6MobilityNewAR	Implements the micro-mobility management protocol –new AR (Section VI.A)
	IP6MobilityOldAR	Implements the micro-mobility management protocol –old AR (Section VI.A)
	IP6Tunnel	Implements IPv6-IPv6 tunneling as virtual interfaces
	IP6Buffer	Buffers packets, as it waits for a Fast Neighbor Advertisement message to arrive
	IP6FNDSolicitor	A neighbor solicitor for IPv6 that also handles Fast Neighbor Advertisements
	IP6RAdvertiser	Sends Router Advertisements, both solicited and unsolicited
Downlink path	FECEncoder	Performs packet FEC enconding on k packets, producing $n - k$ redundant ones
	FECController	Controls the *FECEncoder* by setting parameters k and n according to the channel quality
Uplink path	PolicingMeter	Police aggressive uplink flows

cards. For example, using the Intel IXP 1200 network processor cards for a moderate U.S. $700 per piece, one can parallelize operations and aggregate services, while providing high performance (up to 5 Gb/s). On the other hand, upgrading each AP can be much more expensive if one does not use DIRAC. Though each AP typically costs about U.S. $150, one has to take into account that a large number of them (in the order of 50–100 per access router, according to a study by Juniper Research [13] for campus-sized environments) will have to be upgraded. Moreover, since the centerpiece is placed at the RC in DIRAC, centralized management for subnets is made possible, which in turn reduces the administration overhead.

V. IMPLEMENTATION

This section describes the implementation of the DIRAC prototype, along with the way RC and RAs work together in the specific hardware and software environment.

A. Router Core (RC)

The RC works with the IPv6 protocol on a commodity PC. We choose IPv6 as our working protocol environment because its extensibility and inherent mobility support have been well articulated in the literature [14]. In fact, the cellular network industry has been favoring mobile IPv6 over mobile IPv4. Our implementation is therefore based on IPv6, but it can be easily ported to IPv4.

RC is implemented as a set of *Click elements* within the Click router framework [7], under Linux. We chose Click as the implementation platform for the RC, since its extensible and modular architecture allows for rapid prototyping. Click also provides standard functionality of the forwarding path, such as routing table and address resolution. For DIRAC, we developed 13 new Click elements (Table II provides a glossary, along with a short description) that implement the desired functionality. Unlike most existing Click elements, they are shared and comparatively large, due to the complicated operations that they carry out.

Fig. 6 shows a two-interface (one wireless, and one wired) RC configuration with the Click elements. *IP6FastHandover* and *DownlinkScheduling* are two compound Click elements to be described in Section VI. The downlink and uplink packet forwarding engines are shown in dotted lines in the figure.

The *downlink* forwarding engine operates as follows. Incoming packets from the device interface go through the 802.1d frame classifier *802Classifier* to check whether it is an IPv6 neighbor discovery message, followed by the neighbor discovery classifier *NDClassifier* to check whether it is a router advertisement message. The 802.1d header is stripped by the *Strip802Header*, and the remaining IPv6 header is examined by *CheckIP6Header*. *DLookupIP6Route* performs routing table lookup for the packet, *HopCountDec* decrements its hop count, and passes it on to the *FNDSolicitor* which handles the address resolution. It then goes through the *Downlink Scheduler* and finally is delivered to the interface where the AP/RA is connected. The *uplink* forwarding engine works similarly except that it goes through an *Uplink Policing* element first.

1) API: The aforementioned wireless network services, as well as new ones to be deployed in the future, make use of an API provided by the control engine of the RC. Each service consists of its own elements (in the Click framework), but all services use the same API to interact with the RC.

Taking actions: Elements take link-layer actions using the API provided by the `execute` call of Fig. 7(a). The *execute* call takes input on which host requests the call, the type of action requested, and the parameters for the action type, and returns a status code of failure/success. The *ActionProcessor* element, which actually implements the action, looks up the requesting host's link-layer address and the respective RA using the API provided by *RegistrationDB* of Fig. 7(e).

Notifying elements: In order for elements to be notified upon an event that happened in the cell, they must register first

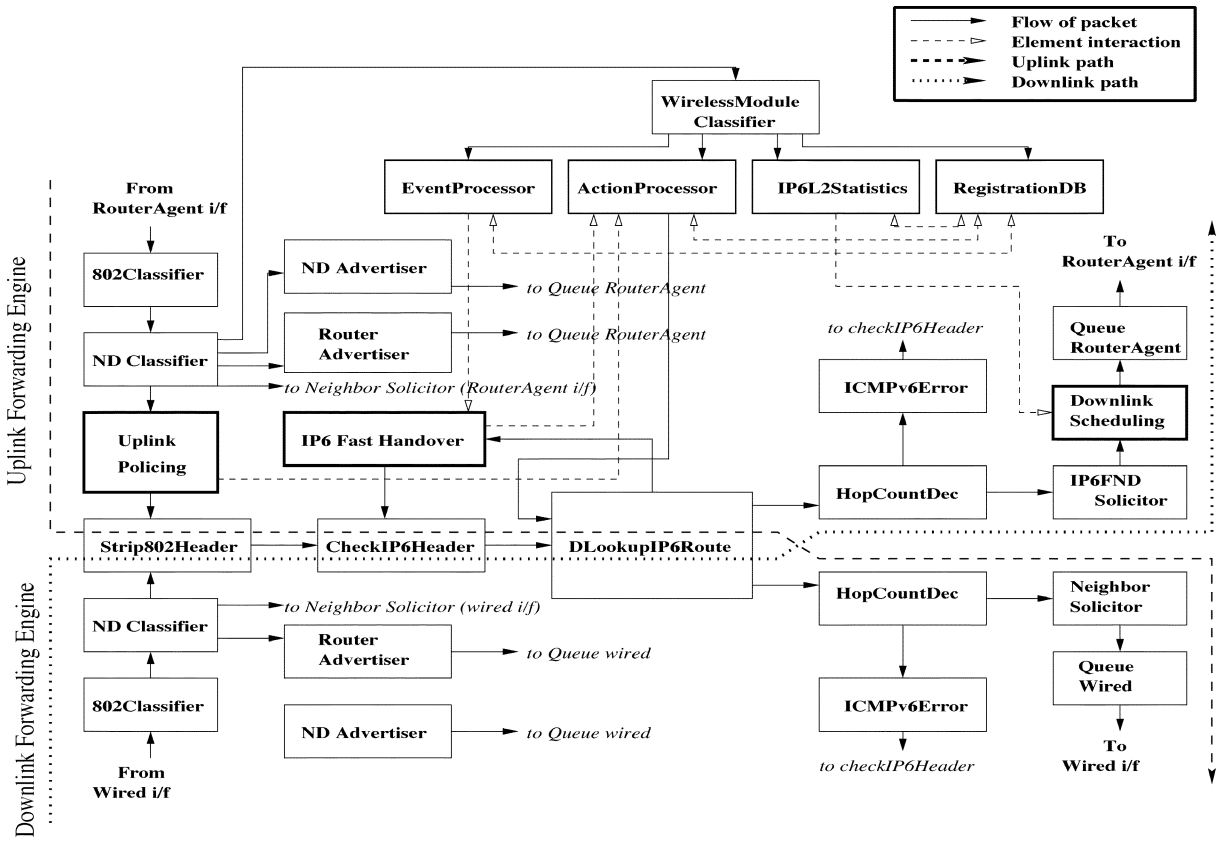

Fig. 6. *Click* element configuration diagram for the RC.

```
execute(out status-code, in node-id,
        in action-type, in params)
```
(a) Request for Action (*ActionProcessor*)

```
cancel(in request-id)
register(in event-type, in element-name,
         in callback-function, out request-id)
```
(b) Registration for Events (*EventProcessor*)

```
element.callback(in request-id,
                 in event-type, in params)
```
(c) Callback Function (*EventProcessor*)

```
get_stat(in node-id, in stat-type, out value)
```
(d) Retrieval of Statistics for link-quality (*IP6L2Statistics*)

```
get_assoc(inout node-id, inout router-agent-id )
get_node(in node-id, out ip6-addr,
         out ll-addr-type, out ll-addr )
```
(e) Queries to the *RegistrationDB*

Fig. 7. API provided by DIRAC control engine.

with the *EventProcessor* via the `register` API of Fig. 7(b). The *register* call lets the element specify the type of events it is interested in, its name, and a callback function. *EventProcessor* subsequently returns a unique ID for that request. The callback function is then stored in a linked list, one for each type of event.

Once *EventProcessor* receives an event notification message from an RA, it invokes the callback mechanism, which provides exactly once, in-order semantics for each element to be notified of a concrete event. Specifically, it traverses the linked list and invokes the *callback* call of Fig. 7(c). The call specifies who requests callback, the type of events for callback, and event parameters.

Monitoring link quality: An element learns of the channel statistics each host perceives by invoking the `get_stat` call of Fig. 7(d). The call asks for the host ID, the type of statistics it is interested in, and returns the current channel statistics. The current implementation returns only a single normalized value for channel quality computed by the RA. We plan to enrich the statistics with more elaborate information in the near future.

In the current implementation, elements learn the statistics via queries. A more proactive mechanism could follow the approach of [15]: Interested elements specify the lower and upper bounds for the value of a statistic; *IP6L2Statistics* element then triggers callback handlers when the statistic falls into these predefined bounds.

B. Router Agent (RA)

The RA is implemented on the OpenAP platform [8]. The hardware consists of a WL11000 SA-N board, which is equipped with a wired Ethernet controller (NE2000), an AMD ELAN SC400 embedded processor running at 33 MHz, 1 MB of Flash RAM, and a RS-232 serial interface for the output of the console. The wireless PCMCIA 802.11b network card is made by U.S. Robotics, using Intersil Prism2 chipset. The hardware platform is programmable through a relatively simple process; it runs an embedded version of Linux 2.4.17 in our implementation.

The Intersil chipset offers a "Host AP" operation mode. In this mode, the card and the host driver act as an AP for mobile clients and provide all management and control functionality of the 802.11 standard. The firmware handles time-critical

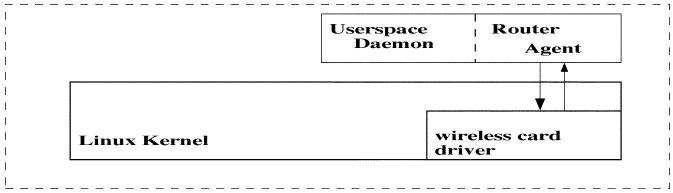

Fig. 8. Implementation of the RA.

tasks such as beaconing and frame acknowledgment, while the management functionality is provided by the host driver. Part of the driver, which provides the interface to the card, runs as a kernel module, and the remaining piece that handles the 802.11 management sublayer runs as a user-space daemon [16]. The user-space daemon opens a raw socket to the device that appears as a virtual interface to Linux. Through this socket, it is able to transmit and receive management frames.

The RA (see Fig. 8) is implemented as part of the host computer driver [16], and extends both its kernel- and user-level components. To collect the link-layer statistics, we implemented an *ioctl()* call in the kernel driver that retrieves the values from the hardware configuration records of the card, although a more efficient implementation would perform the same operation using task_queues to eliminate context switches. The user-level part of the RA periodically calls this ioctl(), and sends the results to the RC via the wired network. Furthermore, it registers with the event-loop of the driver, to intercept 802.11 management frames, and also apply actions requested by the RC in a hardware-specific manner (e.g., modify configuration settings such as MAC access lists).

VI. Prototype Wireless Services

To demonstrate the practicability of DIRAC in implementing wireless and mobility aware services, we design and implement three protocols within the DIRAC router framework. All three protocols are wireless specific and they are: 1) a link-layer informed fast handover protocol; 2) a channel-adaptive FEC-based packet forwarding solution; and 3) link-assisted policing. The above three services sample the rich wireless-adaptive protocols in the literature. They *span* both the control and data planes, and both uplink and downlink. Not only do they provide considerable performance gains, they also showcase the viability of these new wireless services within our router framework, which are otherwise not feasible in current router systems.

A. Link-Layer Informed Fast Handover

The fast handover service showcases how the control-plane element benefits from link-layer feedback. When a mobile user roams between subnets in a campus environment and uses mobile IPv6 only [14], the handover latency can be significant and the transient packet loss nonnegligible [17]. A fast handover protocol helps to reduce this latency and minimize loss through setting up a tunnel between two ARs during handoff, so that packets in transit can be forwarded by the old AR to the MN via the new AR. Recent work [5], [18] proposes to use link-layer triggers to either predict or respond to a handover event, and our

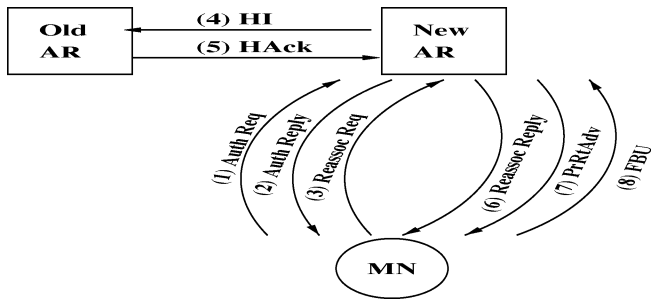

Fig. 9. Link-layer informed fast handover protocol.

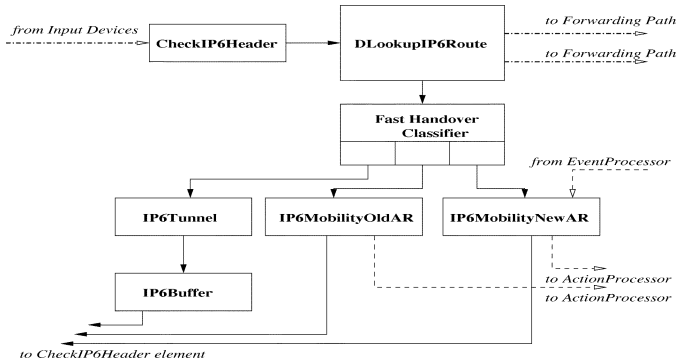

Fig. 10. Compound *IP6 Fast Handover* element.

protocol adapts such earlier proposals [5] to the specific 802.11 link-layer technology.

The main idea of the proposed solution is as follows: when a MN roams from one cell in a subnet to another cell in a different subnet, link-layer handover will happen first. This signals the earliest time the IP-layer fast handover may happen. We may simply use the link-layer handoff signal to trigger the IP-layer handover, and embed the IP-layer handover in the link-layer process. In the 802.11b network, we make use of the *ReAssociation* message of the link-layer handover process that precedes the network layer, and use it as the event that triggers the network-layer handoff service. Fig. 9 shows the message exchange of the protocol. It involves 8 messages, similar to the recent IETF draft [5].

The protocol is implemented as a compound element in Click, i.e., *IP6FastHandover* in Fig. 6. It consists of four elements in Fig. 10. *IP6MobilityNewAR* implements the operations of the new access router for the MN. It registers with *EventProcessor* in order to be notified upon the occurrence of a reassociation event. It also takes action employing *ActionProcessor* to allow link-layer association by the RA. *IP6MobilityOldAR* implements the operations of the old access router for the MN, while *IPTunnel* and *IP6Buffer* elements establish tunnels and temporarily buffer packets, respectively.

This protocol also requires implementation efforts on the MN part. We developed the client-side functionality and integrated it with an open-source distribution of mobile IPv6 for Linux.[2] The protocol runs as part of the MIPv6 kernel module, and is enabled at runtime via a *sysctl()* call. This function also temporarily disables the native movement detection algorithm of MIPv6.

[2]Available at: http://www.mipl.mediapoli.com.

B. Channel-Adaptive FEC-Based Downlink Forwarding

Adaptive FEC-based forwarding seeks to address the wireless HoL blocking problem. When the destination host of the HoL packet experiences channel error burst, the packet is retransmitted continually until the maximum retransmission count is reached. This blocks transmissions of subsequent in-queue packets destined to other hosts perceiving error-free channels. Its negative effect includes considerable transmission delay and jitter, and reduced throughput.

The lightweight FEC-based protocol that solves the HoL problem works as follows: the RC requests an action from the RA to disable retransmissions, since they incur HoL blocking over the error-prone channel. However, in order to compensate for loss of reliability due to lack of retransmissions-based error recovery, FEC is used to recover losses. It is implemented at the RC, which adapts its encoded redundancy based on link-layer channel statistics provided by the RA. RAs do not have retransmission queues, and report link-layer statistics measured in frames lost due to errors. Lost frames are identified by the respective lost ACKs of the 802.11 protocol. The FEC-based forwarder implements Deficit Round Robin scheduling on a per-host basis, to avoid transmitting consecutive packets from an error-prone flow.

The chosen FEC algorithm [19] operates at the packet level. It takes two parameters (n, k), where k denotes the number of application packets, and n denotes the total number of packets (including the redundant $n-k$ packets) to be transmitted during an interval $[t, t+T]$. We adjust k and n based on the channel statistics, i.e., the frame loss[3] measured for each host and calculated as $frames\_loss = (frames\_errors/frames\_txmit)\cdot$, where $frames\_errors$ is the number of lost frames due to errors, and $frames\_txmit$ is the total number of transmitted frames. Then, k and n are adjusted periodically to satisfy the relation of $k/n = 1 - frames\_loss$. The receiving host is able to *reconstruct* the original k packets whenever it receives *any* k copies out of these n packets.

The above design is implemented in two Click elements shown in Fig. 11. *FECController* and *FECEncoder*. *FECController* periodically queries the *IP6L2Statistics* element for channel statistics, and adjusts k and n of the *FECEncoder* element. *FECEncoder* encodes packets sent by *FlowClassifier*, and produces $(n-k)$ redundant packets.

C. Link-Layer Assisted Uplink Policing

This protocol seeks to police aggressive uplink flows via link-layer mechanisms. In a link-layer unaware scheme, the router *posterior* drops all the packets once they reach the router, but they have already consumed the wireless bandwidth. In the link-layer assisted design, the router uses the link-layer access control mechanism at the AP, i.e., 802.11 deauthentication frames, to police an aggressive flow. This effectively denies such a flow *temporary* network access over the wireless channel.

Such a mechanism can be used by the network for the prevention of denial-of-service attacks originating from wireless nodes. And while this threat may not be severe for current 802.11b WLANs, due to their limited bandwidth, it is expected to become more serious in the future, once the 802.11a/g wireless access technologies at 54 Mb/s be more widely deployed.

[3]We also experimented with other link-layer metrics, e.g., channel quality expressed in dBm reported by the wireless card. However, our results show that there seems to be no quantifiable correlation between this value and the actual frame loss.

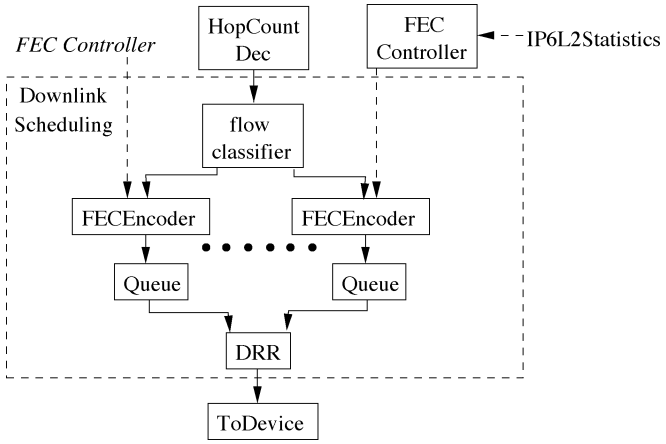

Fig. 11. Compound element for the adaptive FEC.

The protocol has two building blocks: arbitrator and enforcer. The arbitrator decides which uplink flows are aggressive by checking a flow's actual transmission rate R_{actual} against its expected fair share R_{share}. The enforcer takes two progressively severe shaping actions with increasing dropping intensity over time: RED-like dropping, and temporary denial of access. For aggressive, responsive flows (e.g., transmission control protocol (TCP) friendly flows), RED-like dropping should be sufficient to shape the flow over the uplink wireless channel. For persistent nonresponsive flows, temporary denial of access controls the damage. However, such flows are still granted long-term fair share over $[t, t+T]$. This is achieved by setting the access denial period t_{denial} as $t_{\text{denial}} = (1 - R_{\text{share}}/R_{\text{actual}}) \cdot T$. The t_{denial} value is notified to the RA which informs the link layer to enforce it. These building blocks are implemented as a Click compound element, which requests from the RAs the use of a link-layer mechanism (the 802.11 deauthentication mechanism in particular) in the device driver to temporarily deny a flow's access.

VII. System Evaluation

The evaluation of our "proof-of-concept" prototype seeks to answer two questions.

- How much overhead do the primitives of link-layer information introduce to the system?
- How beneficial is such an "informed" operation of the router for the wireless part of the network?

For the first question, we measure the overhead for processing *events*, *actions*, and *statistics*. For the second question, we evaluate the three wireless services of Section VI. The network components used in the testbed experiments are listed in Table III. The specific configuration for each experiment is to be described in each section.

A. System Overhead

We first gauge how much processing overhead DIRAC incurs. We measure the consumed CPU time using the Pentium

TABLE III
TESTBED COMPONENTS USED IN THE EXPERIMENTS

node (role)	*Hardware*
S (Source)	Intel Pentium III 450MHz, 128MB
(CN/SA)	Intel Ethernet Express 10/100
MN1, MN2, MN3	Intel Pentium III 1.1GHz, 256MB
(Mobile Nodes)	Tulip & Netgear Fast Ethernet Cards
R1, R2	Intel Pentium III 900MHz
(Routers)	256MB, Intel Ethernet Express 10/100

TABLE IV
AVERAGE CPU TIME COST FOR THE PRIMITIVES AND BASIC FORWARDING (IN NANOSECONDS)

Operation	*Time*(ns)
Action	498
Event	1712
Statistics report	32
Basic forwarding	1299

cycle counter, when running the code relevant to the processing of *events, actions*, and *statistics*. The results are then compared with the forwarding cost associated with delivering a packet from one interface of the RC to another. This provides a relative measure for the overhead of the control engine in DIRAC, and demonstrates the scalability of the approach in an enterprise environment, where there are a large number of APs connected to the same access router, all of which transmit link-layer information periodically.

The microbenchmarks for this experiment are collected on machine R1 of Table III, which serves as the RC. We perform repetitive invocations of the functions that process the primitives of Section IV, and compute the average cost of each. The cost of the basic packet forwarding is defined as the amount of CPU time spent in the elements of Fig. 6 of the forwarding path divided by the number of packets delivered. It is obtained by using host S of Table III as the source, host MN1 as the destination, and R1 as the router. The input load is 100 000 packets, generated at a relatively low rate in order not to overwhelm the hardware that runs the RC with interrupt handling.

Table IV gives the experimental results. The processing overhead for each of the three primitives of actions, events and statistics is comparable to the forwarding cost. Actions and statistics incur minor processing overhead, but events incur more. Specifically, for *statistics* processing, only 32 ns are required by the *IP6L2Statistics* element to extract and store the values for a report from a RA. Our experiment also shows that additional 95 ns are required to classify the packet as a report message that should be pushed to the *IP6L2Statistics* element for further processing.

The large difference between processing a statistics report and an event, is because the former has been highly optimized: only three memory-copy operations using pointer arithmetic are required to extract the values stored in the 10-byte report of channel quality statistics per host. However, the *EventProcessor* element still uses the highly convenient, but expensive string operations that the Click framework provides. It can definitely be optimized similarly, a task that we leave as future work.

At this point, we should note that *events*, as well as *actions*, are not expected to occur very frequently in practice, probably

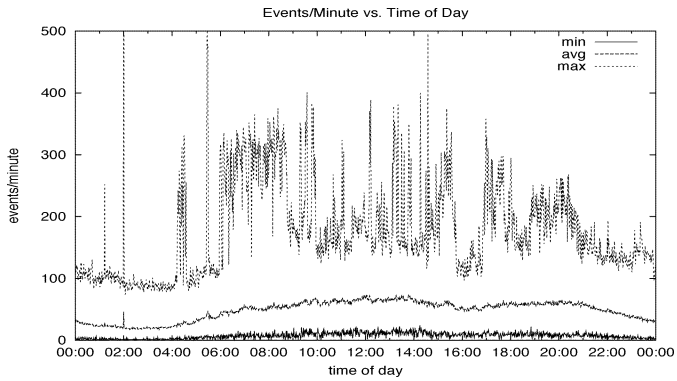

Fig. 12. Events/minute (min/avg/max) versus time of day.

in the order of one every few hundreds of milliseconds at most. To validate this hypothesis, we analyze *syslog* traces from the Dartmouth College campus, collected over a period of four months (Spring 2002), from the Dartmouth College campus. These traces log the layer-2 activity of the campus-wide, 802.11 wireless network, namely, the *association*, *disassociation*, *roaming*, *authentication*, and *deauthentication* events. Fig. 12 plots the rate of events/minute (min/avg/max values) as a function of "time of the day," for all days of the trace. Our assumption of one event every few hundred milliseconds indeed holds, as we can see from the figure: around 60 events/minute are reported at any time during the day, which is not that frequent. From this analysis, we conclude that the system overhead associated with the report of *events*, and *actions* is not a primary concern.

To explore the scalability of the RC with the number of registered APs, we set up machine S as a source that transmits UDP packets at several rates (to emulate various numbers of APs, each of which transmits one report every 50 ms). Each packet carries dummy statistics for 10 clients. Our experiment shows a linear relationship between the cost for processing statistics reports and the number of APs registered with the RC, estimated in the order of a few hundreds of microseconds. For example, the cost for processing 1000 packets/s from 50 APs is around 140 μs. However, this should be taken just as an estimate, since packets are generated from a single source and all reports carry statistics for the same "clients," a fact that takes advantage of the L1 data cache of the processor.

A similar linear relationship is revealed when we study how the processing cost at the RC scales with the number of MNs associated with each AP. In another experiment, packets are transmitted at a high transmission rate of 2000 packets/s, emulating the operation of 100 APs connected to the RC, each of which sends one report every 50 ms. For up to 100 mobile nodes per AP—already a large number due to bandwidth limitations—the processing overhead is less than 1 μs. It is exactly 629 ns for processing channel statistics for 50 mobile nodes that are all carried in the same report.

B. Performance of Wireless Services

The ultimate goal of DIRAC as a wireless router system is to improve the router performance in wireless networks. This will be achieved through implementing wireless services described

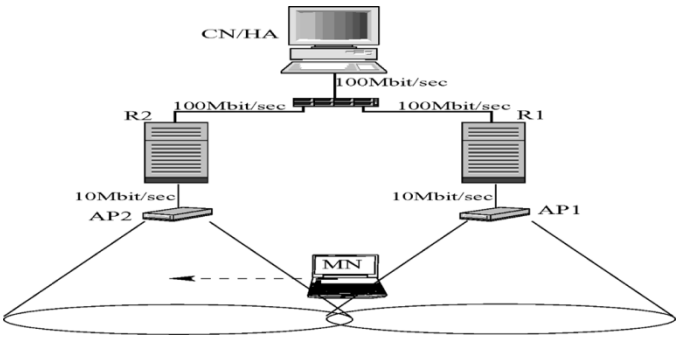

Fig. 13. Experiment environment for fast-handover evaluation.

in Section VI, which address host mobility, improve packet forwarding over wireless links, and provide better sharing of the channel. Our measurements show that such services indeed provide considerable performance gains.

1) Host Mobility: We use the testbed configuration of Fig. 13 to evaluate the performance of the link-layer informed fast handover proposal of Section VI-A. A node roams from the vicinity of the cell served by access router R1, to the one served by R2. Node S acts both as a home agent (HA) for mobile host MN1 and as a correspondent node (CN) that transmits traffic to MN1.

A total number of 10 handoffs are performed, and different types of traffic (CBR, video over UDP, audio over TCP) are transmitted from one run to another. We use *tcpdump* to measure the time when the signals of the protocol are sent and received on the new access router (R2). Table V breaks down the delays of signaling messages involved in the fast handover protocol (see also Fig. 9). The table shows that only 7.9 ms—in addition to the inevitable link-layer switching delay from one AP to another, typically, several hundred milliseconds [20]—is required to establish the tunnel, which is created after the ReAssociation Reply message is sent by the new AR. Once the tunnel is set up, data packets for the MN arrive at the new AR via the tunnel with the old AR, thus minimizing transient packet loss.

From Table V, it takes the MN 18.6 ms to receive its new care-of-address and gateway (PrRtAdv message) after it starts receiving packets from the new point of attachment. This delay is large compared with the other table values. This is due to the fact that the PrRtAdv message is sent after the receipt of a notification message from the RA that completes transmission of the ReAssocReply message. However, since the RA runs as a user-level program at the AP, its execution is deferred due to the execution of the driver code (also the code for the bridge), which runs in the kernel, has higher priority, and keeps the processor busy by forwarding data traffic.

Fig. 14 plots the average end-to-end delay of UDP packets transmitted at a rate of 50 packets/s with a packet size of 512 bytes, during the layer-3 handoff experiment. The particular version of our wireless cards (Orinoco Gold for the MN and Prism2 for the AP) gives a link-layer (L2) switching latency of around 500 ms, which matches the value reported by another independent experiment in the literature [20]. The tunnel setup time of the L3 handover protocol is 8.2 ms, after which the packets are transmitted over the tunnel for about 5 s (we artificially delayed the transmission of a binding update

TABLE V
LATENCY INVOLVED IN THE LINK-LAYER INFORMED FAST HANDOVER PROTOCOL (IN MILLISECONDS)

	mean	median	stdv
HAck - HI	4.2	3.6	1.5
ReAssocReply - ReAssocReq	7.9	7.4	1.8
PrRtAdv - ReAssocReply	18.6	18.8	3
FBU - PrRtAdv	8.1	6	6

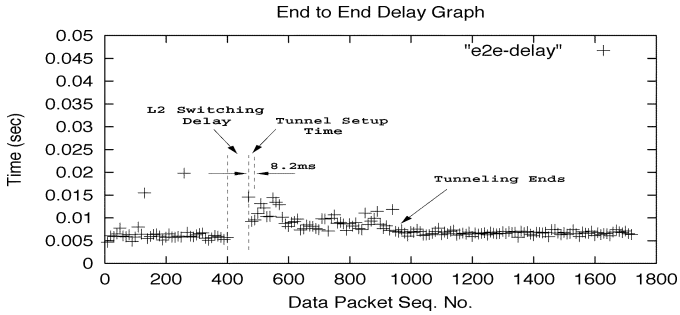

Fig. 14. Average end-to-end delay of UDP packets during the fast handover experiment.

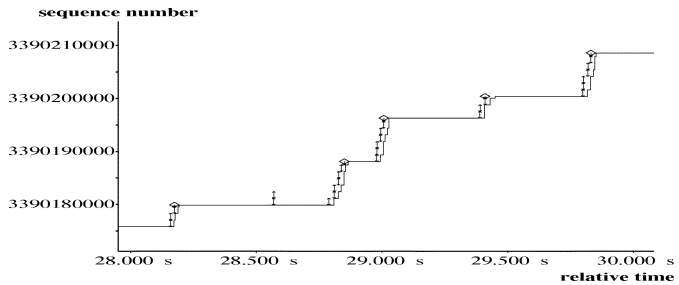

Fig. 15. TCP sequence graph during fast handover.

of the MIPv6 protocol [14], to observe the tunneling effect). Shown in the figure, for the duration of the tunnel, packets experience an increase of 5 ms in their end-to-end delay, making the introduced overhead acceptable even for demanding real-time applications. As a last note, the latencies measured are very close to the ones reported by Koodli and Perkins in [9].

Using the same testbed configuration (Fig. 13), we also run an experiment with audio transmission over TCP. More specifically, the CN runs the Icecast MP3 streaming server that delivers audio streams over TCP transport at 128 kb/s, while the MN performs a handover from AR1 to AR2. The particular audio traffic exhibits the following characteristic, which can also be observed from Fig. 15 that plots the time-sequence graph with the sequence numbers of the TCP segments on the y axis, and the relative time from the start of the TCP trace on the x axis: every approximately 400 ms, TCP packets carrying audio data are transmitted back-to-back in groups, with the last packet of each group having the PSH flag set. In the specific snapshot of Fig. 15, and after inspection of the *tcpdump* traces that were collected, the fast handover incident starts at 28.2 s. The link-layer switching time takes almost 500 ms as in the previous case, and before it completes, two segments are transmitted by the streaming server at relative time 28.568 s. As a result, they are lost and get retransmitted after 200 ms. The latency of the L3 handover protocol is only 7.351 ms, and the whole handover procedure manages to finish before the retransmission of the two

TABLE VI
HOL BLOCKING PROBLEM AND THE FEC-BASED SOLUTION

average quality	stdv	MN1	MN2	MN3
10.38	1.53	9903	9830	2647
12.38	1.46	9980	9940	7450
15.02	2.01	9980	9978	9581
20.17	1.54	9943	9903	9985
25.32	1.23	9925	9951	9984
35.53	1.97	9942	9936	9949
44.77	2.27	9928	9927	9954

TABLE VII
HOL BLOCKING PROBLEM AND THE DEFAULT METHOD OF LINK-LAYER RETRANSMISSIONS

average quality	stdv	MN1	MN2	MN3
9.47	2.17	4808	4807	3038
12.21	1.81	7559	7554	6558
14.81	1.76	9999	9999	9822
20.44	1.98	9999	9999	9988
25.38	1.15	9999	9999	9990
35.11	2.29	9999	9999	9996
45.35	2.42	9998	9999	9998

lost packets, which are eventually received at 28.792 s, while the MN is still using its old care-of-address. As in the previous experiment, the dominant factor for the overall latency of the fast handover protocol is the time needed for link-layer switching, almost two orders of magnitude larger than that of L3 handoff.

2) FEC-Based Downlink Forwarding: We now demonstrate the HoL blocking problem of Section VI-B, and evaluate the FEC-based solution.

The testbed consists of a source machine (S), an access router (R1), and three mobile nodes (MN1, MN2, and MN3). The source transmits 10 000 packets, 1024 bytes each, to each MN at a rate of 65 packets/s/node. Two of the MNs (MN1 and MN2) experience excellent channel quality. For the third one, we change its location by selecting seven spots so that its perceived channel quality varies from really bad to very good, one spot at a time, as shown in Tables VI and VII. We compare the performance of the FEC-based solution against the default method for link-layer retransmissions, the default value of which is 8 in our cards. For each location and method, we run three rounds of experiments interleaving the two methods, in order to minimize fluctuations in channel quality due to interferences. Moreover, to provide fair comparisons, we lock the receiving rates for all MNs to 11 Mb/s, by disabling the multirate support of the driver in the AP card. The link-layer statistics sampling interval at an AP is 100 ms.

Tables VI and VII show the number of packets received by each node, out of the 10 000 packets sent to each one of them, for various channel qualities experienced by MN3. From the tables, we make the following observations:

First, the HoL blocking problem and its solution by the FEC-based approach are demonstrated from the results of the first two rows of both tables. With the link-layer retransmission method, the throughput of MN1 and MN2 has coupling effect with MN3. When MN3 experiences bad channel and its throughput suffers, so do MN1 and MN2. However, in the FEC-method, this negative effect does not show up any more. Even though MN3 still suffers from low throughput, MN1 and MN2 enjoy very high

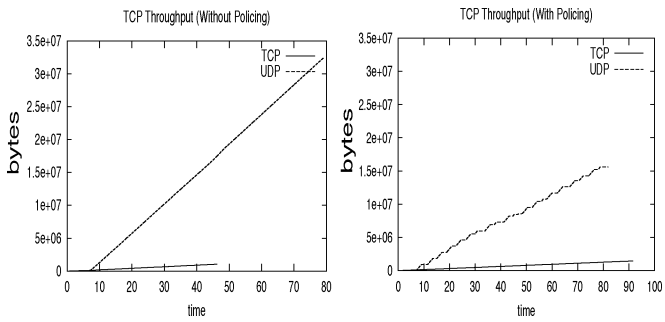

Fig. 16. Throughput without and with policing.

throughput. This exactly shows the impact of HoL blocking. In the retransmission method, excessive retransmissions for MN3 cause buffer overflows at the AP. This buffer drop also happens for MN1 and MN2, since all three share the buffer at the AP. However, the FEC-based approach, which also produces lots of redundant packets for MN3, causes buffer overflow only for the queue of MN3 at the RC. This early drop at RC, rather than AP, leads to isolation for the other two flows; the FEC-based approach has an inherent rate-control mechanism.

The second observation is mostly related to the wireless channel itself: the more the quality metric (measured in average quality and standard deviation in the tables) approaches the value of 10, which seems to be the threshold at which the wireless card of the MN starts scanning for another AP to initiate handoffs, the sharper the drop is in the number of packets received. At higher values, the channel behaves in a more predictable way.

3) Policing: We evaluate the effectiveness of policing with a testbed consisted of two mobile nodes MN1 and MN2 competing for the wireless channel. MN1 is running Icecast, an MP3 streaming server that uses the TCP protocol to deliver audio streams at 128 k/s. MN2 is an aggressive UDP source that generates 3.6 Mb/s of traffic, enough to consume the most of the channel by itself.

Fig. 16 shows that without a policing mechanism, the aggressive MN2 occupies a significant portion of the wireless channel and leaves little operational bandwidth for MN1. TCP operated by MN1 is disrupted midway since it can no longer operate throughout the experiment. In the same figure, with policing, however, MN1 is protected from MN2's destructive behavior. Whenever the RC deems the MN2 being overly aggressive, it will instruct the AP to penalize MN2 by temporarily restricting its access to the AP. Thus the benevolent MN1 can still operate at its full capacity. The figure shows that the throughput of MN2 is constantly being limited by the RC and the MN1 is able to run to the completion of our experiment without being shut down.

In summary, DIRAC provides primitives for interacting with the wireless link-layer without introducing much overhead to the system. It scales well to the number of MNs, consuming processing power less than 2.5% of the cost for a standard packet forwarding. It takes less than 1 μs to process channel states for 50 clients per AP. It can also support a large number of APs; less than 140 μs are required to process statistics reports from 50 APs, each of which transmits 20 reports/s. Through its API, a number of network services critical to the wireless environment

can be readily deployed. The fast handover service establishes a tunnel for packet forwarding in less than 10 ms. FEC-based forwarding solves the wireless HoL blocking problem with overhead comparable with the default retransmission based solution. The policing service protects flows from aggressive users.

VIII. Lessons Learned

The first learned lesson is that, as in other systems areas, it is important to "make the common case fast." This is certainly true in the router design context, particularly on the data path. In our system, processing channel statistics is an important common case. Therefore, it is critical to minimize the processing of link-layer statistics, in order to allow the router to scale to a large number of hosts and APs.

The second issue involves the OS scheduling of computations on a device that performs forwarding. Naturally, a higher priority is given to the basic forwarding functionality, while other control-plane operations receive a smaller fraction of the processing capacity. This also applies to the RA on APs: Since the bridge performs a limited-scope L2 forwarding of Ethernet frames (from the wired interface to the wireless and backward), the RA and user-level implementation of the 802.11 daemon have to be scheduled at regular intervals, in order to execute the management functionality. This can be done either by having the bridge giving up the CPU voluntarily, in order to allow execution of the RA, or by implementing the latter inside the kernel.

The last issue is related to the collection of statistics that characterize channel quality on a per-host basis. Initially, we implemented a mechanism so that MNs send periodic reports back to the AP, regarding the perceived channel quality. These reports used SNR measured in dBm. However, the estimation of frame loss from this metric was not very accurate. For this reason, we went ahead and modified the driver of the AP card to obtain the frame loss statistics of Section V. A more elaborate set of statistics can be collected via the *CommTallies* [16] facility of the wireless card, but unfortunately they are not available on a per-host basis.

IX. Related Work

Motivated by enhancing routers with new capabilities such as assorted filters, firewalls, and traffic prioritization, router design for the wired Internet has been an active research area recently. Most of such routers are software based and seek to support easy customization of router functionality. Examples of such systems include Scout and its extension with network processors [1], Click [7], Router Plugins [21], and XORP [22]. Although all of these proposals provide extensible and general frameworks for router development, they lack specific support for working in wireless networks. A distributed router design appears in [23]. However, the focus there is to achieve high throughput through a cluster of PCs. In [24], which can also be viewed as a distributed design over the hierarchy of network processors, the focus is on supporting active networks.

There are home wireless router systems available in the market, including the 3Com home wireless router, Intel AnyPoint Home Network, and Lucent Orinoco RG-1000 Residential Gateway, to name a few. The main focus of such systems is to bridge the wireless channel with the wired Internet. The claimed new features are mainly on security such as integrated firewall. These systems do not take a distributed architecture. The MIT personal router [25] is concerned with the automatic selection of providers of wireless access, based on dynamic modeling of the user and knowledge of a market of wireless services. DIRAC serves as network edge router and improves the performance of forwarding engine over the wireless channel and support fast handover.

Other commercial solutions for the enterprise environment attempt to address similar issues but in a narrow scope. The Vernier Networks 6500 [26] system focuses mostly on security and management. The goal is to secure a wireless network by enabling security protocols and enforcing access rights across multiple network domains. The Lucent SpringTide 7000 Wireless IP Service [27] provides several mobile IP services mainly through tunneling. The Nomadix Service Engine Software Modules [28] use techniques based on ARP to identify new users and address user mobility. No data-plane forwarding issues are addressed in these systems. In Aruba Network's Aruba 5000 and Aruba 50 systems [29], the proprietary APs transmit link-layer information feedback to the WLAN switch, in order to enhance security and roaming functions.

Many components in the router system have been studied in the literature. Wireless scheduling has been an active research area in recent years [3], [4]. Several fast handover solutions [5], [17] have been proposed, and system solutions for mobility support have been devised and implemented [30]. Our distributed router architecture borrows freely from the literature, but is not tied to a particular protocol. It rather provides all those primitives that are necessary for the implementation and deployment of these protocols.

Cross-layer interactions have been a popular design guideline in wireless networking. The Mobiware toolkit [31] spans several layers of the protocol stack, from programmable MAC, to transport and adaptive applications. We take a more controlled approach to cross-layer interactions in DIRAC, to reduce overhead, achieve better scalability and extensibility.

X. Conclusion

Router is expected to be a critical architectural component in the emerging packet-switched cellular networks, such as campus, enterprise, and metropolitan wireless networks based on 802.11 technology. Conventional routers do not handle mobility and wireless link issues. The solution principle, well documented in the literature, is to take adaptive actions based on channel conditions and mobility patterns.

This paper describes DIRAC, a wireless router system. The goal is to design and implement a systems framework that enables wireless protocols and provides router services for wireless data networks. DIRACs unique approach is to let the RC interact, in a controlled manner, with the wireless link layer, which is in the best position to know the network dynamics. The interaction is twofold: the RC takes feedback input from the link layer, and dictates actions to the link layer. This leads to renovated designs in both data and control planes of a router and facilitates implementation and deployment of new wireless services.

To this end, we implement a distributed router architecture, in which the RC interacts with each RA at each AP. We also implement three prototype wireless services, spanning both control and data planes, and covering both the uplink and downlink forwarding path on the data plane. Together, they showcase how a wide spectrum of wireless protocols can be implemented and evaluated within DIRAC. DIRAC is built using off-the-shelf hardware, including a PC and several APs.

Ongoing work seeks to support firewalls for roaming users, implement a inter-AP, coordinated resource management protocol to support VoIP, to add energy efficiency feature via power-saving mode for mobile hosts, to explore performance optimization techniques in DIRAC, and to gain more experiences through a larger testbed deployment.

REFERENCES

[1] T. Spalink, S. Karlin, L. L. Peterson, and Y. Gottlieb, "Building a robust software-based router using network processors," in *Proc. Symp. Operating Syst. Principles*, 2001, pp. 216–229.

[2] Y. Gottlieb and L. Peterson, "A comparative study of extensible routers," in *Proc. IEEE Open Arch. Netw. Program.*, New York, Jun. 2002, pp. 51–62.

[3] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE ACM Trans. Netw.*, vol. 7, no. 4, pp. 473–489, 1999.

[4] X. Liu, E. Chong, and N. B. Shroff, "Transmission scheduling for efficient wireless network utilization," in *Proc. INFOCOM*, Apr. 2001, pp. 776–785.

[5] G. Dommetry, A. Yegin, C. Perkins, G. Tsirtsis, K. El-Malki, and M. Khalil, "Fast handovers for mobile IPV6," Internet Draft, draft-ietf-mobileip-fast-mipv6-04.txt, 2002.

[6] J. Ahn and J. Heidemann, "An adaptive FEC algorithm for mobile wireless networks," USC/ISI, Tech. Rep. ISI-TR-555, Mar. 2002.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

[8] (2001) Open AP Platform. [Online]. Available: http://opensource.instant802.com/

[9] R. Koodli and C. Perkins, "Fast handovers and context transfers in mobile networks," *ACM Comput. Commun. Rev.*, vol. 31, no. 5, pp. 37–47, Oct. 2001.

[10] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media access for multirate ad hoc networks," in *Pro. 8th Annu. Int. Conf. Mobile Comput. Netw.*, 2002, pp. 24–35.

[11] L. Yang, P. Zerfos, and E. Sadot, "Architecture taxonomy for control and provisioning of wireless access points (CAPWAP)," Internet Draft, draft-ietf-capwap-arch-06.txt, Nov. 2004.

[12] L. Yang, R. Dantu, T. Anderson, and R. Gopal, "Forwarding and control element separation (forces) framework," Internet Draft, draft-ietf-forces-framework-08.txt (Work in Progress), 2003.

[13] Juniper Research, Juniper Research, [Online]. Available: http://www.juniperresearch. com.

[14] D. Jonhson, C. Perkins, and J. Arkko, Mobility support in IPv6, Internet Draft, 2002.

[15] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile application-aware adaptation for mobility," in *Proc. 16th ACM Symp. Oper. Syst. Principles*, Saint Malo, France, 1997, pp. 276–287.

[16] J. Malinen. Intersil Prism2 Driver. [Online]. Available: http://hostap.epitest.fi

[17] H. Yokota, A. Idoue, T. Hasegawa, and T. Kato, "Link layer assisted mobile IP fast handoff method over wireless LAN networks," in *ACM MobiCom*, 2002, pp. 131–139.

[18] P. McCann, Mobile IPv6 fast handovers for 802.11 networks, Internet Draft, 2002.

[19] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Comput. Commun. Rev.*, pp. 24–36, 1997.

[20] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the IEEE 802.11 MAC layer handoff process," UMD/UMIACS, Tech. Rep. UMIACS-TR-2002-75, 2002.

[21] D. Decasper, Z. Dittia, G. Parulkar, and B. Platter, "Router plugins: A software architecture for next generation routers," *IEEE ACM Trans. Netw.*, vol. 8, no. 1, pp. 2–15, Feb. 2000.

[22] International Computer Science Institute, Center for Internet Research.. Xorp: Extensible open router platform. [Online]. Available: http://www.xorp.org

[23] P. Pradhan and T. Chiueh, "A cluster-based, scalable edge router architecture," State Univ. New York, Stony Brook, Tech. Rep., [Online]. Available: http://www.ecsl.cs.sunysb.edu/prashant/papers/design.ps.gz.

[24] N. Shalaby, L. Peterson, A. Bavier, Y. Gottlieb, S. Karlin, A. Nakao, X. Qie, T. Spalink, and M. Wawrzoniak, "Extensible routers for active networks," in *Proc. DARPA Active Netw. Conf. Exposition (DANCE)*, San Francisco, CA, May 2002, pp. 92–116.

[25] P. Faratin, J. Wroclawski, G. Lee, and S. Parsons, "The personal router: An agent for wireless access," in *Proc. AAAI Fall Symp. Pers. Agents*, N. Falmouth, MA, Jul. 2002, pp. 13–21.

[26] Vernier Networks. Vernier networks system 6500. [Online]. Available: http://www.verniernetworks.com/AMCS6500.html

[27] Lucent Technologies. Springtide 7000 wireless IP service switch router. [Online]. Available: http://www.lucent.com/livelink/090094038 0004ac9_Brochure_datasheet.pdf

[28] Nomadix Networks. Nomadix service engine. [Online]. Available: http://www.nomadix.com/downloads/products/NSE_Data_Sheet.pdf

[29] Aruba Networks. (2003) Aruba 500. [Online]. Available: http://www.arubanetworks.com/products/5000

[30] A. Miu and P. Bahl, "Dynamic host configuration for managing mobility between public and private networks," in *Proc. 3rd Annu. USENIX Symp. Internet Technol. Syst. (USITS)*, Mar. 2001, pp. 147–158.

[31] O. Angin, A. Campbell, M. Kounavis, and R. Liao, "The mobiware toolkit: Programmable support for adaptive mobile networking," *IEEE Pers. Commun. Mag.*, pp. 32–43, Aug. 1998.

Petros Zerfos (S'99) received the B.E. degree from the National Technical University of Athens, Athens, Greece, and the M.S. degree in computer science from the University of California, Los Angeles (UCLA). He is currently working towards the Ph.D. degree in the Computer Science Department, University of California.

His research interests include wireless and mobile networking and computing, security, and large-scale distributed systems.

Gary Zhong received the B.S. degree from the University of California, Davis, and the M.S. degree from the University of California, Los Angeles (UCLA).

His research interests include wireless networks, mobile computing and services, sensor networks, mobile ad hoc networks, and network security.

Songwu Lu (M'96) received the M.S. and Ph.D. degrees from the University of Illinois at Urbana–Champaign, Urbana.

He is currently an Assistant Professor of Computer Science at the University of California, Los Angeles (UCLA). His research interests include wireless networking, mobile computing, wireless security, and computer networks.

Dr. Lu is a member of the Association for Computing Machinery (ACM) since 1999. He received the National Science Foundation (NSF) CAREER Award in 2001.