Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Networks 50 (2006) 1675–1691

Computer Networks

www.elsevier.com/locate/comnet

Improving aggregated channel performance through decentralized channel monitoring ☆

Puneet Sharma [a], Jack Brassil [b], Sung-Ju Lee [a,*], Kang G. Shin [c]

[a] *Mobile and Media Systems Lab, Hewlett–Packard Laboratories, 1501 Page Mill Road, ms1181, Palo Alto, CA 94304, United States*
[b] *Mobile and Media Systems Lab, Hewlett–Packard Laboratories, Princeton, NJ 08540, United States*
[c] *Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, United States*

Received 29 April 2004; received in revised form 23 February 2005; accepted 17 June 2005
Available online 24 August 2005

Responsible Editor: E. Chong

Abstract

Aggregating low-speed WAN links into a higher-speed logical link promises to improve data-transfer rates to collaborating communities of wireless mobile multihomed devices. Such bandwidth aggregation systems must adapt to link dynamics as the number of links and the channel conditions vary with time due to mobility, power dissipation, and channel interference. A monitoring architecture that accurately measures the link dynamics and promptly feeds this information to the system is vital to realize significant bandwidth aggregation performance gains. In this paper we present various architectural design alternatives for such a monitoring system, and evaluate them using both analysis and simulation. We show that a properly-designed monitoring system can accurately measure and quickly respond to changes in communication link performance while minimizing the control overhead.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Monitoring system; Bandwidth aggregation; Wireless community networks

1. Introduction

Users of wireless mobile computing devices seeking Internet connectivity in a public setting often face a choice between convenience and performance. One might locate, approach, and connect to a public wireless access point using a high-speed local area network (LAN) such as IEEE 802.11x, or accept nearly ubiquitous but much slower access using a wide area network (WAN) such as a 2.5 G or later-generation cellular network.

Although networks that provide high-speed access to mobile users are currently under

☆ A preliminary version/subset of this paper was presented at IFIP-TC6 Networking 2004, Athens, Greece, May 2004.
* Corresponding author. Tel.: +1 650 857 3894.
E-mail address: sjlee@hpl.hp.com (S.-J. Lee).

development (e.g., *EvDO*, 4 G cellular systems), they will not be widely available soon. To meet this need today, we have proposed an alternative, complementary solution to high-speed Internet access through collaborative resource sharing [1]. A group of multihomed wireless, mobile computing and communication devices in close proximity dynamically form communities interconnected through their compatible high-speed LAN interfaces; we call these ad hoc groups *Mobile Collaborating Communities* (MC^2), or simply *communities*. Each community member independently uses its WAN interface to create a communication *channel* to a remote inverse multiplexing or *aggregation* proxy, and optionally offers full or partial access to this channel to other community members. Each member volunteers to forward packets received on its WAN link to receiver(s) on the LAN. The set of channels connecting the participating community members to the proxy can be logically combined with an inverse multiplexing protocol to yield a higher-speed *aggregated channel* than is available from any one of the individual members. Hence, members using the aggregated channel enjoy higher bandwidth—and higher communication performance—than any one member alone could receive.

Striping data across multiple, parallel communication channels is a conventional communications technique used to improve system performance or reliability in varied but relatively static settings [2,3]. But due to end-device heterogeneity, mobility, and time-varying link transmission characteristics, an aggregated wireless channel is highly dynamic, and the challenge is to assemble, administer, and monitor its operation in a decentralized fashion.

In an earlier paper [1] we presented the initial design, simulation and implementation of a collaborative bandwidth aggregation system that is both practical and readily deployable. A key contribution of that work was to show that significant performance gains can be realized by adapting shared WAN link use to the specific application requirements of the flows sent over the aggregated channel. For a typical scenario, we demonstrated that the packet loss rate of a CBR video stream on an aggregated channel could be reduced by 71% by properly assigning packets to preferred links. But achieving these performance gains requires the aggregation system to be continuously aware of the communication characteristics of the constituent links.

In this paper we show that both WAN link communication performance as well as community membership dynamics must be accurately monitored and efficiently communicated to use an aggregated channel effectively. We explore the tradeoffs encountered in properly designing a decentralized monitoring system. Using a combination of *ns*-based simulations [4] and theoretical analysis we present a decentralized monitoring architecture and protocols designed to balance both system responsiveness and bandwidth efficiency. We also show how an inverse multiplexer should use measurements—possibly neither up-to-date nor consistent—to make decisions about proper channel use.

The rest of the paper is organized as follows. Sections 2 and 3 explore the requirements and issues associated with decentralized monitoring. Section 4 introduces a preferred monitoring architecture capable of meeting our system goals, and Section 5 presents simulation results exploring how effectively our proposed architecture balances the goals of responsiveness and bandwidth efficiency. An analysis of how an inverse multiplexing proxy should make decisions based on distributed, late arriving and possibly inaccurate measurements is presented in Section 6. Section 7 describes how proxies that perform functions in addition to bandwidth aggregation can greatly improve the efficacy of aggregated channels, both with and without the help of traffic sources and receivers. Our conclusions are drawn in Section 8.

2. Monitoring requirements and design goals

2.1. Background

Prior to discussing the requirements and design goals of a monitoring architecture we briefly review the design and operation of a bandwidth aggregation system. Fig. 1 shows a system that can be readily deployed by a network access pro-

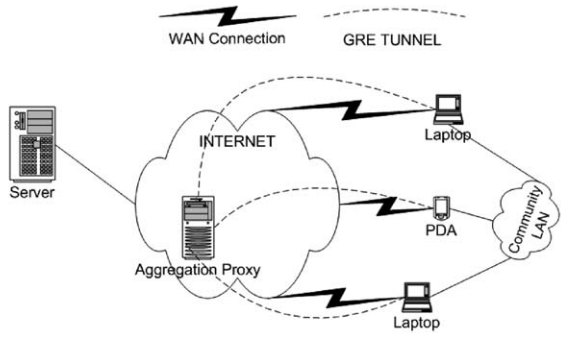

Fig. 1. A bandwidth aggregation service architecture.

vider, wireless telecommunication service provider, or a content distribution network operator. The specific implementation we have proposed has three principal components: a dedicated appliance providing aggregation proxy services, a standard LAN-based announcement and discovery protocol for mobile host community construction and maintenance, and standard protocol tunnels to facilitate both communication across shared links and packet forwarding at mobile hosts.

The dedicated aggregation proxy performs inverse multiplexing at the application layer, intelligently striping downstream packets across available links to the community. Generic routing encapsulation (GRE) [5] tunnels create channels between the proxy and participating MC^2 members, and support packet forwarding. This approach requires no modification to community members, as most operating systems (Linux, FreeBSD, Windows, etc.) today have built-in support for GRE tunnels. Each packet received by a member over the tunnel is automatically decapsulated and forwarded via the wireless LAN to the destination host. Since the destination is oblivious to which members forwarded the data packets, no additional data reassembly functionality is required at the receiver. Standard announcement and discovery protocols such as the service location protocol (SLP) [6] are relied upon for community and aggregated channel formation and management. More details about these system implementation choices and the performance of the prototype we constructed can be found in [7].

Aggregating wireless bandwidth to mobile hosts has also been considered by other researchers [8]. Connection sharing that enables use of a mobile device's single, idle WAN connection by other mobile devices, is studied in [9]. The goal of the mobile grouped devices (MOPED) project [10] is to make a user's set of devices appear as a single Internet-connected entity. The MOPED routing architecture builds a *multipath* layer to encapsulate packets between the home agent and user devices by using a new lightweight encapsulation protocol called *multipath routing encapsulation* (MRCAP). High-speed Internet connectivity is achieved by adapting the MobileIP home agent to support aggregation of multiple links at the network and transport layers.

Adaptive inverse multiplexing for CDPD wireless networks is examined in [11]. In this scheme packets are split into fragments of size proportional to the observed throughput of component links. Here the goal is to create variable fragments sizes such that each fragment can be transmitted in roughly the same amount of time. The fragment size of each link is dynamically adjusted in proportion to the link's measured throughput. The bandwidth of mobile users with multiple interfaces is aggregated at the transport layer in pTCP (parallel TCP) [12]. pTCP is a wrapper that interacts with a modified TCP called TCP-virtual (TCP-v). A TCP-v connection is established for each interface, and pTCP manages send buffers across the TCP-v pipes. Striping is performed by pTCP and is based on the congestion window size of each TCP-v connection. When congestion occurs on a certain pipe, pTCP performs data reallocation to another pipe with a larger congestion window.

The stream control transmission protocol (SCTP) [13] also provides reliable service between multihomed devices. Though in its current form SCTP only uses multiple links for redundancy, it can be easily extended to support striping and load sharing.

Though each of the above systems takes a different approach to tapping other users' communication resources, they share a common need to perform channel monitoring in order to use shared resources efficiently. Our objective is to design a channel monitoring architecture that could be used in a variety of settings, including the above systems.

2.2. Challenges of monitoring systems

An aggregation proxy is responsible for assigning incoming traffic flows to available WAN channels. We refer to this function as *flow mapping* or *channel control*. A proxy might also be able to modify the incoming flows themselves (i.e., source control). The goal of monitoring communication dynamics is to provide a proxy's channel and traffic controllers with prompt and accurate information on the condition and number of WAN channels available between the proxy and the community. Only with this information can a proxy perform intelligent channel and source control in the face of rapid changes to the communication channels. As we will explore in Section 6, one of the challenges for the flow mapper is how to use the measurement data it receives intelligently. For instance, frequently remapping flows to channels based on transient (fluctuating) channel quality measurements would not necessarily improve overall system performance [14].

We anticipate that both the availability and the quality of communication channels between a proxy and an MC^2 to vary with time. Community membership will change as mobile hosts join and leave the community, due to either end-system failures (e.g., power exhaustion) or simply moving out-of-range of LAN communications. Wireless WAN channel quality may change often and unpredictably because of fading, interference, and location-dependent coverage gaps. Delay and delay jitter will change as the heterogeneous, CPU-limited devices forwarding packets between WAN and LAN interfaces are subject to time-varying computing workloads. Hence, the parameters we expect our monitoring system to measure include:

- *Link quality*: raw and available bandwidth, delay, jitter, packet loss rate, signal strength.
- *Community membership*: number of available WAN channels, participation time in system.
- *Forwarding capability*: delay, jitter, available processing power.

Beyond a channel's communication parameters, certain associated information might also be maintained—but not necessarily measured—by the monitoring system. This might include the 'cost' of a channel, or its expected departure time.

Though we anticipate that a community member will be capable of explicitly announcing its pending departure (from the community) to other members, one of the most difficult challenges our monitoring system faces is rapidly detecting sudden and *unannounced* leaves. We envision a LAN-based monitoring agent capable of tracking membership, including announced leaves and new members' joins. Such an agent would likely rely on an existing service discovery protocol, and a new member joining the MC^2 would register its identity and present available resource information. Such a system would likely have to be supplemented with an active mechanism to detect leaves. For example, the monitoring agent can periodically issue an echo request message (e.g., *ping* or *hello*) to active members and await a reply. The question of how often the monitoring agent should probe the members arises immediately. Clearly, there is a tradeoff between the probing overhead and the freshness of membership information. While we cannot afford to have excessive control message overhead in membership maintenance, we will typically assume that LAN bandwidth is a relatively plentiful resource.

To illustrate the importance of low latency in reporting WAN channel status to the aggregation proxy in improving the performance of an aggregated channel, we simulated an aggregation system with three community members. Each member offered a WAN channel with 20 kb/s bandwidth. Each channel has a time-varying packet loss rate (unknown to the proxy) that cycles as follows: a loss rate of 1% for 50 s, followed by a loss rate of 5% for 50 s, and then a loss rate of 10% for 50 s. The cycle is repeated multiple times during the lifetime of the session. The changes in loss rates across the three links are synchronized such that at any instant there is exactly one channel that has error rate of 1%, one channel with 5% and one channel with 10%. Thus, the total error rate is the same throughout the experiment.

An application-aware aggregation proxy [7] seeks to map hierarchically layer-coded [15] video to these three available channels. The simulated layered video consists of base layer (layer 0) and two enhancement layers (layers 1 and 2). Each layer is modeled as a 20 kb/s CBR stream. Using the channel loss rate as the reliability metric, the aggregation proxy maps each layer onto one of the three channels, ideally with higher layers assigned to increasingly less reliable channels; we referred to this flow assignment as the *layer priority striping* (LPS) algorithm in [7]. Fig. 2 shows the packet loss rate of each layer when the reporting latency (i.e., feedback delay) is varied. The feedback delay is defined as the time difference between the instant when the channel error rate changes and the time when the aggregation proxy remaps the flows onto the channels based on the newly-available information. As expected, the feedback delay decreases aggregated channel performance; the base layer is not transmitted over the most reliable channel during the feedback delay period following each loss rate transition event. In fact, when the feedback latency is larger than 18 s, the loss rate of the base layer exceeds that of enhancement layer 1.

In general, the change in layer i's packet loss rate l_i is proportional to the feedback delay δ. Let the duration of a session be $N * T$ seconds where the link loss rate changes every T seconds. Let $P(i,j,k)$ be the packet loss rate of the channel during period k to which layer i has been assigned in period j. Then, layer i's packet loss rate can be written as

$$l_i = \frac{\sum_{j=1}^{N} P(i,j,j) * (T-\delta) + P(i,j-1,j) * \delta}{N * T}.$$

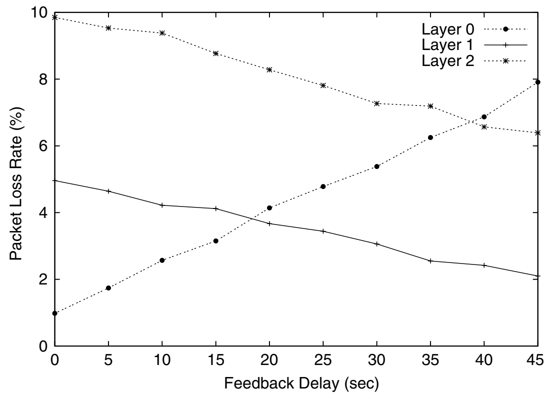

Fig. 2. The effect of reporting latency on aggregation performance.

In the above example we assumed that the aggregation proxy received *correct* measurements late. But measurement errors can also cause suboptimal mappings of application subflows to WAN channels. Hence, it is important for a monitoring system to measure and report the channel conditions accurately, and a tension exists between taking the time required for accurate measurements and keeping reporting latency short. In certain situations the system will tolerate even large measurement errors and continue to perform well. For instance, in the above example, even substantial errors in measuring link reliability would maintain the optimal channel ordering from most to least reliable.

In summary, our design goals for the overall monitoring system are:

- accurate measurement of link quality,
- low latency in reporting changes in link quality and community membership,
- low control message overhead,
- no or little software modification to the community members,
- minimal member performance degradation due to community participation,
- scalable design to support multiple aggregation proxies and large community memberships,
- scalable aggregation proxy capable of supporting a large number of communities simultaneously, and
- robustness to failures of members and their channels.

3. Design choices

Designing an effective monitoring system forces us to face a variety of issues, including:

- *Architecture*: at which locations in the system should monitoring be performed? How do we design a scalable monitoring architecture capable of supporting both large community sizes and multiple proxies? What protocols should be used to feed the monitored information back to the aggregation proxy?
- *Measurement*: how should WAN channel communication performance and community mem-

bership dynamics be measured? Should measurement rely on active or passive techniques, or both? How do we minimize the burden of measurement placed on community members?

- *Configuration*: how do we dynamically set design parameters (e.g., proxy update interval, measurement intervals, active membership probing intervals) particularly as the community size and traffic changes? At what point should an aggregation proxy use measurement data it receives to decide to remap flows to available channels?

In the rest of this section we investigate design choices related to the above questions and discuss their strengths and weaknesses. This investigation will lead us to present a monitoring architecture in Section 4 which balances the many tradeoffs we must make.

The first and most important architectural issue we face is identifying the location of measurement points in the system. A *monitoring agent* will perform measurements at each of these points, and exchange information between themselves and the aggregation proxy. These agents may reside on one or more community members (i.e., mobile hosts), at the aggregation proxy, or both; we will exclude from our discussion the possibility that any type of dedicated equipment be required for monitoring, as that would preclude spontaneous formation of an MC^2.

3.1. Community member-based monitoring

An agent may be located at one or more community members to monitor WAN channel condition and membership dynamics. Let us consider how such a system would operate. An arriving host seeking to participate in a pre-existing community discovers the community using a service discovery protocol (e.g., SLP) and registers with the monitoring agent(s). A member seeking to leave the community (i.e., an announced departure) broadcasts a departure notice to the community, and is deregistered by the monitoring agent(s). An active mechanism is used by monitoring agents to detect unannounced departures; an agent periodically probes the existence/condition of the community members. In such a case, the probing period is an important design parameter and must be determined by making a tradeoff between the probing overhead and the accuracy of the monitored information. On a high-speed LAN (e.g., IEEE 802.11x) the messaging overhead is not a significant issue, but the processing load and power consumption the agent imposes on a community member is an important issue. This is a particular concern if relatively few of the community members are providing monitoring services, such as when a single member is appointed or elected as the sole monitoring agent. The fact that a member serving as a monitoring agent consumes more power and processing than a regular member suggests that it is beneficial to have the agent's role rotated or shared among members. This also argues for power and processing availability at each node to be included in those parameters that are measured and maintained by monitoring agents.

The above sketch of system operation serves to highlight several of the advantages of deploying monitoring agents at a community member. A community member can quickly and easily track membership changes. But while a member can assess the quality of its own WAN channel to the proxy, it has very limited visibility of the characteristics of other WAN channels. Moreover, a protocol must be established for identifying the members to serve as agents. Clearly, relying on a single (or even a few) monitor(s) can result in both a performance and reliability bottleneck.

This bottleneck problem can be solved by either replicating the monitoring agent or making every member a monitoring agent, i.e., distributed monitoring. We opt to use distributed monitoring which works as follows. Each member broadcasts its channel characteristics and associated information (e.g., communication costs and its energy balance) periodically, upon detection of an event, or when a threshold is exceeded. Each broadcast is timestamped. Upon receiving such a broadcast all the other members update the corresponding entry of their copy of the community communication status database. The aggregation proxy obtains a copy of the database in either of two ways. First, the proxy requests a copy of the database from any community member. Requests can be sent to a randomly-selected member, or a member identified by inspection of the most recent database the proxy has received. For example, an inquiry might be directed to a member with ample advertised available processing power, residual energy, or network bandwidth. A proxy might issue such an inquiry periodically, or be driven by an event such as the need to remap channels for a newly-arriving flow. The second way that a proxy obtains the database is simply by receiving an update report periodically or when a monitoring agent observes a significant local event (e.g., sudden channel failure).

Such a decentralized monitoring system is very attractive because it clearly improves overall system reliability and eliminates a potential bandwidth bottleneck. Note that each member's database need not be a perfect representation of current system state. Making each member a monitoring agent provides the best overall visibility of conditions of every channel.

3.2. Proxy-based monitoring

An alternative measurement architecture places a single monitoring agent at the location where the WAN channels terminate and the channel allocation is done. Depending on the link technology, a proxy may be able to detect an indication of a WAN channel failure rapidly. In other cases a proxy-based monitor might be able to infer failures over longer time periods. For example, a proxy observing a long duration flow using a transport protocol with end-to-end feedback (e.g., TCP) might conclude that a failure has occurred if traffic associated with that flow trickles to a halt. Here a proxy is using TCP as an implicit monitor of channel characteristics. Observing multiple coincident TCP rate decreases across multiple flows sharing a single channel would be a stronger indication of a failure.

A proxy-based monitoring system has the great advantage of simplicity; monitoring agents do not have to be deployed at members, no coordination is required, and no protocols need be defined. But the proxy's single vantage point provides low visibility to overall system state. Indeed, when a channel failure *is* detected a proxy is unlikely to know the cause, or other related effects.

3.3. Hybrid proxy- and member-based monitoring

It is clear that a combination of proxy- and member-based monitoring can be used to capture the most information about the current state of the system. As we demonstrated in Fig. 2, providing the proxy with the most complete and up-to-date measurements improve channel allocation decisions and overall system performance. However, as the amount of measurement information that a proxy receives increases, the proxy is faced with ever more complicated decisions about how to allocate channels. Section 6 illustrates this complexity by providing an analysis of a proxy facing a simple binary decision.

When supporting multiple aggregation proxies, independent of the above described monitoring architectures, the proxies need to communicate with each other to exchange information of overlapping channels and members.

3.4. Measurement techniques

Though our monitoring system relies on the ability to measure channel characteristics, our focus is to identify appropriate existing measurement techniques, not invent them. There are numerous approaches to measuring and estimating link bandwidth and delay in the Internet [16]. Active probing schemes typically use `pathchar` [17] to obtain link information [18]. The RTT of each hop is measured for variable packet sizes to calculate link bandwidth [19]. Packet pairing [20–23] is another popular technique for estimating link bandwidth. In this scheme, end-to-end path capacity is obtained from the dispersion between two packets of the same size transmitted one after the other. A centralized approach for measuring bandwidth and delay using tools such as SNMP [24] and IP probes is proposed in [25].

Passive measurement schemes such as SPAND [26] do not use any probing messages and instead rely on observing traffic generated by the applications. In wireless networks radio signal-to-noise ratio (SNR) can be used to estimate hop-by-hop wireless link bandwidth [27]. SNR information can often be provided by a wireless network interface card (e.g., IEEE 802.11x card). A network service architecture that collects wireless channel conditions and provides them to the applications is proposed in [28].

4. Monitoring system for MC^2

We now describe a distributed monitoring architecture designed to meet the various system requirements and goals introduced earlier. The proposed architecture is decentralized; every community member participates in monitoring. Each member has a monitoring agent which joins a well-known multicast group G_m for exchanging community status information. Each monitoring agent broadcasts a *local report* R_l addressed to G_m on the LAN. Each local report contains information about the current state of the member and its offered WAN link(s). An illustrative format of a local report is shown below:

```
struct member_state {
  unsigned int  member_id;
  double        battery_power;
  double        cpu_load;
  /* One or more of the link_state
  records */
  struct link_state {
    unsigned int  link_id;
    unsigned int  timestamp;
    double    bandwidth_estimate;
    double    loss_estimate;
    double    forwarding_delay;
    double    lifetime;
    double    signal_strength;
  }
}
```

The *ttl* of the local reports is set to 1 to restrict its scope to the LAN. Upon receiving a local report from member m_i, each member updates the information about member m_i in its locally-maintained community status database. In steady-state each member has up-to-date information about all community members. Each member issues a single packet containing the local report once every local reporting interval I_l. Though local report traffic grows linearly with the number of community members, this is not a concern for the following reasons. First, LAN bandwidth is plentiful, and report sizes are small.[1] Messaging overhead will be limited, and actions described below will help avoid redundant information exchange.

The collective information about the community members is sent to the inverse multiplexing proxy in *proxy reports* R_p. The community reports its current state to the proxy once every proxy reporting interval I_p. Instead of electing a particular member to send proxy reports, every member shares the responsibility. Each member sets a suppression timer for duration of $I_p + \delta$, where δ is a uniform random variable on $[0, S_d]$. Upon expiration of its suppression timer, member m_i sends R_p to the proxy via its local WAN link, and also multicasts the same report to the community on group G_m. Upon receipt of the multicast proxy report the members other than m_i cancel their suppression timers and report transmissions. At the same time, each member reschedules timers to send a proxy report for the next interval. Since R_p has the latest information about all the members, newly-arriving members that have incomplete information about the community obtain complete system information quickly. Maintaining a distributed database is also advantageous for other reasons. Decentralization alleviates the potential problem of a control traffic bottleneck by spreading the traffic over multiple WAN links. Sharing responsibilities does not put an undue burden on any one node, provides fault-tolerance, and system reliability remains high even in a challenging 'high turnover' environment where members are arriving and departing at very high rates. A *soft-state* approach is used for maintaining member information in the monitoring databases. If the state is not periodically refreshed it is purged from the database. This approach also serves to purge the database of records of members who departed silently.

[1] Report sizes can be even smaller when schemes such as delta encoding [29] are used.

The system designer should configure the monitoring system to achieve high system responsiveness while limiting report traffic. Note that the maximum time between a state change and the proxy's knowledge of it is bounded by $I_l + I_p + S_d$. Increasing the reporting intervals I_p and I_l reduces both messaging traffic and responsiveness. Properly configuring these timers is challenging, as the optimal values depend upon community membership dynamics, the time-varying communication characteristics of WAN links, and the requirements and dynamics of the flows sent over the aggregated channel.

Where possible we opted to use passive methods for measuring channel characteristics. For example, it is reasonable to assume that each member has access to and monitors physical layer information such as the SNR of its wireless links. In some cases this information can also be used to estimate link quality parameters such as loss rate and bandwidth that are advertised in the local reports.

5. Simulation experiments

To explore the challenging problems associated with monitoring system configuration we turned to an *ns-2* based simulation [4]. To begin we set the value of both reporting intervals I_l and I_p to 1 s. Fig. 3 shows the number of reports sent to the proxy in each proxy reporting interval I_p. As desired, the number of reports per reporting interval stays close to 1 even as the number of community members increases. The suppression algorithm is only slightly less effective in preventing multiple reports per interval in large communities (i.e., occasionally two reports are sent in one interval). If necessary, the number of instances of multiple reports can be reduced further by increasing the value for parameter S_d which controls the spread of the suppression timers. Fig. 3 also plots the average number of proxy reports per interval sent by each member (R_p/I_p per member) with error bars showing the maximum and the minimum. As the community size increases the number of reports sent by each member declines as the reporting task is distributed across all community members. Note that the variability of the reports issued from member to member is very little; the reporting task is fairly equally split between all the members.

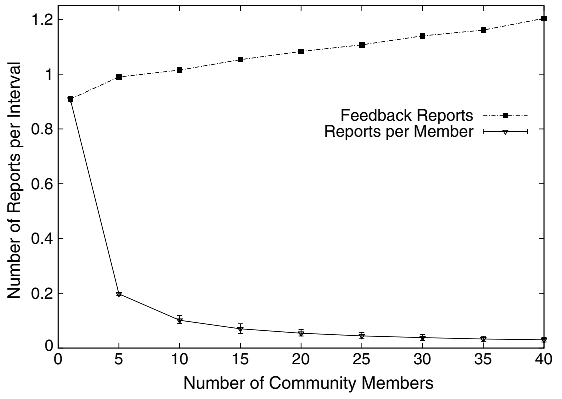

Fig. 3. The number of proxy reports issued per member per reporting interval I_p, and the average number of reports received by the proxy per reporting interval I_p.

Though in our simulations all the members participated equally in the reporting process, in practice members will have differing capabilities (e.g., remaining battery life, compute power), so the system should permit different levels of participation by different members. Only members with sufficient memory and WAN bandwidth need to collect the information from the other members and share the load of informing the proxy. Biased suppression timers are one means of achieving this type of load balancing; more capable members can simply set shorter suppression timers (smaller value of S_d).

We also studied how the feedback latency varies with different settings of the reporting interval I_p. For this study, we generated a sequence of 100 events, each representing a change in the link state (such as bandwidth or loss rate) of a particular member. A member was chosen randomly from a 10-member community for each event. A change event occurs every 10 s period at a random time picked from a uniform distribution on [0, 10]. The average feedback latency for this sequence of 100 events is shown in Fig. 4 with the error bars showing the maximum and the minimum. As expected, the average feedback latency increases as I_p increases. We also observe that the maximum feedback latency is bounded by the reporting interval

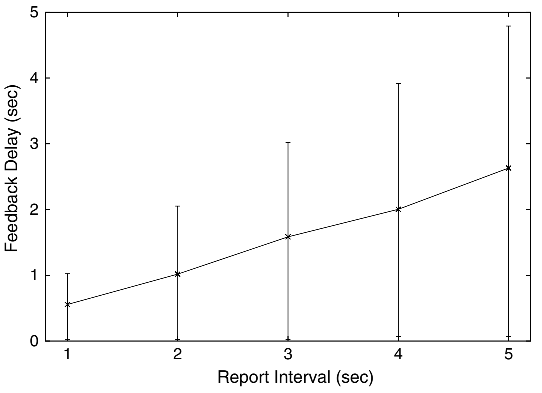

Fig. 4. The effect of reporting interval I_p on feedback latency.

I_p. Although the feedback latency is low for small values of I_p, the amount of reporting traffic is large. This tradeoff between reporting overhead and reporting latency can have a significant effect on overall system performance because the WAN bandwidth between the agent and the proxy is relatively scarce, and the channel carries both data and control traffic. The reporting interval can be increased without greatly affecting the feedback latency by generating reports that are triggered by a significant event, e.g., a member departure, a measured channel characteristic exceeding a certain threshold. Different application requirements also need to be considered when evaluating this tradeoff between overhead and latency.

6. Design and analysis of measurement-based decision algorithms

The aggregation proxy receives reports from community-based monitoring agents and must decide on a preferred assignment of arriving packets to available links. Such an assignment might, for example, rely on an ordering of links according to measurements of reliability (i.e., packet loss), delay, or throughput. In this section we consider the proper design of the proxy's decision algorithm. In general, we seek an algorithm capable of responding rapidly to changes in link communication performance, while avoiding the potentially costly overhead associated with unnecessarily frequent reassignments.

Some care must be taken in designing an algorithm that uses link measurements. Measurement accuracy can be heavily dependent on the underlying measurement techniques themselves. The system we seek must be capable of responding effectively to both relatively slowly-varying link communication performance characteristics, as well as sudden, unexpected link failures. In general, while better measurement accuracy can be achieved by both longer measurement intervals and sophisticated measurement techniques, the former can reduce overall system responsiveness and the latter can increase overall system costs or overhead.

Suppose we use measurements of WAN packet loss as an example; we expect communication links of mobile, wireless devices to exhibit time-varying packet loss behavior. This variability can easily lead to frequent, and possibly unnecessary, changes in a proxy's hypothesized ordering of links by reliability. Indeed, a poorly-designed allocation algorithm might cause a proxy to hypothesize a different ordering in nearly every measurement interval. We will next show how common this problem is—even for links with stationary loss processes and relatively little variability. We will then introduce approaches to the design of a proxy's decision algorithm to avoid such undesired oscillations.

Suppose that for each of M links $L_i : i = 1, 2, \ldots, M$ we measure the packet loss rate during consecutive, non-overlapping intervals of duration T seconds and model each link's loss rate as a sequence of continuous-valued random variables $x_i(t)$, i.e., $x_i(t), x_i(t+T), x_i(t+2T), \ldots$ As an illustration we will assume that the loss rate on each link is independent from interval to interval and is time-homogeneous, i.e., $x_i(t) \equiv x_i$.

Suppose that in each measurement interval we order the links according to their *measured* loss rate (in that interval only) from the least reliable to the most reliable. What is the likelihood that we will see a different ordering from one interval to the next? If the joint probability density $f_{\vec{x}}(x_1, x_2, \ldots, x_M)$ of the M random variables measuring the packet loss in each interval is known then we can write the probability of any specific ordering, e.g.,

$$\Pr[x_1 > x_2, \ldots > x_M] = \int_{x_1=0}^{1} \int_{x_2=0}^{x_1} \cdots \int_{x_M=0}^{x_{M-1}} f_{\vec{x}}(x_1, \ldots, x_M)\, \mathrm{d}x_1 \cdots \mathrm{d}_M. \quad (1)$$

In general, of course, we do not know the joint probability density f. However in some cases it is reasonable to assume that the losses will be independent from link to link, such as when the communication links are provided by different operators, or if the physical link technologies are dissimilar. We could then write the joint density as

$$f_{\vec{x}}(x_1, x_2, \ldots, x_M) = \prod_{i=1}^{M} f_{x_i}(x). \quad (2)$$

Let us begin with what might be a common, albeit worst case. Suppose that each link's loss rate is well modeled by a uniform random variable on (l,h) with $0 \leqslant l < h \leqslant 1$ or

$$f_{x_i}(x) \equiv f_x(x) = \begin{cases} \frac{1}{h-l} & l < x < h, \\ 0 & \text{elsewhere.} \end{cases} \quad (3)$$

That is, each link's loss rate is independent and identically distributed; no link is more reliable than any other. If we substitute Eqs. (3) and (2) into Eq. (1) and integrate we can find the probability that the link measurements indicate *any* arbitrary ordering by loss rate is simply

$$\Pr[x_1 > x_2 \cdots > x_M] = \frac{1}{M!}, \quad (4)$$

which, as we expect, equals the multiplicative inverse of the number of permutations of M links.

This simple result tells us that for a system with as few as $M = 4$ links with statistically identical loss characteristics the probability is $1 - \frac{1}{4!}$ (i.e., greater than 95%) that from one interval to the next the measurements will indicate a different ordering of link reliability. Note that this is the case independent of the variability of measurements on the links. Clearly we seek to avoid designing a system that reallocates links in nearly every measurement interval, and indeed, in this case the system would be doing so by switching between links which would yield no long term advantage.

The difficulty in correctly identifying the most reliable links occurs even when the link loss rates are dissimilar; even when a clear ordering of reliability is known, measurements can frequently indicate a different ordering. For example, suppose that we have a system of four links that are known to have independent uniform densities $f_x(x), f_x(x + \Delta), f_x(x + 2\Delta), f_x(x + 3\Delta)$, each density successively translated by a constant amount Δ. That is, the average loss rate of link i is Δ higher than link $i - 1$. Though the reliability ordering of the four links is clear, it is still the case that our measurements will frequently steer us wrong. Substituting these densities into Eq. (1) and evaluating can tell us the probability that we fail to guess the correct ordering in each measurement interval. Fig. 5 shows how frequently we guess wrong as we increase the value of the shift Δ.

One means of avoiding oscillation in a measurement-based channel allocation system is to introduce both memory of previous measurements as well as hysteresis in the proxy's decision algorithm. As a simple illustration, suppose we have a system with two links which our proxy seeks to order according to their measured reliability, while limiting potentially frequent and costly switching between the two links. Let $m_t \in \{0, 1\}$ correspond to the link *measured* as most reliable in time interval t, and let state $l_t \in \{0, 1\}$ correspond to the link the proxy's decision algorithm has selected as the most reliable at time t (based on measurements

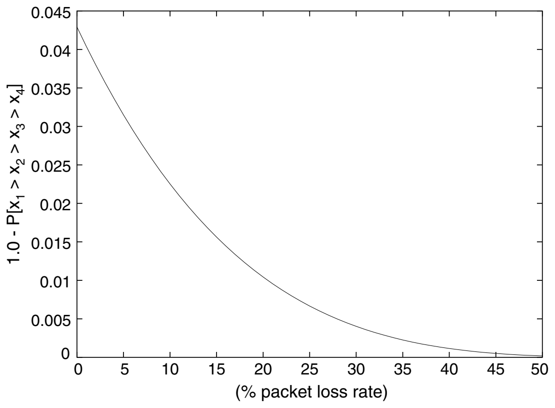

Fig. 5. The probability that link ordering changes from one measurement interval to the next as the shift Δ increases.

in both the current and past intervals). Suppose the proxy selects the most reliable link by using the measurements it has received in each of the last N measurement intervals as follows:

- If in state 0 a proxy will decide to switch to state 1 if $N - D$ of the last N measurements indicate that state 1 is more reliable.
- If in state 1 a proxy will decide to switch to state 0 if $N - D$ of the last N measurements indicate that state 0 is more reliable.

The constant D, $0 < D < N$, is a parameter whose value determines the thresholds at which the proxy decides to switch the link it sees as most reliable. We can model the two link system as shown in Fig. 6, where state $\{k, l\}$ indicates that k of the last N measurements indicated that link 1 was most reliable, and the system is in state l (i.e., the proxy has decided that link l is more reliable). We may write the state transition probabilities as follows. Suppose the probability that link 1 is measured as most reliable in each interval is p. Then the transition from state $\{k, l\}$ to $\{k + 1, l\}$ requires that link 1 was measured as most reliable in the current interval *and* that link 0 was measured as most reliable N intervals ago, conditioned on the event that link 0 was identified as most reliable in k of the $N - 1$ intervening intervals. We write this probability as

$$a(k) \triangleq \Pr[\{k+1, l\} | \{k, l\}] \tag{5}$$

$$= p \cdot \frac{\binom{N-1}{k} p^k (1-p)^{N-k-1}}{\binom{N}{k} p^k (1-p)^{N-k}} \cdot (1-p) \tag{6}$$

$$= p\left(1 - \frac{k}{N}\right), \quad k = 0, 1, \ldots, N. \tag{7}$$

The equations for the remaining transition probabilities are found similarly, and are as follows:

$$b(k) \triangleq \Pr[\{k-1, l\} | \{k, l\}] \tag{8}$$

$$= (1-p)\left(\frac{k}{N}\right), \quad k = 0, 1, \ldots, N, \tag{9}$$

$$c(k) \triangleq \Pr[\{k, l\} | \{k, l\}] \tag{10}$$

$$= (1-p)\left(1 - \frac{k}{N}\right) + p\left(\frac{k}{N}\right). \tag{11}$$

Fig. 6 depicts the model of a system with a memory of $N = 10$ previous measurement intervals and $D = 3$. Table 1 shows the steady-state probabilities for this system for the case where it is equally likely that measurements reveal one link as more reliable than the other in each interval. The two state system changes state every

$$a(N - D - 1) \cdot p[N - D - 1, 0] + b(D + 1) \cdot p[D + 1, 1] \tag{12}$$

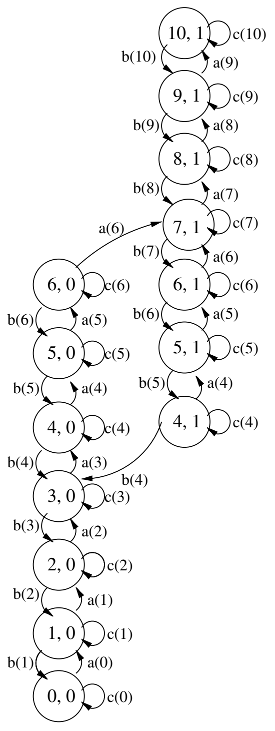

Fig. 6. A chain modeling a two link (state) system with $N = 10$ and $D = 3$.

Table 1
State probabilities for the two state system of Fig. 6 with $p = 0.5$

$p[0,0] = 0.000977$	$p[4,1] = 0.061523$
$p[1,0] = 0.009766$	$p[5,1] = 0.123047$
$p[2,0] = 0.043945$	$p[6,1] = 0.143555$
$p[3,0] = 0.117187$	$p[7,1] = 0.117188$
$p[4,0] = 0.143555$	$p[8,1] = 0.043945$
$p[5,0] = 0.123047$	$p[9,1] = 0.009766$
$p[6,0] = 0.061523$	$p[10,1] = 0.000977$

measurement intervals. For the case of the system with $N = 10$, $D = 3$, and $p = 0.5$, this average period equals 20.317 measurement intervals, indicating far less frequent switching than if the decision algorithm did not employ both memory and hysteresis. For a given measurement reporting period T, the algorithm designer can set values of the measurement memory capacity N and the transition threshold D and use Eq. (12) to balance system responsiveness with an acceptable switching frequency.

7. Optimizing performance using joint channel and traffic control

Up to this point we have focused attention on how an aggregation proxy uses feedback about WAN channel characteristics to perform channel selection. To the extent that a proxy is aware of each flow's application requirements link assignments can be made that maximize the utility of those assignments. For example, a proxy might assign the base layer of a layered video stream to a reliable channel, and the enhancement layer to a less reliable channel.

In this section we consider how a proxy can optimize channel selection for *adaptable* flows. Adaptable flows have traffic characteristics (e.g., bandwidth) that a proxy can either directly modify or cause the traffic source to modify. Given an adaptable flow, a proxy cannot only select WAN links most appropriate for the flow, but also modify the flow to more closely match the characteristics of those available links. Multiple description video is one example of such traffic; a proxy can choose to match the stream to limited available bandwidth by dropping as many components as necessary. Done properly, a receiver will perceive higher quality of service if the bandwidth is reduced to the available channel capacity. Discarding the forward error correction (FEC) information associated with a stream is a second example of an 'end-system-blind' traffic adaptation that a proxy can perform without the cooperation of either the traffic source or receiver.

In general, however, the number of such blind adaptations that an aggregation proxy can enforce is rather limited. But a more intelligent proxy—one that provides integrated traffic management services in addition to channel aggregation—can potentially do more. An intelligent *multifunction* proxy can communicate WAN channel information back to a source capable of adapting its traffic. At the transport level, this information could simply be a congestion notification (e.g., an ICMP source quench). This type of feedback could be particularly valuable in cases where the proxy has received information from WAN endpoints of a *pending* event, such as the anticipated departure of a community member (and its associated WAN channel).

More sophisticated multifunction proxies can also be envisioned, such as one that could perform both channel aggregation and RTSP [30] proxy services. For a media-on-demand session, such a proxy could issue RTSP commands to renegotiate transport parameters for a session in progress. In other examples, a multifunction proxy could transcode a video stream to match channel bandwidth [31], or supplement a media stream with an associated FEC stream specifically fashioned to overcome the WAN link packet loss characteristics known to the proxy.

For most data traffic, TCP's end-to-end congestion control mechanism is adequate for adapting sources to aggregated channels. But the mismatch between separately designed adaptable traffic sources and aggregated channels can be best demonstrated by examining the behavior of streaming media over TCP. The overwhelming amount of media traffic streamed today on the public Internet is sent over TCP—or more specifically HTTP. We next explore why this adaptable traffic class often performs poorly on aggregated channels, and suggest that a multifunction aggregation proxy can serve as a remedy.

Stored audio and video available on demand from a media server is frequently encoded at multiple bit-rates to support rate adaptation for transmission over either congested or low bandwidth links. For example, Real Network's Surestream technology [32] supports multiple bit-rate encoding and midstream rate adaptation in the face of network congestion. However, the number of permissible rates is typically few, and the granularity of rate adjustments is often large; we refer to this as *coarse-grain rate control*. As an example, music might be encoded as MPEG-2 layer III audio at the set of rates of 128, 96 and 64 kb/s. A media server will typically begin transmitting a stream at the highest available rate, and reduce the rate until the detected frequency of packet loss indicated by receiver feedback is acceptably small.

Sharing network resources with data applications encourages media applications to use either TCP-friendly rate control mechanisms [33,34], or TCP itself. But TCP's aggressive probing for available bandwidth is poorly suited to a multirate source's ability to make only large, discrete rate adjustments. Further, a media application has no effective means of communicating its bandwidth needs to TCP, while the transport layer provides little help to the application seeking a large rate increase. In fact, an attempt to increase rate in the face of insufficient available bandwidth can result in lower perceived service quality than no attempt at all.

Media servers can and do switch between components of multiple bit-rate encoded content in response to the reception feedback received from the client. But our experience with media servers suggests that while they effectively reduce rates in the face of congestion, few if any increase rate upon the return of sufficient bandwidth. Hence, bandwidth added in mid-session via WAN channel aggregation fails to be captured to augment the quality of media delivered to receivers.

Why do media servers opt not to increase rate when additional bandwidth capacity becomes available? The first reason is that they are typically unaware of the available bandwidth. That is, without taking additional action to either measure or capture available bandwidth, they receive no explicit indication of bandwidth availability. A second reason is the recognition by application developers that frequent changes to media quality level—even for the better—are perceived as disruptive by viewers and listeners. A third reason is that it is not obvious when and how to best re-establish the higher rate. An application can attempt to either actively probe for available bandwidth (perhaps out-of-band using 'dummy data') or blindly attempt to grab available bandwidth by just beginning to send data at the desired higher rate.

But because of the typically large separation in encoding rates, simply sending data at the higher rate will result in packet loss if insufficient bandwidth is available to support that higher rate. Such packet loss can adversely affect quality as it causes TCP's reduction of the congestion window that results in playout buffer starvation and playout pauses. The unfortunate result can be that the attempt to acquire more bandwidth can result in service disruption at the client, who might have realized better quality had the media server merely stayed put at the lower bandwidth.

How can a multifunction proxy help with efficient coarse grain rate increases by a media server? Let us assume that the aggregated channel represents the bandwidth bottleneck in the end-to-end connection. A multifunction proxy can receive measurements from the monitoring system and inform the media source when additional bandwidth becomes available. The traffic source would respond to the receipt of this information by choosing to switch to a higher rate transmission. The advantage of such an approach is that a media source would risk increasing its rate only at those times when it is highly likely that it will be able to reach and sustain a higher rate. An additional benefit of this approach is that the multifunction proxy can implement a sharing policy to divide available bandwidth between multiple streams. For example, a proxy might inform lower rate media sources of available bandwidth prior to higher rate sources in an attempt to share bandwidth fairly.

While the above example illustrates how a multifunction proxy can potentially improve end-to-end performance by jointly controlling channel selection and the carried traffic, the ability to achieve these performance gains rests heavily on our ability to craft an accurate and robust link monitoring system.

8. Conclusion

We have designed and evaluated the performance of a decentralized channel monitoring system to support wireless bandwidth aggregation. An architecture that fairly distributes the burden of monitoring among community members can be made highly robust and responsive while limiting control message overhead. The monitoring architecture we have proposed in this paper is independent of the specific implementation of link aggregation, and can be used to support other aggregation and channel sharing systems [9–11].

Aggregating low-speed links to form a higher-speed logical link appears deceptively simple in principle. But as the communication characteristics of the underlying links grow increasingly erratic—as is the case in the challenging mobile setting we consider—potential performance improvements can vanish quickly.

Perhaps the simplest demonstration of this is the case of dividing a TCP flow across two links. Suppose that one link is extremely reliable, but the second rapidly fluctuates between functioning and failing. Then, lower throughput can easily result by aggressively trying to use both links rather than simply settling for the throughput that can be realized with the reliable link. Hence a monitoring system that can accurately track communication link behavior and promptly inform a channel aggregator is crucial to achieving real performance gains in a practical bandwidth aggregation system.

References

[1] P. Sharma, S.-J. Lee, J. Brassil, K.G. Shin, Handheld routers: intelligent bandwidth aggregation for mobile collaborating communities, in: Proceedings of IEEE BROADNETS, San Jose, CA, 2004, pp. 537–547.

[2] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, D.A. Patterson, RAID: high-performance, reliable secondary storage, ACM Computing Surveys 26 (2) (1994) 145–185.

[3] C.B.S. Traw, J.M. Smith, Striping within the network subsystem, IEEE Network 9 (4) (1995) 22–32.

[4] *ns-2*, the network simulator. Available from: <http://www.isi.edu/nsnam/ns>.

[5] D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, Generic routing encapsulation (GRE), RFC 2784, IETF, March 2000.

[6] E. Guttman, C. Perkins, J. Veizades, M. Day, Service Location Protocol, version 2, RFC 2608, IETF, June 1999.

[7] P. Sharma, S.-J. Lee, J. Brassil, K.G. Shin. Handheld routers: Intelligent Bandwidth aggregation for mobile collaborating communities, Technical Report HPL-2003-37R1, HP Laboratories, May 2003. Available from: <http://www.hpl.hp.com/techreports/2003/HPL-2003-37R1.html>.

[8] D.S. Phatak, T. Goff, A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments, in: Proceedings of IEEE INFOCOM, New York, NY, 2002, pp. 773–781.

[9] M. Papadopouli, H. Schulzrinne. Connection sharing in an ad hoc wireless network among collaborative hosts, in: Proceedings of NOSSDAV, Florham Park, NJ, 1999, pp. 169–185.

[10] C. Carter, R. Kravets, User device cooperating to support resource aggregation, in: Proceedings of IEEE WMSCA, Callicoon, NY, 2002, pp. 59–69.

[11] A.C. Snoeren. Adaptive inverse multiplexing for wide area wireless networks, in: Proceedings of IEEE GLOBECOM, Rio de Janeiro, Brazil, 1999, pp. 1665–1672.

[12] H.-Y. Hsieh, R. Sivakumar, A transport layer approach for achieving aggregate bandwidth on mutli-homed mobile hosts, in: Proceedings of ACM MobiCom, Atlanta, GA, 2002, pp. 83–94.

[13] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, Stream Control Transmission Protocol, RFC 2960, IETF, October 2000.

[14] M. Kim, B. Noble, Mobile network estimation, in: Proceedings of ACM MobiCom, Rome, Italy, 2001, pp. 298–309.

[15] S. McCanne, M. Vetterli, Joint source/channel coding for multicast packet video, in: Proceedings of IEEE ICIP, Washington, DC, 1995, pp. 25–28.

[16] R.S. Prasad, M. Murray, C. Dovrolis, K. Claffy, Bandwidth estimation: metrics, measurement techniques, and tools, IEEE Network 17 (6) (2003) 27–35.

[17] Pathchar. Available from: <http://www.caida.org/tools/utilities/others/pathchar>.

[18] A.B. Downey, Using pathchar to estimate Internet link characteristics, in: Proceedings of ACM SIGCOMM, Cambridge, MA, 1999, pp. 241–250.

[19] K. Lai, M. Baker, Measuring link bandwidths using a deterministic model of packet delay, in: Proceedings of ACM SIGCOMM, Stockholm, Sweden, 2000, pp. 283–294.

[20] J.-C. Bolot, End-to-end packet delay and loss behavior in the Internet, in: Proceedings of ACM SIGCOMM, San Francisco, CA, 1993, pp. 289–298.

[21] C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure? in: Proceedings of IEEE INFOCOM, Anchorage, AK, 2001, pp. 905–914.

[22] K. Lai, M. Baker, Measuring bandwidth, in: Proceedings of IEEE INFOCOM, New York, NY, 1999, pp. 235–245.

[23] V. Paxson, End-to-end Internet packet dynamics, IEEE/ACM Transactions on Networking 7 (3) (1999) 277–292.

[24] J. Case, M. Fedor, M. Schoffstall, J. Davin, A simple network management protocol (SNMP), RFC 1157, IETF, May 1990.

[25] Y. Breitbart, C.-Y. Chan, M. Garofalakis, R. Rastogi, A. Silberschatz, Efficiently monitoring bandwidth and latency in IP networks, in: Proceedings of IEEE INFOCOM, Anchorage, AK, 2001, pp. 933–942.

[26] M. Stemm, R. Katz, S. Seshan, A network measurement architecture for adaptive applications, in: Proceedings of IEEE INFOCOM, Tel Aviv, Israel, 2000, pp. 285–294.
[27] J. Zhang, L. Cheng, I. Marsic, Models for non-intrusive estimation of wireless link bandwidth, in: Proceedings of PWC, Venice, Italy, 2003.
[28] B.-J. Kim, A network service providing wireless channel information for adaptive mobile applications: proposal, in: Proceedings of IEEE ICC, Helsinki, Finland, 2001, pp. 1345–1351.
[29] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Transactions on Information Theory 24 (5) (1978) 530–536.
[30] H. Schulzrinne, A. Rao, R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326, IETF, April 1998.
[31] E. Amir, S. McCanne, H. Zhang, An application level video gateway, in: Proceedings of ACM Multimedia'95, San Francisco, CA, 1995, pp. 255–265.
[32] Real Networks. Available from: <http://www.realnetworks.com/products/producer/features.html>.
[33] W. Tan, A. Zakhor, Real-time Internet video using error resilient scalable compression and TCP-friendly rate control, IEEE Transactions Multimedia 1 (2) (1999) 172–186.
[34] The TCP-friendly web site. Available from: <http://www.psc.edu/networking/tcp_friendly.html>.

Puneet Sharma received a Ph.D. in Computer Science from the University of Southern California, Los Angeles in 1998. Prior to that he earned a B.Tech. in Computer Science and Engineering from the Indian Institute of Technology, Delhi. Currently, he is a Senior Research Scientist at Hewlett–Packard Laboratories, Palo Alto, California. At HP labs he conducts research in Wireless and Mobile Networking, Overlay Network Services, Network Measurement and Monitoring. Email: puneet@hpl.hp.com.

Jack Brassil received the B.S. degree from the Polytechnic Institute of New York in 1981, the M.Eng. degree from Cornell University in 1982, and the Ph.D. degree from the University of California, San Diego, in 1991, all in electrical engineering.

He has been with Hewlett–Packard Laboratories since 1999. He currently is a Research Scientist and Program Manager in Princeton, NJ. Prior to that he managed a research team in Palo Alto, CA, investigating Internet streaming media, information hiding, and communication networks and protocols. Before joining HP he held multiple research positions at Bell Laboratories in Murray Hill and Holmdel, NJ. He is a Senior Member of the IEEE and a member of the IEEE Communications Society. Email: jtb@hpl.hp.com.

Sung-Ju Lee is a research scientist at the Mobile and Media Systems Lab (MMSL) of HP Labs. He received his Ph.D. in Computer Science from the University of California, Los Angeles. He published nearly 50 papers in the field of computer networks. He is currently an associate-editor-in-chief for ACM SIGMOBILE's Mobile Computing and Communications Review (MC2R) and serves on the editorial board of Elsevier Science's Ad Hoc Networks Journal. He was a co-guest editor of the Wireless Communications and Mobile Computing's special issue on Mobile Ad Hoc Networking, was a co-TPC chair for the first IEEE conference on Sensor and Ad Hoc Communications and Networks (SECON 2004), was a co-TPC chair for the first ACM workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH 2003; held in conjunction with MobiCom 2003), serves as a technical program committee and organizing committee member of various prestigious networking related conferences. He is also a steering committee member of IEEE SECON and ACM WMASH. He is a member of ACM, ACM SIGMOBILE, ACM SIGCOMM, IEEE, IEEE Communications Society, and IEEE Computer Society. His research interests include computer networks, mobile networking and computing, wireless LANs, ad hoc and mesh networks, overlay networks, media over networks, and large-scale service infrastructure networks. Email: sjlee@hpl.hp.com.

Kang G. Shin is the Kevin and Nancy O'Connor Professor of Computer Science and Founding Director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan.

His current research focuses on QoS-sensitive networking and computing as well as on embedded real-time OS, middleware and applications, all with emphasis on timeliness and dependability. He has supervised the completion of 53 Ph.D. theses, and authored/coauthored around 600 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He has co-authored (jointly with C.M. Krishna) a textbook "Real-Time Systems," McGraw Hill, 1997.

He has received a number of best paper awards, including the IEEE Communications Society William R. Bennett Prize Paper

Award in 2003, the Best Paper Award from the IWQoS in 2003, and an Outstanding IEEE Transactions of Automatic Control Paper Award in 1987. He has also coauthored papers with his students which received the Best Student Paper Awards from the 1996 IEEE Real-Time Technology and Application Symposium, and the 2000 UNSENIX Technical Conference. He has also received several institutional awards, including the Research Excellence Award in 1989, Outstanding Achievement Award in 1999, Service Excellence Award in 2000, Distinguished Faculty Achievement Award in 2001, and Stephen Attwood Award in 2004 from The University of Michigan; a Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002; and 2003 IEEE RTC Technical Achievement Award.

He received the B.S. degree in Electronics Engineering from Seoul National University, Seoul, Korea in 1970, and both the M.S. and Ph.D degrees in Electrical Engineering from Cornell University, Ithaca, New York in 1976 and 1978, respectively. From 1978 to 1982 he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the US Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, and International Computer Science Institute, Berkeley, CA, IBM T.J. Watson Research Center, Software Engineering Institute at Carnegie Mellon University, and HP Research Laboratories. He also chaired the Computer Science and Engineering Division, EECS Department, The University of Michigan for three years beginning January 1991.

He is Fellow of IEEE and ACM, and member of the Korean Academy of Engineering, is serving as the General Chair for the 3rd ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys'05), was the General Chair of the 2000 IEEE Real-Time Technology and Applications Symposium, the Program Chair of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chair of the 1987 RTSS, the Guest Editor of the 1987 August special issue of *IEEE Transactions on Computers* on Real-Time Systems, a Program Co-Chair for the 1992 *International Conference on Parallel Processing*, and served numerous technical program committees. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991–1993, was a Distinguished Visitor of the Computer Society of the IEEE, an Editor of *IEEE Transactions on Parallel and Distributed Computing*, and an Area Editor of *International Journal of Time-Critical Computing Systems*, *Computer Networks*, and *ACM Transactions on Embedded Systems*. Email: kgshin@eecs.umich.edu.