Distributed Channel Monitoring for Wireless Bandwidth Aggregation

Puneet Sharma[1], Sung-Ju Lee[1], Jack Brassil[1], and Kang G. Shin[2]

[1] Hewlett-Packard Laboratories, Palo Alto, CA
{puneet, sjlee, jtb}@hpl.hp.com
[2] University of Michigan, Ann Arbor, MI
kgshin@eecs.umich.edu

Abstract. Aggregating low-speed WAN links into a higher-speed logical link promises to improve data-transfer rates to collaborating communities of wireless mobile multi-homed 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. We present various architectural design alternatives for such a monitoring system, and evaluate them using 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.

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 LAN such as IEEE 802.11x, or accept nearly ubiquitous but much slower access using a WAN such as a 2.5G or later-generation cellular network.

Although networks that provide high-speed access to mobile users are currently under development (e.g., *EvDO*, 4G 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 [13]. A group of multi-homed 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), though we will refer to them simply as *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 [3, 16]. 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 [13] 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. 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. Our conclusions are drawn in the final section.

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. Figure 1 shows a system that can be readily deployed by a network access provider, 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

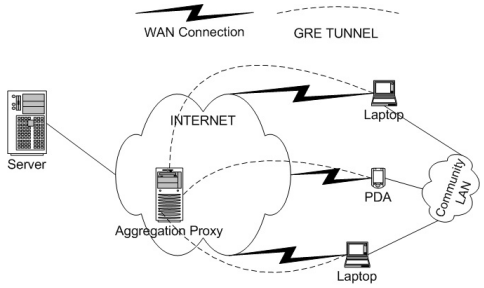

Fig. 1. A bandwidth aggregation service architecture.

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 [6] 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 are used for community and aggregated channel formation and management.

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. 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.

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 seconds, followed by a loss rate of 5% for 50 seconds, and then a loss rate of 10% for 50 seconds. 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 [13] seeks to map hierarchically layer-coded [9] 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 [13]. Figure 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 transi-

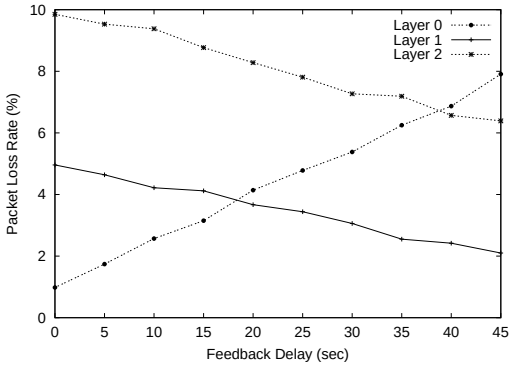

Fig. 2. The effect of reporting latency on aggregation performance.

tion event. In fact, when the feedback latency is larger than 18 seconds, 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

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

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 obviously important for a monitoring system to measure 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 answer key questions, 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 membership 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 ability to assess 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 the 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) either periodically, or upon detection of an event, or when a certain 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 Figure 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. An analysis of the proxy facing simple binary decision detailed in [12] illustrates this complexity.

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 [11]. Active probing schemes typically use `pathchar` to obtain link information [5]. The RTT of each hop is measured for variable packet sizes to calculate link bandwidth [8]. Packet pairing [4] 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 and IP probes is proposed in [1].

Passive measurement schemes such as SPAND [15] 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 [17]. 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 [7].

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).

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 as LAN bandwidth is plentiful, and report sizes are small. 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. [3] Messaging overhead will be limited, and actions described below will help avoid redundant information exchange. Finally, in practical settings we anticipate that the size of most collaborating communities will be small, perhaps tens of members.

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.

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

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

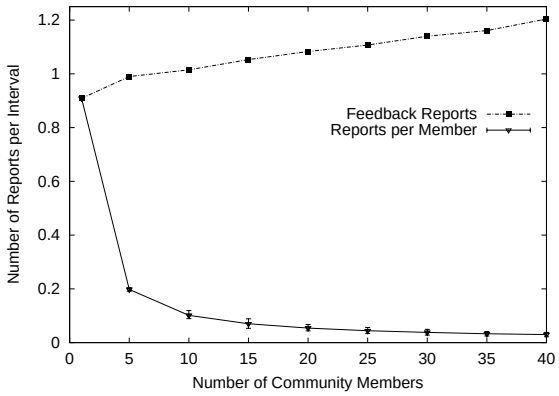

Fig. 3. The number of proxy reports issued per member per reporting interval I_p, and the average number of reports received by the proxy per reporting interval I_p.

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 problems associated with monitoring system configuration we used *ns-2* based simulation. We set the value of both reporting intervals I_l and I_p to 1 second. Figure 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 2 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. Figure 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.

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).

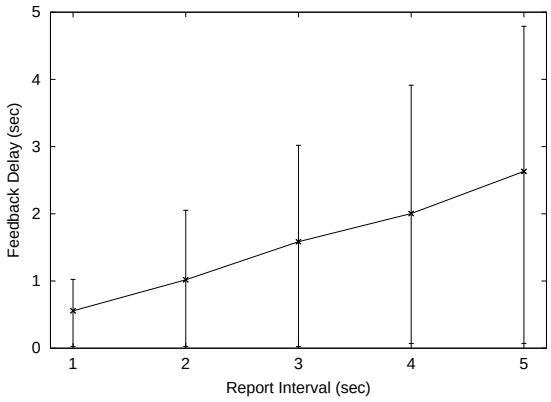

Fig. 4. The effect of reporting interval I_p on feedback latency.

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 second 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 Figure 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 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.

6 Conclusion

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. 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.

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 [2, 10, 14]. We are also exploring use of monitored information in a multifuction proxy to perform joint channel and traffic control [12].

References

1. Y. Breitbart, C.-Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz, "Efficiently monitoring bandwidth and latency in IP networks," in *Proceedings of IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 933–942.
2. C. Carter and R. Kravets, "User device cooperating to support resource aggregation," in *Proceedings of IEEE WMSCA*, Callicoon, NY, June 2002, pp. 59–69.
3. P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-performance, reliable secondary storage," *ACM Computing Surveys*, vol. 26, no. 2, pp. 145–185, June 1994.
4. C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proceedings of IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 905–914.
5. A. B. Downey, "Using pathchar to estimate Internet link characteristics," in *Proceedings of ACM SIGCOMM*, Cambridge, MA, Sept. 1999, pp. 241–250.
6. D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic routing encapsulation GRE," IETF, RFC 2784, Mar. 2000.
7. B.-J. Kim, "A network service providing wireless channel information for adaptive mobile applications: Proposal," in *Proceedings of IEEE ICC*, Helsinki, Finland, June 2001, pp. 1345–1351.
8. K. Lai and M. Baker, "Measuring link bandwidths using a deterministic model of packet delay," in *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000, pp. 283–294.
9. S. McCanne and M. Vetterli, "Joint source/channel coding for multicast packet video," in *Proceedings of IEEE ICIP*, Washington, DC, Oct. 1995, pp. 25–28.
10. M. Papadopouli and H. Schulzrinne, "Connection sharing in an ad hoc wireless network among collaborative hosts," in *Proceedings of NOSSDAV*, Florham Park, NJ, June 1999, pp. 169–185.
11. R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, Nov. 2003.
12. P. Sharma, J. Brassil, S.-J. Lee, and K. G. Shin, "Distributed channel monitoring for wireless bandwidth aggregation," HP Laboratories, Technical Report HPL-2003-171, Aug. 2003. [Online]. Available: http://www.hpl.hp.com/techreports/2003/HPL-2003-171.html
13. P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin, "Handheld routers: Intelligent bandwidth aggregation for mobile collaborating communities," HP Laboratories, Technical Report HPL-2003-37R1, May 2003. [Online]. Available: http://www.hpl.hp.com/techreports/2003/HPL-2003-37R1.html
14. A. C. Snoeren, "Adaptive inverse multiplexing for wide area wireless networks," in *Proceedings of IEEE GLOBECOM*, Rio de Janeiro, Brazil, Dec. 1999, pp. 1665–1672.
15. M. Stemm, R. Katz, and S. Seshan, "A network measurement architecture for adaptive applications," in *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000, pp. 285–294.
16. C. B. S. Traw and J. M. Smith, "Striping within the network subsystem," *IEEE Network*, vol. 9, no. 4, pp. 22–32, July/Aug. 1995.
17. J. Zhang, L. Cheng, and I. Marsic, "Models for non-intrusive estimation of wireless link bandwidth," in *Proceedings of PWC*, Venice, Italy, Sept. 2003.