Handheld Routers: Intelligent Bandwidth Aggregation for Mobile Collaborative Communities

Puneet Sharma,* Sung-Ju Lee,* Jack Brassil,* and Kang G. Shin†
*Mobile & Media Systems Lab, Hewlett-Packard Laboratories, Palo Alto, CA 94304
†Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor, MI 48109

Abstract

Multi-homed, mobile wireless computing and communication devices can spontaneously form communities to logically combine and share the bandwidth of each other's wide-area communication links using inverse multiplexing. *But membership in such a community can be highly dynamic, as devices and their associated WAN links randomly join and leave the community. We identify the issues and tradeoffs faced in designing a decentralized inverse multiplexing system in this challenging setting, and determine precisely how heterogeneous WAN links should be characterized, and when they should be added to, or deleted from, the shared pool. We then propose methods of choosing the appropriate channels on which to assign newly-arriving application flows. Using video traffic as a motivating example, we demonstrate how significant performance gains can be realized by adapting allocation of the shared WAN channels to specific application requirements. Our simulation and experimentation results show that collaborative bandwidth aggregation systems are, indeed, a practical and compelling means of achieving high-speed Internet access for groups of wireless computing devices beyond the reach of public or private access points.*

1 Introduction

An increasing number of multi-homed wireless mobile computing devices are being equipped with two distinct types of wireless communication interfaces: a local area network (LAN) interface such as IEEE 802.11x, and a wide area network (WAN) interface such as a 2.5G or later generation cellular link. The capabilities of these interfaces differ greatly, most notably with the available LAN bandwidth exceeding the WAN's bandwidth by one to three orders of magnitude. For the foreseeable future we anticipate that this bandwidth disparity between local and wide area wireless network connections will remain intact.

Public high-speed Internet connectivity from such devices is now typically achieved by connection via the wireless LAN interface to an access point which is connected to a high-speed, wired connection. It remains unlikely, however, that opportunistic deployment of these access points will ever realize ubiquitous — or even relatively geographically broad — access. Even where access points are densely deployed, seamless roaming between access points remains a technical challenge, and may not serve the business interests of either access point operators, venue owners or service providers. Further, even where access point coverage is rich, the transmission rate of the wired connection — typically 1.5 Mb/s — is limited and shared among a possibly large group of users, and unlikely to increase significantly in transmission speed in the foreseeable future.

To overcome the limited geographic coverage of public access points, we envision an alternative, complementary solution to high-speed Internet access through collaborative resource sharing. A group of wireless, mobile computing and communication devices in close proximity can dynamically form communities interconnected through their compatible high-speed LAN interfaces; we call these ad hoc groups *Mobile Collaborative Communities (MC^2)*. Each MC^2 member independently uses its WAN interface to create a communication *channel* to an inverse multiplexer, and optionally offers to other members (full or partial) access to this channel. The set of participating channels connecting the MC^2 members to the inverse multiplexer can be logically combined with an inverse multiplexing protocol to yield a higher-speed *aggregated channel* that is available from any one of the individual MC^2 members. The participating members acting as *handheld routers*, receive some of the packets destined to other members over their WAN links and forward them onto the LAN.

Due to end-device heterogeneity, mobility, and time-varying link transmission characteristics, the system we consider here is highly dynamic, and must be assembled, administered, and maintained in a decentralized fashion. We present the design of a collaborative bandwidth aggregation architecture that is both practical and readily de-

ployable. A key contribution we make is showing that significant performance gains can be realized by adapting shared WAN link selection to the specific application requirements of the communication flows. As an illustration, we demonstrate how the quality of a hierarchically-layered video stream transmitted over lossy channels can be improved by a priority/application-aware traffic assignment.

The rest of the paper is organized as follows. Section 2 explores the issues and tradeoffs faced in creating a decentralized inverse multiplexing system. Section 3 introduces algorithms for the assignment of application flows to heterogeneous WAN channels, and Section 4 describes the specific system architecture we chose to study. Performance evaluation results from an *ns*-based simulation are presented in Section 5, and Section 6 describes the implementation of a prototype system used to corroborate our findings. Related work is summarized in Section 7, and our conclusions are presented in the final section.

2 Issues, Challenges, and Approaches

The challenge of designing an effective inverse multiplexing system becomes harder when we recognize that the components are heterogeneous, imperfect, and supporting time-varying workloads. For example, WAN link transmission characteristics (i.e., bandwidth, packet latency, loss) will vary, possibly dramatically as end-devices move around. Links from different service providers may be of dissimilar technologies with different costs, complicating link selection. Links of the same type from a single network operator might have dependent or correlated transmission characteristics or outages.

The potentially large latencies introduced by packet forwarding through power- and processing-limited mobile computing devices is also a challenge. Disparities in the forwarding latency on different paths traversing heterogeneous computing devices with time-varying computing workloads can introduce packet misordering in the end-to-end path that can affect certain applications adversely. For example, non-interactive multimedia streaming applications will typically be lightly affected, though larger client buffer capacities might be desired. Although packet reordering might not reduce multimedia application performance noticeably, it can complicate TCP RTT computation and decrease TCP throughput.

2.1 Multiplexing Layer

A key issue in our overall system design is the identification of the preferred protocol layer for the multiplexing function. Since IP performs routing and multiplexing, it is natural to consider a network layer multiplexing implementation. An IP-based solution could be implemented exclusively at the communicating end-systems; in this case any packet scheduling, reordering, and reassembly would occur, as usual, only at the source and the destination. Though such a network layer implementation can be achieved in several ways, each requires end-system kernel modification, restricting the availability of channel aggregation to data transfers between modified end-systems. An additional disadvantage of network layer striping is that it could restrict the channel assignment policies (i.e., the intelligent mappings of flows to available channels) that we might seek to implement, since the network layer is generally not aware of application characteristics and requirements. Performing multiplexing at the network layer, however, does have the advantage that it would not require any changes to existing applications.

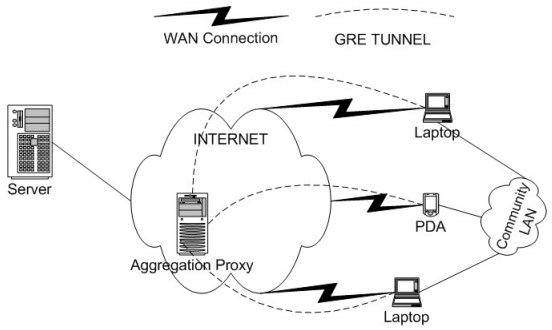

Figure 1. A bandwidth aggregation service architecture.

An alternative solution is to perform multiplexing at the transport layer. Once again, end-system protocol stacks would require modifications, though transport-layer channel assignment policies could potentially be made more easily aware of application requirements. The obvious deployment issues associated with either network- or transport-layer multiplexing suggest a role for solutions using application-layer multiplexing. Although such an implementation would incur more packet processing overhead, it requires no kernel modification and is easy to install, maintain and monitor. Application-layer multiplexing also permits controlling packet scheduling on a per-application, per-connection or per-packet priority basis. Like transport-layer and network-layer multiplexing, application-layer multiplexing is also transparent to the applications and does not require modifications to the applications themselves.

2.2 Forwarding Mechanism

What forwarding mechanism should an inverse multiplexer use to transmit a packet over a chosen channel? Irrespective of a packet's destination, different packets must traverse different routes. There are several means of achieving

this. One approach is to change each packet's destination address to the IP address of the appropriate MC^2 member's WAN interface. When a packet arrives at the MC^2, its destination address would be reverted back to the original MC^2 member destination address. This would, in a sense, be similar to providing a Network Address Translation (NAT) service, albeit in a distributed manner. But packet modification and processing overhead at the forwarding nodes associated with this approach might be significant.

Another packet forwarding approach could use *loose source routing* to forward a packet through the intermediary interfaces associated with the desired WAN channel to traverse. This would avoid the need to provide a special NAT-like packet forwarding service beyond ordinary IP routing itself. However, loose source routing has multiple, well-known weaknesses (e.g., use of IP options, extra router processing) as well as limited router support, making its use largely unworkable.

A preferred packet forwarding implementation would use *tunnels* between the inverse multiplexer and each MC^2 node. Tunneling has long been used to establish static paths, and most OS network stacks today have built-in support for tunnels. In such a system packet forwarding would operate as follows. Unicast packets sent from an Internet-connected source would be routed normally to the inverse multiplexer, where each would then be forwarded, according to the multiplexer's flow-to-channel assignment policy, to the tunnel corresponding to the appropriate WAN channel. Upon arrival at the MC^2 node, the packet would be decapsulated and forwarded on the wireless LAN to its intended destination. In this simple case, all upstream traffic would be sent over a single WAN link, typically — but not necessarily — the receiver's own. Figure 1 shows a bandwidth aggregation service architecture using Generic Routing Encapsulation (GRE) [3] tunnels.

2.3 Proxy Placement

Another key question in the design of our system is the appropriate placement of the inverse multiplexer in the end-to-end connection. In principle, this function can be located at almost any point between the WAN link terminations and the connection end-point (e.g., origin server), including the end-point itself. The preferred location depends on many factors including the type of WAN links, who is providing the aggregation service, whether collaborating devices agree to connect to a common multiplexing point, and how generally accessible the multiplexing service must be from a wide range of origin servers.

If all the WAN links from a MC^2 terminate at the same point, a preferred location for the inverse multiplexer is that termination point. It is natural to think of a *proxy* providing this service, and to ease our discussion, we will simply use this term to refer to the location of the inverse multiplexer, regardless of whether a distinct physical component is used to implement the function. If the proxy is located near the WAN link termination points, then it is likely easier and more efficient for a wide range of services to use the proxy to transfer data to the MC^2. In such a case the aggregation service can be provided as a value-added service by the MC^2 members' common Internet Service Provider (ISP). The proxy can alternatively be located at the network edge close to the origin server, or even at the origin server itself. While this location avoids the potential restriction of requiring a common WAN link termination point, MC^2 members might have to communicate with different aggregation services to communicate with different servers. As we will describe later, one can also envision an aggregation service being provided by a third party by placing proxy in the middle of the network.

2.4 Collaboration Incentives

Aggregated bandwidth channels can be realized only when hosts willingly collaborate by sharing their communication channels. Willingness to collaborate is not an issue for a single user with multiple mobile devices (e.g., cell phone, PDA, laptop, etc.) forming a 'community' (i.e., personal area network), nor might it be an issue for colleagues or acquaintances. But what are the incentives for collaboration between hosts owned by multiple parties with little or no pre-existing relationship? Clearly, if many community members seek access to the same content (e.g., multicast video) then the members will be well motivated to take advantage of faster download or streaming. But if each host seeks to receive unique content, will the community members each be willing to sacrifice their computing and communications resources?

We have reason to be optimistic that collaborating communities will occur spontaneously in public settings. The benefits of aggregated channels are high, particularly when members seek to access large files or high-quality media objects. In some cases no viable alternative means of access will exist. In addition, the 'cost' of the resources being offered (e.g., bandwidth) is relatively low, members can experiment at low risk by offering partial resources, and offered resources may be reclaimed with little difficulty. The recent success of peer-to-peer file sharing leads us to believe that device owners may be willing to share communication and computational resources as readily as they do information, particularly if they directly and immediately benefit from resource sharing.

Table 1. Categorization of channel allocation.

	Application-aware	Application-agnostic
Channel adaptive	Layer Priority Striping[a]	WRR, WFQ
Channel non-adaptive	Not applicable[b]	Random, Round-robin

3 Channel Allocation and Packet Striping

For each active flow a proxy is responsible for two tasks. First, the proxy must select a set of channels on which to forward packets to the MC^2 destination. Second, the proxy must intelligently stripe arriving packets across those channels. Efficient channel allocation and striping algorithms map or remap the flows to the channels based on both application requirements and the number and the condition of available channels. Hence, the algorithms we examine in this section are both *application-aware* and *channel-adaptive*. As an example, the algorithms we consider would seek to assign a flow from an audio or video source to channels that would maintain that application's stringent delay or delay jitter requirements, while assigning bulk data transfer (e.g., FTP) flows to channels that might incur longer delays but are reliable. Of course, both the number and condition of assignable channels might vary over a given flow's lifetime. Channel allocation and striping algorithms can be categorized along the following orthogonal dimensions:

- **Channel-adaptive**: These algorithms assign packets on different channels according to the channel conditions such as bandwidth, loss, and delay. For example, a Weighted Round Robin (WRR) algorithm stripes packets to channels in proportion to each channel's available bandwidth.
- **Application-aware**: Striping algorithms can also use knowledge or a *profile* of an application flow and its end-system requirements for channel selection and packet striping. Since applications can have different profiles, each application would potentially need a different algorithm. These algorithms promise to provide better performance than application-agnostic algorithms, but they have the burden of obtaining information about a flow's requirements. This information can be obtained explicitly from the traffic source, or may be inferred by examining the flow itself, or some combination of both. For instance, a source might mark its packets (e.g., ToS field in the IP header) or a proxy might infer application type from destination information (e.g., TCP or UDP port numbers) or even the application payload.

A given striping algorithm can be both channel-adaptive and application-aware, as summarized in Table 1. [1]

[1](a) We propose layer-priority striping for hierarchically-layered videos in Section 3.4. (b) Application-aware algorithms are application-specific and also require channel information.

3.1 Application Characteristics

Each application flow can be described by itself (intra-characterization) or against other application flows (inter-characterization). Examples of the former include Multiple Description video Coding (MDC) [1] and the imprecise computation model [5] that is widely used in the real-time computing community. That is, an application flow has multiple representations or versions expressing different degrees of satisfaction (being minimally-to-fully satisfactory). The proxy must allocate and schedule resources to at least guarantee the minimum degree of satisfaction for each given application flow. That is, timely delivery of the base layer or essential part of each application flow must be guaranteed, and the enhancement layer or the optional part receives lower priority. For more general types of applications (including video), an application flow itself is also characterized by its minimum packet interarrival time, burstiness, multiple QoS levels, bandwidth, loss rate, delay, and jitter requirements.

On the other hand, the inter-characterization deals with relative importance among different applications, rendering their priority order. In general, it is more "beneficial" to give more important application flows priority over less important ones in scheduling their data transmission or allocating bandwidth.

3.2 Channel Characteristics

The number and condition of channels between the proxy and MC^2 can change with time due to many factors including interchannel interference, and communication failures due to MC^2 members' departures, device mobility, or power depletion. While a proxy must be continuously aware of channel conditions, it does not have the benefit of observing packet reception or MC^2 member behavior directly. The proper design of a monitoring system providing such feedback is crucial to achieving superior system performance, and we have investigated it in detail in a separate paper [9].

We assume that our bandwidth aggregation system has a two-sided channel monitor (i.e., one side on the MC^2 and the other side at the proxy) that is jointly responsible for detecting membership changes, "sensing" channel characteristics (e.g., bandwidth, error rate, latency, security, reliability, cost, etc.) and ensuring that the proxy has reasonably current channel information. Further, to facilitate WAN channel monitoring and overall system reliability and responsiveness, the MC^2-based agents are distributed, operating on many or all of the member devices [9].

The proxy is thus capable of ordering channels in its resource pool according to the application requirements of arriving flows. For example, channels can be sorted accord-

ing to their delay and reliability characteristics, and then the proxy may choose the n most reliable channels for transporting the base layer (or essential part) of a video flow while choosing less reliable channels for the enhancement layer.

3.3 Allocation/Reallocation of Channels

Each application flow f_i; $1 \leq i \leq k$, is assumed to have been demultiplexed into an ordered (according to the application characteristics) set of $n_{f_i} \geq 1$ subflows $\{sf_j : j = 1, \ldots, n_{f_i}\}$. The traffic of each subflow sf_j is represented by either a simple token bucket model (ρ_j, σ_j) or a linear bounded arrival process $(p_j, s_j^{max}, b_j^{max})$, where

ρ_j:	average token drain rate,
σ_j:	bucket size,
p_j:	minimum or average time separation between two consecutive packets,
s_j^{max}:	maximum packet size (in bytes),
b_j^{max}:	maximum burst size (in bytes) for subflow j.

Let $C = \{ch_\ell : \ell = 1, \ldots, n_c\}$ be an ordered (according to their condition) set of channels available. Note that the size and ordering of this set changes with time and will be updated by the monitor. The problem is now to select one or more channels from C on which to assign each subflow j. This selection must also be adapted to reflect the changing number and condition of available channels.

We want to map a demultiplexed application flow $f_i = \{sf_j^i : j = 1, \ldots, n_{f_i}\}$ to a changing set of channels $C = \{ch_\ell : \ell = 1, \ldots, n_c\}$. Recall that the subflows of f_i are ordered according to their importance to the application, while the channels are ordered according to their relevance to the application requirements. For example, $f_v = \{sf_1^v, sf_2^v\}$ and $C = \{ch_1, ch_2, ch_3\}$ where sf_1^v and sf_2^v represent the base and enhancement layers of a video stream f_v, respectively, and ch_i's are ordered according to their reliability or their signal-to-noise ratio values. In this case sf_1^v may be transported via ch_1 and ch_2, and sf_2^v via ch_3, assuming that the former requires two channels while the latter requires only one channel.

In general, as many topmost (say, k) channels as necessary for transporting sf_1^i are assigned first to sf_1^i, and then repeat the same procedure with the remaining channels for sf_2^i, and so on. If sf_1^i does not need the entire bandwidth of channel ch_k, the remaining bandwidth of this channel is assigned to sf_2^i, and ch_k will transmit the packets of sf_1^i and sf_2^i using a Weighted Round-Robin (WRR) scheduling algorithm where the weights between the two subflows are determined based on the ch_k's bandwidth assigned to sf_1^i and sf_2^i. Also, if there is not enough bandwidth available, the least important subflows are not transported at all, realizing a form of imprecise computation [5]. The more general case of multiple application flows is described in [8].

3.4 Example: Assignment of Video Flows

The potential benefit of application-aware channel assignment is best illustrated by considering the case of video traffic. First, a high-quality video flow might be of sufficiently high bandwidth that it could not be transmitted over a single WAN channel. Second, link transmission characteristics can directly affect the perceived quality of the transmission. We present three 'strawman' algorithms, based on simple heuristics, for striping video packets.

- **Layer-Priority Striping (LPS)**: This algorithm can be used for video streams that are hierarchically layer-coded. This encoding process generates a base layer ℓ_0 containing information required for decoding, and one or more optional enhancement layers ($\ell_i : i = 1, \ldots, n$) in a hierarchical structure of cumulative layers. The reconstruction is progressive (i.e., enhancement layer ℓ_k can only be used if all sublayers $\ell_i : i = 0, \ldots, k-1$ are available). Thus, the layer index i corresponds to the layer priority.

  The LPS algorithm matches the layer-priority to the channel reliability as described in Section 3.3. For instance, the base layer ℓ_0 is assigned to the most reliable channels, where the channel loss rate is used as the metric for reliability. The packets for each layer are striped in WRR fashion onto the allocated channels. If a new channel with higher reliability becomes available, allocation of layers is shifted up to channels with higher reliability. Similarly, if the channel with the highest reliability becomes unavailable, the allocation is shifted down.
- **Frame-Priority Striping (FPS)**: This algorithm can be used for MPEG video traffic. The MPEG video stream is separated into three subflows (sf_I, sf_P, sf_B) based on frame types. The priority order for the frames in MPEG Group of Pictures (GoP) is I>P>B. Similar to the LPS algorithm, the channels are allocated according to the subflow priority. The I-frame subflow (sf_I) is sent over the most reliable channels, and so on.
- **Independent-Path Striping (IPS)**: This algorithm is well suited to multiple state video coding [1], where a stream is encoded into multiple *independently* decodeable subflows. Moreover, information from one subflow can be used to correct the errors in another subflow. Hence, it is important for a receiver to successfully receive as many complete subflows or components as possible, and it is desirable to achieve a low correlation of loss across different subflows.

  The IPS algorithm tries to achieve path diversity by allocating a separate channel for each description. Since the

video can be reconstructed (albeit at lower quality) even if one or more entire subflows are lost, video reception is protected against one or more complete channel failure(s).

4 Architecture

Considering the many systems issues identified in Section 2, we chose a channel-aggregation architecture that is both simple and scalable. Figure 1 shows the proposed architecture which permits deployment by various types of network transport and service providers, including content owners, Internet access providers, wireless telecommunication service providers, or content distribution network operators.

The system architecture has three principal components: a dedicated appliance providing channel-aggregation proxy services, standard LAN-based announcement and discovery protocols, and standard protocol tunnels. The dedicated aggregation proxy performs inverse multiplexing at the application layer.

Generic Routing Encapsulation (GRE) [3] tunnels are used to create channels between the proxy and participating MC^2 members, and support packet forwarding. This approach requires no modification to MC^2 members, as most contemporary OSs (e.g., Linux, FreeBSD, Windows) have built-in support for GRE tunnels. Each packet received by a MC^2 member over a GRE tunnel is automatically decapsulated and forwarded via the wireless LAN to the destination device. Since the destination is oblivious to which MC^2 node forwarded the data packets, no additional data reassembly functionality is required at the receiver.

To participate in MC^2 formation and channel aggregation, a standard announcement and discovery protocol is required on end-devices. The choice of a standard protocol enables end-devices to participate in other types of resource or service discovery and access. Though the specifics of these protocols are beyond the scope of this paper, Jini, Universal Plug and Play (UPnP), and the Service Location Protocol (SLP) may all be suitable candidates.

5 Performance Evaluation: Simulation

We evaluated the proposed bandwidth aggregation system using the *ns-2* simulator. Figure 2 shows the network topology we used for simulating an entire end-to-end system. The number of MC^2 members was varied from 2 to 14, and the MC^2 members were interconnected via an 11 Mb/s wireless LAN. In our experiments with homogeneous WAN links, the link bandwidth was set at 115.2 kb/s, roughly consistent with currently-available 2.5G cellular services. With the exception of the single dedicated receiver, each MC^2 member was equipped with both

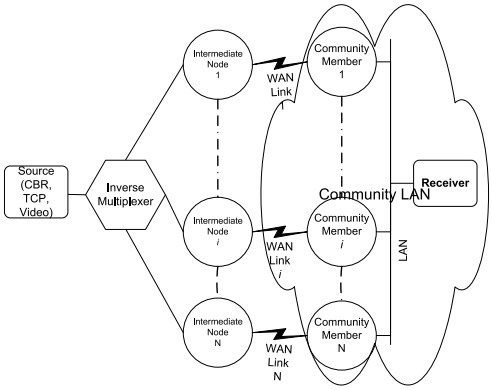

Figure 2. Simulation topology.

a WAN and a LAN interface. The receiver could communicate upstream only using one of the other members as a gateway. We consider a variety of scenarios with varying link characteristics such as bandwidth, loss, and membership dynamics. We first evaluate the benefits of bandwidth aggregation for different applications: we use (1) bulk file transfer over TCP and measure TCP throughput, and (2) CBR traffic over UDP and measure packet loss rate. We then study how much performance improvements application-aware striping can make using layered video as an example application. For experiments with TCP and UDP traffic we implemented three application-agnostic striping algorithms: random, round-robin (RR), and weighted round-robin (WRR).[2] We implemented the LPS algorithm described in Section 3.4 for application-aware, channel-adaptive striping algorithms. Due to space limit, we show only a subset of the results from [8].

5.1 TCP Throughput

We first evaluate the effect of the addition or deletion of a WAN link in an aggregated channel on TCP throughput. Let's consider the simple case of a fixed membership MC^2. We measured TCP throughput by transferring a 1 MB file from a data source to a MC^2 receiver using $2 \sim 14$ identically-configured links aggregated into the shared pool. To provide a baseline for measured TCP throughput, we also performed the experiment with a single channel (i.e., no aggregation).

Figure 3 plots the measured TCP throughput as the MC^2 size changes. The average throughput achieved with a single link was 103.2 kb/s. As expected, the TCP throughput increases nearly linearly as the number of links grows under both RR and WRR policies until saturation occurs with six links. This saturation occurs due to the limit imposed by the receiver's maximum window. As the number of available

[2] To be precise, since packets are not fragmented in the proxy we have implemented the Surplus Round Robin approximation of bit-WRR.

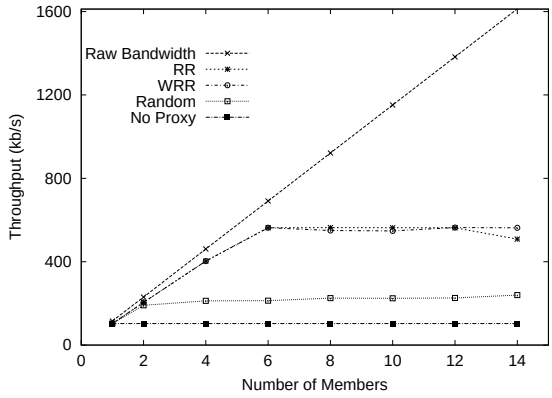

Figure 3. TCP throughput as a function of MC^2 size.

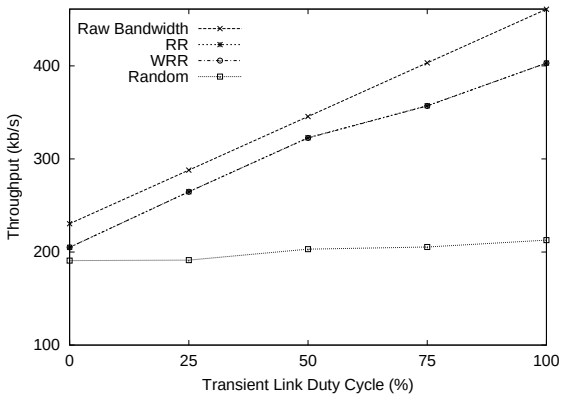

Figure 4. TCP throughput with two persistent links and two transient links.

channels increases, the bandwidth-delay product increases, but TCP cannot utilize all the available bandwidth because of the small receiver window. The TCP throughput continues to increase linearly if the receiver-advertised window is increased to accommodate a larger bandwidth-delay product. The random policy does not perform as well as (W)RR because it causes undesired side effects, such as packet reordering and unstable RTT calculation, thus reducing the TCP throughput.

We next explore TCP performance for the highly-dynamic case where the channels were frequently added or removed from the pool. In this scenario, two links always remain active in the pool and two links periodically join the pool simultaneously for *up-time* and leave for *down-time*. The sum of *up-time* and *down-time* was kept constant at 20 seconds. That is, an *up-time* of 20 seconds is same as striping continually over four links (i.e., 100% duty cycle) and a *down-time* of 20 seconds is the same as continually striping over only two links (i.e., 0% duty cycle). In this set of experiments there was no significant difference in the achieved throughput for RR and WRR striping. Hence it is difficult to distinguish between the two in the figures presented here. Figure 4 shows that as the duty cycle increases, the average TCP throughput increases for RR and WRR, whereas the random striping cannot effectively utilize the available bandwidth of the transient link. Even though two of the links in the pool are rather short-lived, channel-adaptive striping is able to utilize their capacity to improve the transfer rate.

Table 2. CBR loss rate (%) as a function of MC^2 size.

# of members	Random	RR	WRR	No proxy
2	75.15	75.15	75.15	87.57
4	50.31	50.3	50.32	87.57
6	25.48	25.45	25.5	87.57
8	1.14	0.61	0.59	87.57
10 or more	0	0	0	87.57

5.2 CBR Media Traffic over UDP

Many media applications generate CBR traffic carried over UDP. We studied the loss observed for an 8×115 kb/s = 920 kb/s CBR stream from a video source to a MC^2 destination. The topology used for this set of experiments was the same as the one for the TCP throughput experiments.

Table 2 shows the packet loss rate as a function of the MC^2 size. Without channel aggregation we observe 87.6% loss as the CBR stream rate was eight times the bandwidth of a single link. As more links are pooled, the loss rate decreases.

We also studied the performance of different striping algorithms for UDP streaming over four heterogeneous links of 128 kb/s, 64 kb/s, 32 kb/s, and 16 kb/s, respectively. Table 3 shows the loss rates when a CBR stream of 256 kb/s is sent over the aggregated channel. Random and RR algorithms do not adapt to channel bandwidth and allocate an equal number of packets to each channel. Hence, the lower bandwidth links drop larger amounts of traffic, resulting in higher total loss rates. In contrast, WRR achieves a low overall loss rate by assigning packets proportionally to the bandwidths of various links and distributing the loss uniformly over different links.

Table 3. CBR loss rate (%) over four heterogeneous links.

	Random	RR	WRR
Link 1 (128 kb/s)	0	0	14.1
Link 2 (64 kb/s)	6.93	7.9	13.75
Link 3 (32 kb/s)	53.67	53.95	13.06
Link 4 (16 kb/s)	77.15	76.97	11.54
Total	34.18	34.4	13.25

Table 4. Loss rate (%) with extra available channels.

Protocols	Random	RR	WRR	LPS
Loss rate	5.56	5.58	5.68	1.41

We also evaluated how CBR streaming over UDP is affected by the dynamics of MC^2 membership. Under the same join and leave dynamics as for the TCP throughput experiments, the loss rate decreased with the increase in duty cycle.

5.3 Application-Aware Striping

We now present the results from the application-aware striping experiments. We experimented with the application-aware, channel-adaptive *LPS* algorithm introduced in Section 3.4. The scenarios were so chosen as to elucidate the key benefits of application-aware mechanisms in comparison with application-agnostic schemes.

5.3.1 Availability of Extra Channels

Let's consider a scenario where the proxy has ten channels available for striping data. All the channels are identical except for having different error rates that vary from 1 to 10%. The error rate e_i for channel ch_i was set at i%. The traffic source generated CBR traffic at 30 kb/s and the bandwidth of each channel was 20 kb/s. Thus, at least two channels are required for the transfer. Table 4 shows the average loss rates for the different striping algorithms. If the proxy is unaware of the application profile/requirements, then it will use all the available channels indiscriminately. Hence, the observed loss rate is higher for the application-agnostic striping algorithms. But the proxy using an application-aware algorithm achieves better performance by striping data over only the two channels with minimum loss. Hence, even minimal information, such as the bandwidth requirements of the application, can make a significant improvement in the system performance.

Table 5. Loss rate (%) for layered video with static channels.

	Application-agnostic			Application-aware
	Random	RR	WRR	LPS
Layer ℓ_0	5.07	9.97	6.05	1
Layer ℓ_1	5.28	1.02	4.89	4.96
Layer ℓ_2	5.53	4.81	5.16	9.72

Table 6. Loss rate (%) for layered video in limited static channels.

	Application-agnostic			Application-aware
	Random	RR	WRR	LPS
Layer ℓ_0	18.99	23.57	18.76	0.96
Layer ℓ_1	19.64	12.44	20.53	5.15
Layer ℓ_2	19.89	22.4	19.25	100

5.3.2 Priority-awareness

We now present the results for striping a hierarchically-layered video stream with a base layer ℓ_0 and two enhancement layers ℓ_1 and ℓ_2. Each layer was modeled as a 15 kb/s CBR stream. The topology consists of three MC^2 members, each with a 20 kb/s WAN link. The error rate on the channels was 1, 5 and 10%, respectively. Table 5 shows the percentage loss rate suffered by each layer. As expected, the random striping indiscriminately distributes the loss over all the layers. Since all the layers are constant bit-rate with equal bandwidth and the number of channels is same as the number of layers, the RR algorithm stripes all the packets from one layer to one channel. Instead of the loss being spread over all the layers equally, the layer sent over the most unreliable link suffers the most loss. The loss rate for the base layer is significantly less with the LPS algorithm. LPS uses priority-awareness to assign the base layer to the most reliable link, and the highest enhancement layer to the link with the highest error rate.

The striping algorithms utilize application-awareness to intelligently drop lower-priority subflows when an insufficient amount of resource is available. To demonstrate this benefit of application, we simulated a scenario with two MC^2 members connected to the Internet via 20 kb/s WAN link. The error rate of the channels was 1 and 5%, respectively. Note that the offered traffic rate exceeds the aggregated channel bandwidth. Table 6 shows the loss experienced by different layers while streaming the same video traffic as described above. Since the two available channels cannot handle the offered load of all the three video layers, the LPS algorithm drops the layer ℓ_2 entirely, improving the loss suffered by the base layer ℓ_0 and the enhancement layer ℓ_1. The application-agnostic mechanisms end up spreading the loss over all the layers.

Table 7. Loss rate (%) for layered video in limited dynamic channels.

	Application-agnostic			Application-aware
	Random	RR	WRR	LPS
Layer ℓ_0	18.88	22.9	18.9	1.01
Layer ℓ_1	19.73	12.78	20.75	4.97
Layer ℓ_2	19.96	22.76	18.88	100

5.3.3 Dynamic Channel Adaptation

What happens if in the above scenarios the link error rates change dynamically? We also simulated the limited channel scenario described earlier with varying channel error rates. Each link has an error rate of 1% for 50 seconds and then 10% for 50 seconds, repeating this cycle several times during the lifetime of the flow. The changes in error rates are distributed such that at any instant one link has error rate of 1% and the other 10%. Thus, the total error rate is the same throughout the experiment. Table 7 shows the measured loss for each layer. With application-agnostic schemes, lack of application knowledge leads to uniform loss rates for all the layers of the flow. In contrast, LPS entirely drops the enhancement layer ℓ_2 due to limited channel availability, to shield layers ℓ_0 and ℓ_1 from loss. Also, it remaps the base layer to the more reliable channel as the channel error rates change. Hence, the loss suffered by the base layer is lower.

6 Implementation, Experiments and Results

We now present a detailed description of the channel aggregation testbed we built and the experiments we performed. The principal goals of the testbed were to validate our proposed architecture, corroborate our simulation results, and explore deployment issues that might not readily emerge from our simulations. Again, readers are refered to [8] for the entire results.

6.1 Testbed Implementation

Figure 5 shows a block diagram of the prototype channel aggregation system we constructed, with dark arrows representing control messages and light arrows representing data traffic. Each MC^2 member runs a compact *Client Connection Manager* (CCM) application. The CCM participates in the announcement and discovery of MC^2 members (and their associated WAN links). The CCM communicates the addition or deletion of links to the *Server Connection Manager* (SCM) which resides on the proxy and maintains the channel resource pool. The CCM also monitors link transmission characteristics such as bandwidth and delay that is provided to the striping proxy.

We implemented a Linux-based inverse multiplexing proxy. The proxy intercepts each packet destined for a

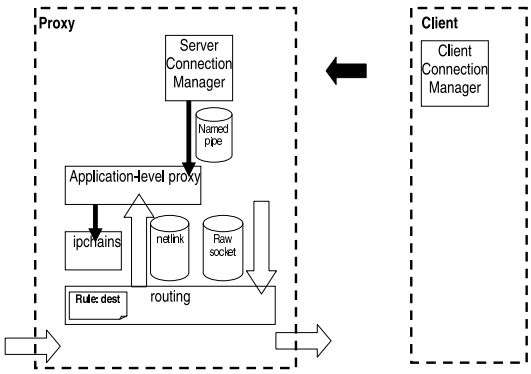

Figure 5. Linux-based implementation of an aggregation proxy.

MC^2 and forwards it to the GRE tunnels corresponding to each active channel. Packet interception at the proxy is handled by *Netfilter*, a packet filtering subsystem in Linux that is primarily used for building firewalls and NATs. For each channel aggregate, the proxy sets up Netfilter's forwarding rules to intercept appropriate data traffic and passes it to the proxy's user-layer forwarding engine. The forwarding engine currently implements both random and round-robin data striping policies.

Data reassembly at the receiving side is automatic and straightforward. Packet forwarding is enabled at each MC^2 node sharing a WAN link. When a packet is received by a node over a GRE tunnel, it is decapsulated and passed to the node's routing engine. Since the destination address of the decapsulated packet corresponds to the receiver's LAN address, the packet is forwarded to the LAN.

We emulated an entire end-to-end system with the topology similar to our simulation topology in earlier section. Notebook computers running Linux (2.2.16) kernel, each with built-in support for GRE tunnels, were used to as MC^2 members. The MC^2 members were connected to each other via a 10 Mb/s Ethernet. WAN links were emulated by connecting a wired serial null modem running PPP to the *NISTnet* network emulator whose transmission link characteristics we could control. As in simulations presented in Section 5, the transmission speed of each serial link was set at 115.2 kb/s. Each MC^2 member, with the exception of the dedicated data receiver, had both an emulated WAN interface and an Ethernet interface.

Traffic generation, collection and measurement was performed using NetIQ's *Chariot* network measurement tool version 4.2. Chariot *end-points* running on the data source and receiver generated various packet flows, emulating reli-

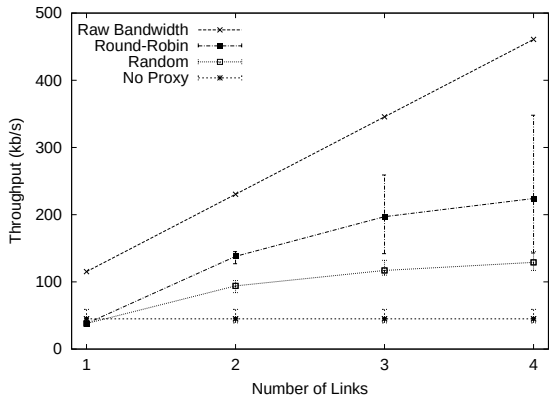

Figure 6. Effect of MC^2 size on TCP throughput.

able data transfers, streams, etc.

6.2 Experimental Results

6.2.1 TCP Throughput

To validate our simulation results in practice, we measured TCP throughput by transferring a 1MB file from a data source to a MC^2 receiver using two to four identically-configured, aggregated links. To provide a baseline for measured TCP throughput we also performed the experiment with a single channel (i.e., no aggregation) both with and without the proxy in the data path. Performance was measured using both round-robin and random striping policies. Figure 6 plots the measured TCP throughput as the number of links in the aggregate pool changes, with error bars showing the minimum and maximum measured throughput among the 50 trials.

The average TCP throughput achieved with no proxy was 45 kb/s. The TCP throughput with a single link and the proxy in the data path is 38 kb/s, not significantly lower than the throughput achieved without a proxy, indicating that the proxy does not introduce a long delay. The TCP throughput measured in the testbed was lower than the simulation results due to PPP overhead and the presence of background traffic. However, the trends with respect to the number of MC^2 members were similar in both the cases.

6.2.2 Streaming Media via UDP

We next conducted experiments to study how high bandwidth streaming is enabled with channel aggregation. In these experiments a server streams a stored media file to the receiver at one of various bit rates (64, 128, 175, and 256 kb/s). Chariot generates a traffic pattern intended to resemble the video transmission of Cisco's IP-TV. RTP is used as the stream transport protocol. Each experiment was repeated 25 times, measuring the loss rate and RTP delay jitter observed by the receiver.

Without channel aggregation the receiver can only receive a stream with negligible loss at the 64 kb/s rate. Higher bit-rate streams suffered more than 70% loss, and due to this high loss rate, the tests were prematurely terminated by Chariot. Note that the limited available bandwidth precludes use of retransmission for loss recovery. Techniques such as Forward Error Correction (FEC) cannot be used in this setting, especially for low-bandwidth links, as it further increases the bandwidth required. Such a high loss rate can severely degrade the perceived stream reception quality, making it unwatchable. Yet striping over just two links reduced the loss rate dramatically for the 128 kb/s stream; every 128 kb/s stream test completed with a loss rate of less than 0.1%. Dynamic link striping was able to sustain a 175kb/s stream over three links and a 256kb/s stream over four links with negligible loss rate less than 0.2%. This result confirms that bandwidth aggregation enables high bandwidth multimedia streams to be delivered to MC^2s, which would otherwise be impossible.

It was also observed that the system generates relatively little jitter. In most cases, the jitter is less than 10 ms with the maximum jitter occasionally exceeding 20 ms. Such small amounts of jitter can be easily absorbed by the receiver buffer in multimedia applications and will have negligible effect on the viewing experience of the video receiver.

7 Related Work

Adaptive inverse multiplexing for CDPD wireless networks is explored in [10]. In this scheme the 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 fragmented packets are then tunneled over multiple links. In this case the endpoints of the WAN connections forming the virtual link are the same.

The bandwidth of mobile users with multiple interfaces is aggregated at the transport layer in pTCP (parallel TCP) [4]. 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. The striping is performed by pTCP and is based on congestion window size of each TCP-v connection. When congestion occurs on a certain pipe, pTCP performs data reallocation to another pipe with large congestion window. One possible problem of this approach is that the congestion window size may not accu-

rately reflect the bandwidth-delay product.

Coordinating communications from *multiple* mobile computing devices has become a new focus of interest. Network connection sharing has been proposed in [6]. This architecture permits use of a single, idle WAN connection among collaborating mobile devices but it does not address aggregation of multiple links into a high capacity bundle.

Our goal of cooperation and resource aggregation among collaborating devices is similar to the vision of the mobile grouped devices (MOPED) architecture [2]. The goal of MOPED project is to enable group mobility such that a user's set of personal devices appear as a single mobile entity connected to the Internet. The MOPED routing architecture builds a new *multipath* layer to encapsulate packets between the home agent and MOPED devices, unlike our approach of using GRE tunnels. MOPED architecture adapts the Mobile IP home agent to support aggregation of multiple links at network and transport layers. In MOPED, modifications to both client and server kernels are also required. Our application-level approach does not require any kernel changes and allows support for different application profiles.

The commuter Mobile Access Router (MAR) project [7] also leverages wireless WAN connection diversity to provide high speed Internet access to mobile users. Instead of using the WAN connections of the users, it relies on pre-provisioning the MAR with different WAN connections, limiting the aggregation to the already exiting links.

8 Conclusion

We have designed, implemented and evaluated a deployable bandwidth aggregation system providing high-speed Internet access to a collaborating community of wireless end-systems. We have demonstrated that the system not only improves access service quality, but enables otherwise unachievable services such as the delivery of high-bandwidth streaming media. Further, we have shown that network and application-aware allocation and assignment policies do indeed improve system performance.

Though not described in this paper, we performed various experiments with bandwidth-adaptive multimedia applications over aggregated connections. Ideally, such applications would measure available bandwidth and smoothly increase or decrease audio or video quality to optimize perceived reception quality. We typically observed an application decreasing its streaming rate to a predefined fraction of its maximum rate; often this rate was well below the available bandwidth of the aggregated connection. The application would subsequently maintain that low rate, remaining non-responsive to any increase in available bandwidth, no matter how large it is. Since the widely-used applications we tested were proprietary, we were unable to modify their adaptation algorithms.

To aggregate bandwidth we have relied upon the conventional technique of inverse multiplexing. But rarely, if ever, has inverse multiplexing been applied in such a dynamic and decentralized setting and made to work. As a result of operating in this challenging environment, we have identified a significant number of technical issues that appear to be fertile areas for future research. Some of these include WAN cost sharing, accounting, information privacy, and security. We have also relied on the assumption that an application's networking requirements are well-known, though such a characterization remains a formidable and long standing problem. Not surprisingly, we have found that aggregating relatively homogeneous communication links is often easier and more successful than working with heterogeneous links. Opportunity lies in the development of simple 'rules-of-thumb' that help identify which links are sufficiently homogeneous that aggregation is likely to perform well.

References

[1] J. Apostolopoulos. Error-resilient video compression via multiple state streams. In *Proceedings of VLBV*, pages 168–171, Kyoto, Japan, Oct. 1999.

[2] C. Carter and R. Kravets. User device cooperating to support resource aggregation. In *Proceedings of IEEE WMSCA*, pages 59–69, Callicoon, NY, June 2002.

[3] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic routing encapsulation GRE. RFC 2784, IETF, Mar. 2000.

[4] H.-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidth on mutli-homed mobile hosts. In *Proceedings of ACM MobiCom*, pages 83–94, Atlanta, GA, Sept. 2002.

[5] J. W. S. Liu, K.-J. Lin, W. K. Shih, R. Bettati, and J. Chung. Imprecise computations. *Proc. IEEE*, 82(1):1–12, Jan. 1991.

[6] M. Papadopouli and H. Schulzrinne. Connection sharing in an ad hoc wireless network among collaborative hosts. In *Proceedings of NOSSDAV*, pages 169–185, Florham Park, NJ, June 1999.

[7] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. MAR: A commuter router infrastructure for the mobile internet. In *Proc. of ACM/USENIX MobiSys*, Boston, MA, June 2000.

[8] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin. Handheld routers: Intelligent bandwidth aggregation for mobile collaborative communities. Technical Report HPL-2003-37R1, HP Labs, May 2003.

[9] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin. Distributed channel monitoring for wireless bandwidth aggregation. In *Proceeding of the IFIP-TC6 Networking 2004*, pages 345–356, Athens, Greece, May 2004.

[10] A. C. Snoeren. Adaptive inverse multiplexing for wide area wireless networks. In *Proceedings of IEEE GLOBECOM*, pages 1665–1672, Rio de Janeiro, Brazil, Dec. 1999.