Distributed Communication Paradigm for Wireless Community Networks

Puneet Sharma Sung-Ju Lee Jack Brassil
Mobile & Media Systems Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94304
Email: {puneet,sjlee, jtb}@hpl.hp.com

Kang G. Shin
Dept. of Electrical Engineering & Computer Science
University of Michigan
Ann Arbor, MI 48109
Email: kgshin@eecs.umich.edu

***Abstract*— Distributed computing has been widely embraced as a cost-effective means of performing compute-intensive tasks by pooling the computational resources of collaborating systems. We envision the emergence of an analogous approach to *communication* resource sharing which we call *distributed communication*. Distributed communication enables sharing a set of relatively low-speed WAN channels emanating from communities of multi-homed devices interconnected with a high-speed wireless LAN. We envisage opportunities to aggregate cellular links from spontaneously formed ad hoc groups of mobile devices, as well as broadband access links (e.g., DSL) from neighboring residences. But, will individuals be willing to share bandwidth as easily as they share bits? A prototype system that we have constructed convinces us that the technical challenges of distributed communication can be overcome. And there appears to be no other means of satisfying the growing demand for access bandwidth as quickly and as cheaply.**

I. Introduction

Computing users are demanding ever more resource-intensive services requiring high-speed Internet access, including delivery of high-quality streaming media to the home. Further, there is growing demand for ubiquitous access, as consumers seek to engage in collaborative activities such as multi-player gaming from mobile and wireless handheld devices. Yet despite tremendous growth in raw networking bandwidth within enterprises and core networks, access link speeds remain modest and appear unlikely to increase significantly soon.

In the case of personal mobile wireless computing devices, multiple wireless communication interfaces have become more prevalent. Often these devices have two distinct types of interfaces: a high-speed LAN interface and a relatively low-speed WAN interface. The wireless LAN interface (e.g., IEEE 802.11x) is primarily used to connect to access points that have a wired connection to the Internet. The WAN interface, such as a 2.5G or later generation cellular link, offer near ubiquitous Internet connectivity when a device is out-of-range of an access point. There exists, however, a great disparity between the bandwidths of the interfaces. For example, an 11 Mb/s WiFi connection has approximately 100 times the typical transmission bandwidth of the Sony Ericsson CG-82 EDGE PC card operating with AT&T's EDGE (Enhanced Data for Global Evolution) service on their GSM/GPRS network.

Yet we claim that higher-speed communication can be achieved even when devices are out-of-range of a wireless LAN access point. Ad hoc or *mobile collaborative communities (MC^2)* can be formed by a group of mobile computing devices in close proximity connected through their compatible high-speed LAN interfaces. By pooling *independent* communication resources, the community can aggregate the bandwidths of all members' WAN links to achieve higher-speed connectivity to all members.

A similar form of communications resource sharing can be envisioned to support broadband residential access. Today Internet access speeds remain limited by both high broadband access tariffs and enforced rate limits. But high-speed wireless LANs are increasingly prevalent in homes, and the geographic reach of these LANs can be readily engineered to reach hot spots (and their associated wired access links) in nearby residences.

In this paper we explore an emerging form of communication link resource sharing we call *distributed communication*. We draw both inspiration and pause from the success of computing resource sharing as developed in the distributed computing community, where resources have long been aggregated to perform computation-intensive tasks that would have otherwise been unwieldy. Resource sharing has already been shown to be of critical importance in mobile computing and networking environments, where both computation and communication resources are often scarce.

We show that a decompose-deliver-aggregate paradigm not only promises to improve communication efficiency and performance but also promises to enable new applications. We argue that a combination of technical advances and economic constraints will result in this type of resource sharing in a wide and growing variety of networking environments. The principal *technical* challenge in creating such systems is that the shared communication resources are geographically distributed (e.g., WAN links on separate hosts) and must be coordinated in a decentralized fashion. The principal *social* question is whether computing users will embrace forming collectives to share bandwidth in the same way as they readily share 'content.'

The rest of the paper is organized as follows: Section II introduces and defines the emerging distributed communica-

tion paradigm. In Section III we explore the issues and challenges of assembling, coordinating, managing, and monitoring communication resources spread across multiple, collaborating hosts We then present our bandwidth aggregation system in Section IV. Additional technical and social challenges for widespread adoption of distributed communication are discussed in Section V. The paper ends with concluding remarks in Section VI.

II. Distributed Communication Paradigm

Distributed computing has primarily focused on pooling the computing resources of a collaborative system for faster computation. Recently, distributed computing has been employed to leverage the idle computing power of several thousands of Internet-connected computers to solve large scientific problems, such as the Search for Extraterrestrial Intelligence (SETI@home) [1] and the Great Internet Mersenne Prime Search (GIMPS) [2]. Protein sequencing, breaking encryption codes, and 3-D film rendering are other examples of problems being attacked with distributed computing. Each task is decomposed into smaller subtasks that are mapped and then executed on a set of collaborating nodes. Upon completion of subtasks, their (partial) results are merged together for the complete result. In an analogous fashion, distributed communication is an emerging paradigm where (i) communication resources (e.g., channels) are pooled, (ii) networking tasks (e.g., layered video streams) are decomposed into subtasks (e.g., component layers), and (iii) these subtasks are mapped to different channels across multiple hosts with communicated components integrated after delivery. In this section we compare and contrast distributed communication with distributed computing.

Low-bandwidth communication channels have long been aggregated to create a virtual high-bandwidth link. But the application of this conventional approach has been largely confined to a single administrative and ownership domain. Striping data across multiple, parallel communication channels has been used to improve system performance or reliability in relatively statically-configured disk storage systems [3] and in fixed, wired LAN–WAN interconnection systems [4]. Inverse multiplexing is used for link-layer aggregation in the Bandwidth on Demand Interoperability Group (BONDING) [5] and in Inverse Multiplexing for ATM (IMA) specifications [6]. In these approaches, each of the WAN connections being "bundled" together originates from — and terminates at — a *common* end-point.

Communication resource pooling has also been applied to mobile, multi-homed wireless devices [7]–[10]. Once again, the links being bundled emanate from a single device, and hence are confined to a single administrative domain.

But the emergence of very high-speed wireless LANs now permits sharing WAN links across multiple end-systems, and frequently across administrative and ownership boundaries. These lines have already been crossed by the distributed computing community.

The primary reason for not adopting truly distributed communication has been a lack of sufficient bandwidth for the aggregation phase. Such limitation does not exist in the emerging mobile collaborative communities as described earlier. Similar to distributed computing, the distributed communication paradigm has three stages: (i) task decomposition and mapping, (ii) subtask execution, and (iii) subtask integration.

The communication tasks can be primarily categorized in two types:

- *Real-time* tasks with timeliness requirement (e.g., streaming).
- *Non-real-time* tasks without any timeliness requirement (e.g., bulk data transfer).

Decomposition of real-time and non-real-time communication tasks into subtasks for distribution must be handled differently. The non-real-time communication tasks are similar to discrete computation tasks where the partial results from the subtasks are integrated when the subtasks are completed. But real-time communication subtasks need to continuously deliver received data to the end-point for timely reassembly. Thus, the decomposition techniques used in distributed computation cannot be directly applied to distributed real-time communication.

Another difference between distributed computing and communication environments is the type of shared resources, and the rate at which those resources can change. Computing resources offered in a distributed computing environment are relatively reliable, though the instantaneous CPU load might change rapidly. But wireless communication channels can be highly unreliable, and may also suffer rapidly changing congestion conditions. Clearly, the mapping and remapping of communication subtasks in a distributed communication setting differs from task assignment in a distributed computing setting.

According to Amdahl's Law [11] for parallel computing the speedup is limited by the fraction of computation task that is parallelizable. The speedup is further reduced due to overheads associated with interprocessor communication, load imbalance and additional computation. Theoretically, the communication tasks are completely parallelizable using bit-striping. Still, it is not possible to achieve ideal aggregate throughput due to protocol issues such as packet reordering on heterogeneous links, etc.

The decompose-deliver-aggregate paradigm of distributed communication not only improves the performance for faster downloads for large files but also enables new applications such as high-quality streaming which might not be possible over a single low-speed WAN link.

Distributed communication community members would each enjoy higher-speed, statistically-multiplexed WAN access, a service often far more desirable than private, but lower-speed access. Other possible applications include receipt of large data-sets quickly using swarming techniques such as *Bittorrent* or via simultaneous file transfers using parallel FTP or GridFTP. Similarly, in case of layered video streaming with each layer assigned to a different communication channel can

deliver streaming media to heterogeneous end-systems both reliably and efficiently [12].

III. Issues and Approaches in Realization of Distributed Communication

We next discuss the main issues and challenges we face in realizing distributed communication systems and present several approaches to address these challenges.

A. Application Decomposition and Integration

In general, a communication application must be decomposed and assigned based on (i) the availability of members and their resources, and (ii) the application's unique features/requirements. More formally, an application $\mathcal{A}$ is decomposed into a set of sub-applications $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n$ which are then assigned to a set of members $\{m_1, m_2, \cdots, m_\ell\}$ by a *task mapping agent*. Note that $n = \ell$ does not hold in general, as ℓ and the amount and availability of each member's resource change with time.

Depending on (i) the number and condition of available channels and (ii) the nature of sub-applications, the mapping agent can make intelligent decisions on allocation/reallocation of channels to the sub-applications [13]. For example, the I-frames of an MPEG-coded video stream may be transmitted via the most reliable subset of available shared channels, while P- and B- frames can be transmitted via less reliable channels, or even omitted if there are not enough channels available.

B. Execution Requirements and Environment

After decomposition, the communication subtasks are executed on the respective nodes they are mapped to. These participating nodes may be asked to dedicate not only communication resources but computing and storage resources also on the behalf of the community. In certain cases, the assigned task may be as simple as packet forwarding and might not require specific task setup at the node. But in some cases the assigned subtasks might be considerably more involved, consuming nontrivial resources. To participate in the parallel download of a large file, a node would typically be asked to execute a partial file transfer over its WAN interface and then over its LAN interface. Thus, the assignment of an appropriate (or optimal) subtask to a given node would vary based on the overall application requirement, the feasibility and performance of decomposition and mapping, and the node's available resources.

Ideally, a participating node should provide an efficient and safe environment for the execution of a subtask. The execution environment will determine how to create and perform the subtasks. These issues regarding the execution environment are similar to those seen in active networks [14]. While packet forwarding is supported in most operating systems, complex subtasks might require active execution environments such as JanOS [15], CANEs (Composable Active Network Elements) [16], and ASP EE [17].

C. Communication Dynamics

Both the availability and the quality of shared communication channels can be expected to vary with time. Yet these dynamics must be monitored and communicated to the task mapper to provide proper task assignments [18]. Community membership itself will change as hosts join and leave the community, due to either end-system failures (e.g., power loss) 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.

How do we monitor these communication dynamics? One possible approach is to deploy a *monitoring agent* within each community to oversee the communication environment.

The main advantage of deploying a monitoring agent here is the ease of detecting changes to community membership quickly. The drawback, however, is that if the agent resides on a single node it is difficult to monitor the WAN channel performance of other nodes in the community. Moreover, relying on a single (or even a few) monitor(s) can result in both a performance and reliability bottleneck.

This problem can be solved by either replicating the monitoring agent or making every member a monitoring agent, i.e., distributed monitoring. Distributed monitoring works as follows: Each member broadcasts its channel characteristics and associated information (e.g., communication costs and its energy balance) either 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 task mapper can obtain a copy of the database in two ways. First, the task mapper can request 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 task mapper has received. For example, an inquiry might be directed to a member with ample advertised available processing power, residual energy, or network bandwidth. An inquiry might be issued periodically, or be driven by an event such as the need to remap channels for a newly-arriving flow. The second way in which a task mapper can obtain 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).

An alternative approach is to perform monitoring at the location where the task mapping is done. When transport protocols that require end-to-end feedback (e.g., TCP) are used, end-hosts can piggyback their measured channel condition and membership information (e.g., will move out, will turn the device off because of low battery power, etc.) on the acknowledgment/feedback packets. Out-of-band signaling may be utilized when transport protocols without end-to-end

handshaking such as UDP are running. Although this approach has the advantage of easily monitoring communication channel state, there may be delay in obtaining the information from every member. To make matters worse, the feedback messages may not reach the other end, due to wireless transmission errors or unidirectional/asymmetric links. Hence, the right preference would be to use a combination of the two approaches.

D. Collaboration Incentives

Distributed communication requires users to collaborate by sharing their communication channels. But what are the incentives for users to collaborate? In the case where many users form a community to simultaneously access the same content (i.e., multicast), participating community members will presumably be well-motivated to take advantage of distributed communication. But what if only one host or a small fraction of members is receiving content at others' expense? Will community members be willing to sacrifice their bandwidth to permit other members to achieve statistical multiplexing gains?

A somewhat related debate is underway with regard to forwarding incentives in ad hoc network routing [19], [20]. In ad hoc networks, when the destination node is outside the radio transmission range of the source node, the communication endpoints rely on intermediate nodes on the path to forward the packets for them. The source and destination nodes will, in turn, forward data packets when they become the intermediate nodes for other communication pairs. Game-theoretic arguments have been advanced to show that collaboration by all participating nodes will result in maximum network throughput. Some researchers suggest use of credit-based, or reputation-based schemes to stimulate cooperation [19]. Forwarding in ad hoc networks, however, is somewhat different from the bandwidth sharing we consider. In ad hoc networks, nodes *rely* on each other to communicate amongst themselves.

In a distributed communications setting, nodes rely on each other not for basic connectivity, but for *improved* performance. As we will see in Section IV, a node completely controls access to its shared communication resources, and can revoke access if its communication needs are not being met by other community members. Ultimately, it is the ability to opt-in to achieve better performance, and the ability to opt-out when necessary, that leads us to conclude that link sharing is viable. Nonetheless, communities are more likely to form within domains where a pre-existing trust (or cost sharing) relationship exists. For example, an individual with multiple devices (e.g., cell phone, PDA, laptop) interconnected with a personal area network can benefit from resource sharing, as can small teams of people working together.

E. Privacy, Security, Fairness and Other Issues

Distributing communications comes at the cost of surrendering one's resources for community consumption. It also incurs the risk of exposing one's communications to eavesdropping. While passing sensitive information across shared channels should be avoided, encrypting data would be a recommended practice. In addition, interleaving communications across multiple users would provide some protection against collusive eavesdropping.

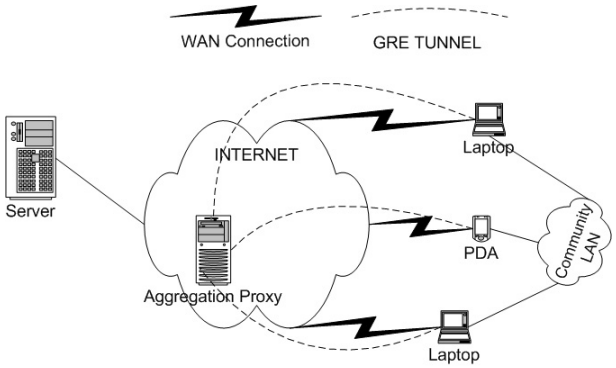

Fig. 1. A bandwidth aggregation service architecture.

Participants in a link sharing group also face the lowering of the barrier of one's personal system to malicious attacks. Clearly, participants must determine what communication subtasks are suitable for execution in their environments. Certain subtasks such as forwarding packets between interfaces using software already installed on a system, arguably pose less of a threat than supporting other more sophisticated and untested subtasks.

Users of file sharing services tend to recognize that such a system may serve a common goal but provide no assurance of 'fairness' in any sense. Users of link sharing may approach such systems in a similar spirit. Indeed, even the notion of a 'system' is not well-defined. While our discussion has focused on a single node participating in one community, nothing precludes a node from participating in multiple disjoint or overlapping communities simultaneously.

IV. Bandwidth Aggregation for MC^2

To explore an instance of a distributed communication system, we constructed a prototype bandwidth aggregation system for *mobile collaborative communities* (MC^2), shown in Figure 1. This prototype has enabled us to investigate many of the technical challenges and tradeoffs described so far. In this section we present this proof-of-concept, identify our design decisions, and describe the system's overall capabilities and limitations.

A. Design Choices

1) Multiplexing Layer: A key issue in the 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 only at the source and at 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 does have the advantage of not requiring any changes to existing applications.

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.

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 is reverted back to the original member destination address. But packet modification and processing overhead at the forwarding nodes associated with this approach would be prohibitive.

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 several, well-known weaknesses (e.g., use of IP options, extra router processing) as well as limited router support, making it 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 operating system 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.

3) Proxy Placement: Another key system design question 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, 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 the MC^2 terminate at the same point, a preferred location for the inverse multiplexer is that termination point.

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. The proxy can also be located at the network edge close to 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 subscribe to different aggregation services to communicate with different servers.

B. A Proxy-based 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 propose 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, striping packets across available links to the community. Generic Routing Encapsulation (GRE) [21] tunnels create channels between the proxy and participating MC^2 members, and support packet forwarding. 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 of which member 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) [22] are relied upon for community and aggregated channel formation and management.

C. Simulation and Testbed Experiments

We studied the performance benefits of a proxy-based bandwidth aggregation system through both simulation and construction of a testbed. We performed various experiments, including the transmission of high-quality video streams via UDP over multiple links with reassembly at a single destination node. These experiments successfully demonstrated that some services that simply could not be handled by the bandwidth of a single connection [13], could be delivered over aggregated connections.

Using the *ns-2* simulator we also measured TCP throughput when transferring a 1MB file from a data source to a receiver using 2$\sim$14 identically-configured links aggregated into the

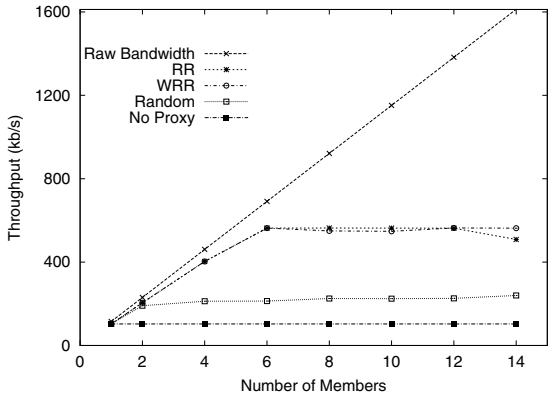

Fig. 2. Simulation result of TCP throughput.

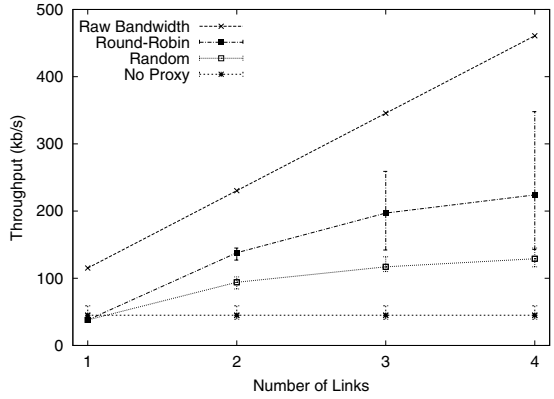

Fig. 3. Testbed result of TCP throughput.

TABLE I

CBR LOSS RATE (%).

# 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

shared pool. We implemented three striping algorithms: random, round-robin (RR), and weighted round-robin (WRR). To provide a baseline for measured TCP throughput, we also performed the experiment with a single channel (i.e., no aggregation).

Figure 2 plots the measured TCP throughput as the MC^2 size changes. The average throughput achieved with a single link was 103.2kb/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 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.

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. Table I shows the packet loss rate as a function of the number of the community members. Without channel aggregation we observe 87.6% loss because the CBR stream rate was eight times the bandwidth of a single link. As more links are pooled, the loss rate decreases.

We also conducted experiments on a Linux-based testbed to validate our architecture and explore deployment issues that might not readily emerge from our simulations. We measured TCP throughput by transferring a 1MB file from a data source to an MC^2 receiver using two to four identically-configured, aggregated links. Similar to the simulation study, we also performed the experiment with a single channel both with and without the aggregation proxy in the data path. Performance was measured using both round-robin and random striping policies. Figure 3 plots the measured TCP throughput as the number of links in the aggregate bundle changes, with error bars showing the minimum and maximum measured throughput among the 50 trials.

The average TCP throughput achieved with no proxy was 45kb/s. The TCP throughput with a single link and the proxy in the data path is 38kb/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 cases. In summary, results from both simulation and our testbed validate that the performance gains associated with practical instantiation of distributed communication are both significant and obtainable.

V. FURTHER CHALLENGES

The design of an effective inverse multiplexing system becomes very challenging when the component links are heterogeneous, imperfect, and support time-varying workloads. Wireless WAN links can be particularly treacherous to use effectively; their 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 communication characteristics and costs, complicating link selection and task assignment. And even what appear to be similar links offered by the same access provider may introduce thorny problems such as 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 major 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, dramatically decreasing aggregate communication rates. For example, non-interactive multimedia streaming applications will typically be lightly affected, though larger client buffer capacities may be desired. Although packet reordering might not reduce multimedia application performance noticeably, it can complicate TCP RTT computation and decrease TCP throughput.

In our investigations we have also observed that certain applications that do not rely on TCP have trouble coping with aggregated channels. For example, several of the most popular streaming media servers use proprietary congestion control algorithms that reduce bit rates in the face of congestion. These control systems typically do a poor job of responding to increases in available bandwidth, and thus failing to capture the benefit of aggregated channels.

The possibility of widespread communication resource sharing might stir the interest of behavioral economists. It is immediately obvious that users will face disincentives to link sharing. Access providers will presumably prohibit carriage of data from users who are not their direct subscribers. Though possibly not anticipated in current legal agreements between access providers and their clients, it is likely that those agreements will be updated to discourage or disallow the practice. This raises the intriguing technical question as to whether a service provider can detect the presence of data traffic forwarded by, but not originating from, a subscriber.

A decline in the price of higher-speed access would also discourage cooperative resource sharing. Barring unforeseeable regulatory and communication infrastructure investment incentives, the risk of such a pleasant outcome seems low. There is scant historical evidence that a windfall awaits access bandwidth consumers. At the same time, we can reasonably expect that technological advances leading to increased aggregate access bandwidth will permit access providers to increase revenues by growing their subscriber base rather than offering more bandwidth to their existing base.

VI. Conclusion

The bandwidth mismatch between high speed wireless LANs and lower speed access networks has set the stage for access link sharing across administrative and ownership domains. We have made a case for distributed communications and its applications to aggregating cellular links and broadband access links (e.g., DSL). The prototype bandwidth aggregation system we have constructed has demonstrated that capturing statistical multiplexing gains is technically feasible. Link aggregation can realize new, highly desirable services such as high-quality streaming media delivery or file sharing (e.g., video downloads) that would otherwise be difficult or costly to achieve.

We believe that an emerging distributed communication paradigm will gain increasing importance with the growing disparity in price and transmission speeds of wireless LANs and Internet access technologies [23]. While there remain many technical challenges in the realization of these systems, they can be solved by the collective effort of the networking systems research community. However, will the users forming collectives to share bandwidth perceive the potential performance gains to be worth the risk? Perhaps the greatest poser is how many users will reply in affirmative to the question: *Buddy, can you spare a baud?*

References

[1] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky, "SETI@home-massively distributed computing for SETI," *IEEE Comput. Sci. Eng.*, vol. 3, no. 1, pp. 78–83, Jan. 2001.

[2] The great internet mersenne prime search (GIMPS). [Online]. Available: http://www.mersenne.org/prime.htm

[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. B. S. Traw and J. M. Smith, "Striping within the network subsystem," *IEEE Network*, vol. 9, no. 4, pp. 22–32, July/Aug. 1995.

[5] BONDING Consortium, "Interoperability requirements for $n \times 56/64$ kb/s calls, version 1.0," Sept. 1992.

[6] ATM Forum, "Inverse multiplexing for ATM specification, version 1.0," July 1997.

[7] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidth on mutli-homed mobile hosts," in *Proc. of ACM MobiCom*, Atlanta, GA, Sept. 2002, pp. 83–94.

[8] A. C. Snoeren, "Adaptive inverse multiplexing for wide area wireless networks," in *Proc. of IEEE GLOBECOM*, Rio de Janeiro, Brazil, Dec. 1999, pp. 1665–1672.

[9] R. Kravets, C. Carter, and L. Magalhaes, "A cooperative approach to user mobility," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 5, pp. 57–69, Oct. 2001.

[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] G. M. Amdahl, "Validity of single-processor approach to achieving large-scale computing capability," in *Proc. of AFIPS Conference*, Reston, VA, 1967, pp. 483–485.

[12] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. of ACM SIGCOMM*, Stanford, CA, Aug. 1996, pp. 117–130.

[13] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin, "Handheld routers: Intelligent bandwidth aggregation for mobile collaborative communities," in *Proc. of IEEE BroadNets 2004*, San Jose, CA, Oct. 2004, pp. 537–547.

[14] K. L. Calvert, S. Bhattacharjee, E. W. Zegura, and J. Sterbenz, "Directions in active networks," *IEEE Commun. Mag.*, vol. 36, no. 10, pp. 72–78, Oct. 1998.

[15] P. Tullmann, M. Hibler, and J. Lepreau, "Janos: A java-oriented OS for active network nodes," *IEEE J. Select. Areas Commun.*, vol. 19, no. 3, pp. 501–510, Mar. 2001.

[16] S. Bhattacharjee, K. Calvert, Y. Chae, S. Merugu, M. Sanders, and E. Zegura, "CANEs: An execution environment for composable services," in *Proc. of DARPA Active Networks Conference and Exposition (DANCE)*, San Francisco, CA, May 2002, pp. 255–273.

[17] R. Braden, B. Lindell, S. Berson, and T. Faber, "The ASP EE: An active network execution environment," in *Proc. of DARPA Active Networks Conference and Exposition (DANCE)*, San Francisco, CA, May 2002, pp. 238–254.

[18] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin, "Distributed channel monitoring for wireless bandwidth aggregation," in *Proc. of IFIP-TC6 Networking Conference 2004*, Athens, Greece, May 2004, pp. 345–356.

[19] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks)," in *Proc. of ACM MobiHoc*, June 2002, pp. 226–236.

[20] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring, "Modelling incentives for collaboration in mobile ad hoc networks," in *Proc. of WiOpt*, Sophia-Antipolis, France, Mar. 2003.

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

[22] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service location protocol, version 2," IETF, RFC 2608, June 1999.

[23] J. Gray, "Distributed computing economics," Microsoft Research, Technical Report MSR-TR-2003-24, Mar. 2003.