Networking Going Postal

Nitin Garg* Sumeet Sobti* Junwen Lai* Fengzhou Zheng* Kai Li*
Arvind Krishnamurthy† Randolph Y. Wang*

Abstract

Making high-bandwidth Internet access pervasively available to a large world-wide audience is a difficult challenge, especially in many developing regions. As we wait for the uncertain takeoff of technologies that promise to improve the situation, we propose to explore an approach that is potentially more easily realizable: the use of digital storage media transported by the postal system as a general digital communication mechanism. We shall call such a system a *Postmanet*. Compared to more conventional wide-area connectivity options, the Postmanet has several important advantages, including wide global reach, great bandwidth potential, low cost, and ease of incremental adoption. While the idea of sending digital content via the postal system is not a new one, none of the existing attempts have turned the postal system into a *generic* and *transparent* communication channel that not only can cater to a wide array of applications, but also effectively manage the many idiosyncrasies associated with using the postal system. In the proposed Postmanet, we see two recurring themes at many different levels of the system. One is the simultaneous exploitation of the Internet and the postal system so we can combine their latency and bandwidth advantages. The other is the exploitation of the abundant capacity and bandwidth of the Postmanet to improve its latency, cost, and reliability.

1 Introduction

Making high-bandwidth wide-area Internet access pervasively available to a large world-wide audience is a daunting challenge. This is especially true in the vast under-developed regions of the world. Instead of waiting for the uncertain takeoff of a number of existing and proposed technologies, which can be many years away, a recent position paper [15, 23, 24] proposes to turn the *existing* world-wide postal systems into a generic digital communication mechanism as digital storage media is transported through the postal "network." The proposed system is dubbed the *Postmanet*.

1.1 Postmanet Advantages

Compared to more conventional wide-area connectivity technologies, the Postmanet enjoys several important advantages.

• *Wide reach.* The postal system is a truly global "network" that reaches a far greater percentage of the world's human population. To leverage the postal system for digital communication, one needs no significant new investment in exotic equipment.

• *Great bandwidth potential.* While the bandwidth potential of a "sneaker net" is well known, some may consider it to be a temporary fluke stemming from the relatively poor capacity of today's Internet. We, however, believe that this is not necessarily the case, if we examine some fundamental technology trends. Storage density of flash memory and magnetic disks has been increasing at the annual rate between 60% and 100%, and it is likely to continue in the foreseeable future. This tremendous rate of improvement is likely to be almost directly translatable to the amount of bytes transportable by the postal system for a fixed cost or in a fixed volume. Besides flash memory and hard disks, the next generation Blu-Ray DVDs can hold up to 27 GB per disc today. Hitachi Research has recently announced multi-layer technologies that can produce 150 GB discs by 2007 and 1 TB discs shortly thereafter. One can also ship multiple units of these storage devices. As better storage devices become available, they can be instantaneously and incrementally translated into Postmanet bandwidth improvements.

In contrast, the wide-area network bandwidth growth is constrained by labor-intensive and costly factors such as how quickly we can dig ditches to bury fibers in the ground, how quickly we can furnish last-mile wiring to homes (an endeavor that can be prohibitively expensive), how quickly we can launch satellites, or how quickly we can erect WiMax (the longer-distance versions of WiFi) towers. These factors are unlikely to improve faster than the exponential growth rate of storage density. Satellite- and WiMax-based solutions may face aggregate bandwidth limitations. And the future of some of these alternatives (such as WiMax) is far from certain. Far from being a temporary fluke, the bandwidth gap between Postmanet and more conventional alternatives is likely here to stay and, indeed, widen. We do not, however, necessarily view Postmanet as a competitor to these other alternatives. Before better alternatives become a widely deployed reality, exploring the Postmanet, an alternative that can already deliver practically infinite bandwidth today, may foster the development of and demand for sophisticated bandwidth-intensive applications, which may one day readily migrate onto alternative connectivity technologies.

• *Low cost.* The low cost advantage of the Postmanet should be attractive to average households, content offerers, and "power users" alike. The goal of providing citizens with affordable access to postal service is typically an integral part of most nations' postal system charters. In the U.S., even if each household sends (and receives) one DVD each day,

*Department of Computer Science, Princeton University, {nitin, sobti, lai, zheng, li, rywang}@cs.princeton.edu.
†Department of Computer Science, Yale University, arvind@cs.yale.edu.

the monthly cost of about $10 compares favorably with existing ISP offerings, especially if we were to consider its vast bandwidth potential. The relatively liberal use of the postal system by AOL and Netflix highlights the low cost advantage of this approach to content offerers. The availability of a public transit system-like Postmanet infrastructure, which allows each household to receive (per postman visit) a single disk that contains customized content from multiple content offerers, can further reduce the cost to all involved. In addition to catering to "low end" users, the cost advantage of the postal system relative to that of a high-speed wide-area network also holds for corporate "power users" shipping large amounts of data [8].

• *Good scalability.* The postal system appears to have tried and tested experience dealing with "flash crowds" such as those seen on tax days or certain holidays.

• *Ease of incremental adoption.* A single pair of Postmanet users can already derive useful value from the system, without having to wait for a massive-scale user community or world-wide infrastructure to develop. From this modest start, the system can grow gradually. This incremental deployment may circumvent the classic "chicken-and-egg" problem associated with the difficulty of simultaneously developing infrastructures, applications, and user populations.

1.2 Goals

The goal of exploring the Postmanet approach is *not* to compete against existing or future alternative network access modes; instead, the goal is to extend, to complement, and to even foster other alternatives.

• *Extending the Internet.* For those who have no access to connectivity or high-bandwidth connectivity, the Postmanet can provide an inexpensive connectivity alternative to enable certain networked applications, especially bandwidth-intensive ones.

• *Complementing the Internet.* The Postmanet has long (but reasonably predictable) latencies. We call such a channel a High Latency High Bandwidth (HLHB) channel. Correspondingly, we call a traditional Internet connection a Low Latency Low Bandwidth (LLLB) channel. For places that have access to both an HLHB channel and an LLLB channel, an interesting problem is how to exploit an integrated and simultaneous use of *both* channels to get the best of both worlds. For example, small requests, acknowledgements, "NAKs," and control messages may be sent along the LLLB Internet, while large messages are staged on mobile storage devices for transmission by the HLHB postal system. Another example of the complementary nature of the Postmanet is that it may increase the availability of the communication subsystem: if the Internet is down for some reason, one still has another alternative. In the rest of this paper, unless explicitly noted, we assume the simultaneous availability of an LLLB link and we examine ways of exploiting it.

• *Foster application development.* The Postmanet is likely to be more quickly realizable compared to more ambitious efforts of making high-bandwidth connectivity widely available. Bandwidth-intensive applications developed for the Postmanet, users who become accustomed to its benefits, and lessons learned can potentially be transferred to the alternatives farther away on the horizon when they become real.

The remainder of the paper is organized as follows. Section 2 discusses example applications, the potential pitfalls of developing ad hoc application-specific solutions, and the importance of general and transparent systems support. Section 3 explores how data on movable storage media is "routed" from its source to its destination. We examine options ranging from those that can provide a good level of service quality by employing data re-copying centers embedded inside the postal system, to "peer-to-peer" disk-forwarding schemes that can be incrementally adopted by end users without relying on an expensive infrastructure. We consider routing algorithms that must account for unconventional routing metrics, such as minimizing the number of movable storage media that any one site needs to handle. In Section 4, we turn our attention to the communication end points. We examine the challenges and opportunities that are unique in the Postmanet environment, how a carefully designed API can address these issues, and our prototype implementation of the API and example applications. We shall see two recurring themes at many different levels of the system. One is the simultaneous exploitation of the Internet and the postal system so we can combine their latency and bandwidth advantages. The other is the exploitation of the abundant capacity and bandwidth of the Postmanet to improve its latency, cost, and reliability. In Section 5, we present simulation results of the various Postmanet routing algorithms. In Section 6, we present measurement results of our prototype. We describe related work in Section 7, and our conclusions in Section 8.

2 Using a P-Router and Its Applications

2.1 Example Applications

Possible applications of the Postmanet include: email with large attachments (such as home movies), web embedded with rich multi-media objects, remote file system mirroring for sharing and/or backup, peer-to-peer file sharing of large multi-media files, publish/subscribe systems for content such as music, TV and radio programs, newspapers, magazines, store catalogs, softwares, and public lectures given at universities, and distance learning systems allowing two-way communications [13, 22]. These applications share the commonalities of their appetites for high bandwidth and their benefiting from simultaneously exploiting the HLHB postal system and an LLLB Internet connection (if one is available). More details of these applications can be found in a position paper [23, 24].

One of the applications that perhaps best illustrate the Postmanet approaches is a hypothetical example called "video almost on-demand." Instead of passively responding to customer requests and forcing customers to wait for their requested content to arrive in the postal system, a video rental company could proactively push encrypted movies to participating customers without having received explicit requests. Large encrypted libraries can accumulate on participating customers' home storage. To view a movie, a customer would purchase a decryption key on-demand from the rental company over the LLLB Internet and gain access to a locally

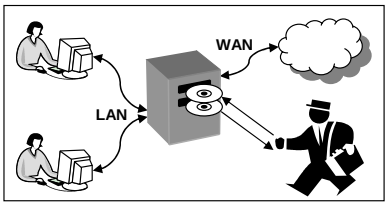

Figure 1: A Postmanet router.

stored and encrypted selection instantaneously. Emerging DRM technologies (such as Microsoft's Palladium [1]) may be needed to prevent unauthorized dissemination or re-use of decrypted content. Although we have called the Postmanet a "high-latency" channel, in this example, by exploiting the plentiful storage capacity and bandwidth of the Postmanet, and by simultaneously using an LLLB channel, one may be able to mask its high latency. This is a theme that will be revisited.

2.2 Generality and Transparency of Postmanet

While specialized solutions (such as those employed by AOL, Netflix, and some researchers working on astronomy data [8]) have emerged, they lack two key desired properties: *generality* and *transparency*: a general Postmanet should be able to cater to a variety of applications; and a transparent Postmanet should minimize manual handling of the storage media being transported. One way of better understanding the importance of these goals is to consider an imaginary Postmanet router device (illustrated in Figure 1).

A Postmanet router (or a *P-router*) is similar to a home DSL router. Instead of always forcing outgoing data through a weak wide-area network, however, the P-router writes some of the outgoing data to a mobile storage media (such as a DVD). The types of storage media used may include read-only or read-write DVDs, flash memory cards, or hard disks. We shall generally refer to these storage devices as *P-disks*. An outgoing P-disk, after being ejected from a P-router, is picked up by a postman for delivery via the postal system. The postman may also drop off an incoming P-disk, whose data appears on a user computer as if it had arrived from a conventional WAN. Therefore, unlike specialized solutions such as those employed by AOL and Netflix, the Postmanet should provide generic two-way communication, just as conventional networks do.

The user of a P-router need not manually inspect or process the content of a P-disk; the user need not manually stage or copy data; and the user need not worry about issues such as potential loss or damage of P-disks in the postal system. Unlike an AOL or Netflix user, who must know what to do manually with these application-specific disks, a Postmanet user's only direct manual interaction with the P-router is limited to the insertion/removal of P-disks into/from P-routers. This is analogous to the fact that low-level details such as packets and routers are minimally visible to a conventional network user.

When a P-router user needs to send to multiple receivers, or when multiple applications need to share the Postmanet, ideally, it would be desirable if only a single outgoing P-disk needs to be sent per postman visit. Similarly, if a P-router user needs to receive from multiple senders, it would be desirable if there is only a single incoming P-disk that contains all the incoming data. This is in contrast to ad hoc application-specific solutions, which never allow, for example, AOL and Netflix data to be placed on a single disk. The sharing of a single Postmanet infrastructure by multiple applications and multiple users is consistent with the multiplexing and demultiplexing jobs performed by conventional networks.

The provision of an application-neutral Postmanet "public transit" system that is easily and cheaply exploitable by any potential communicating parties is important. This is analogous to the fact that the existing Internet is such a generic infrastructure. Without it, a potential innovator who is interested in developing a Netflix-like application may need to reinvent the whole infrastructure from scratch. The coexistence of multiple Netflix-like infrastructures can lead to various forms of inefficiency. Smaller players may not be able to afford to put up their own infrastructure at all.

These generality and transparency goals lead us to believe that system support at various levels is necessary if we were to fully realize the potential of the Postmanet.

3 Routing

The Postmanet has some unique routing metrics. For example, an important consideration is minimizing the number of P-disks received or sent per site per postman visit. In the following discussion, when we say a site "handles" k P-disks, we mean that the site may receive up to k P-disks and send up to k P-disks per postman visit; and when we refer to a "latency" metric, unless explicitly noted, it is in terms of the number of postal system forwarding hops visible to Postmanet participants.

3.1 Routing Strategies

We consider the routing strategies illustrated in Figure 2. In the centralized alternative illustrated in Figure (a), an end user always sends/receives P-disks directly to/from a single data distribution center (called a *P-center*). Although any centralized solutions have obvious disadvantages, an important advantage of this approach is that each end user handles only a single P-disk, regardless how many other sites he communicates with per postman visit: as the P-center copies data from its incoming P-disks to its outgoing P-disks, it first demultiplexes incoming data and then re-multiplexes outgoing data, minimizing the number of P-disks handled in both directions. (Inexpensive robotic arm-operated, multi-drive DVD writers that can generate about 600 DVDs per day already exist today and they can keep manual labor cost to a minimum.)

In the direct peer-to-peer routing alternative illustrated in Figure (b), each user may need to prepare multiple P-disks for transmission, each of which destined for a different intended receiver. This approach has potentially better latency and lower infrastructure cost than that seen in Figure (a), but it may result in each site having to handle many P-disks. In a large scale peer-to-peer file sharing application, for example, the large number of P-disks handled per site could become a severe administrative and cost burden. This is an instance where the answer of "leaving routing to the postal system" is

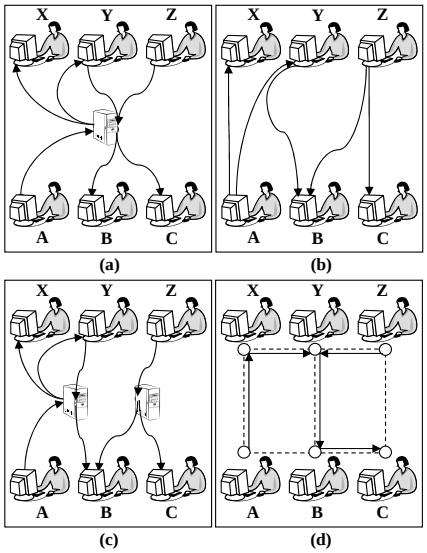

Figure 2: Routing strategies. A solid arrow denotes a single P-disk carried by the Postmanet on one postal hop. A dashed line between a pair of nodes in (d) denotes that it is permissible for these two nodes to exchange P-disks directly with each other. In all four panes, A sends different data items to X and Y, Y sends some other data to B, and Z sends different data items to B and C. (a) Centralized data routing via a single data distribution center. (b) Direct peer-to-peer data routing. (c) Data routing via multiple data distribution centers. (d) Indirect peer-to-peer routing.

insufficient.

In the multiple-P-center approach illustrated in Figure (c), the geographically distributed P-centers allow some degree of geographical awareness in routing decisions, thus achieving latencies that are potentially better than those in (a), but worse than those in (b). The number of P-disks handled per site is limited by the number of P-centers. These advantages do not come for free, however, as the P-centers may require a substantial infrastructure investment. It is also possible to allow the coexistence of the alternatives illustrated in Figures (b) and (c).

In the indirect peer-to-peer routing alternative illustrated in Figure (d), a P-disk arriving at a site may contain data destined for other sites so, in some sense, the data copying tasks of a P-center is now distributed among the peer participating sites. In (d), for example, a P-disk traveling on the $Z \rightarrow Y \rightarrow B \rightarrow C$ route delivers data sent by Y and Z to B and C. Using an analogy, one may view the P-disks as buses and messages as bus passengers: a passenger may need to switch buses to get from its source to its destination. If bus schedules are carefully planned and used, one may be able to limit the number of P-disks handled per site while still achieving good message latencies. An important advantage of this approach is that it does not require a P-center infrastructure.

A potential complication facing any peer-to-peer system is coping with misbehaving participants: a Postmanet user, for example, may fail to promptly forward data destined for his peers, alter or damage data, or read data that he is not supposed to. Routing protocols designed to deal with Byzantine faults [2] use a combination of techniques, including participant monitoring, destination acknowledgements, fault announcements, checksumming and encryption of data, authentication, fault knowledge sharing, and isolating faulty nodes. These Byzantine-tolerant protocols are directly applicable here and they can be integrated with a (suitably modified) Netflix-like service model, in which customers stop receiving additional service if they do not return outstanding discs already in their possession. Proactive data replication on multiple outgoing P-disks along different routes can further improve robustness and performance.

We summarize the desired Postmanet routing characteristics: (1) it can accommodate a large number of simultaneous Postmanet communicators without requiring a site to handle many P-disks per postman visit; (2) it has end-to-end message propagation latencies that are close to those provided by the postal system; (3) it does not require an expensive infrastructure other than the existing postal system; (4) it does not burden Postmanet nodes in an unbalanced manner with data copying tasks that are beyond their own communication needs; and (5) it is robust when faced with misbehaving Postmanet end users. Some of these goals are unique to the Postmanet; these goals often conflict with each other; and we need to strike a proper balance among them.

Option (a) is a special case of option (c); and option (b) can be seen as a special case of option (d). If we can afford it, a properly provisioned infrastructure in terms of a number of geographically distributed P-centers (option (c)) should give the best quality of service. Ideally, the P-centers should be integrated into the existing postal system (or its rough equivalent, such as UPS or FedEx) so that some or all of the post offices themselves serve as P-centers, further minimizing delivery latency. Without relying on a P-center infrastructure, the peer-to-peer model (option (d)) is the quickest way of deploying a Postmanet. It is also possible to mix options (c) and (d). Although it is not the only viable model, we believe that the peer-to-peer Postmanet model is an important one if we were to realize the incremental deployment benefit (explained in Section 1.1). It is this model that we focus on first; and we examine later how P-centers can be integrated into this model.

3.2 Problem Definitions

• *Static routing graphs.* In Figure 2(d), suppose each user is only allowed to directly exchange P-disks with "neighbors" along the dashed lines. By constraining the number of such neighbors for each node, we limit the number of P-disks handled per site. A natural question is how such neighbors are chosen. In graph theoretic terms, the problem of simultaneously limiting the number of P-disks handled per node and maximum latency can be seen as that of constructing a directed graph with a large number of nodes while keeping the diameter and the maximum node degree small. The diameter corresponds to the maximum latency, and the degree of a node corresponds to the number of P-disks it handles. Although the problem of constraining both graph degree and diameter is applicable to general networks, we shall see that the quantitative tradeoffs involved in the Postmanet (between postal system delays and the number of P-disks handled), and the need of generalizing the problem dynamically present unique challenges.

• *Dynamic routing.* The problem posed above concerns a static topology: a Postmanet node may directly exchange P-

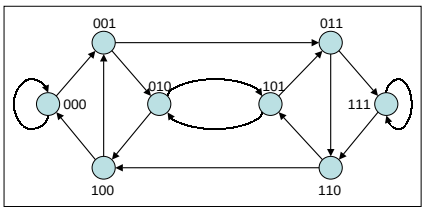

Figure 3: A 3-dimensional de Bruijn graph.

disks only with a small number of pre-determined neighbors. These static constraints may be unnecessarily restrictive. For example, in Figure 2(d), if C desires to send data to A, its data would normally be routed through B. But, there is no reason why C should not be allowed to send a P-disk *directly* to A if, on a given day, it does not overburden either of them. The question concerning a more dynamic approach is how to allow for such routing flexibilities without causing problems such as too many P-disks being handled by any one node on any given day. This is a routing optimization problem unique to the Postmanet.

• *Disseminating routing information and coordinating routing actions.* The questions are: (1) how is the traffic information (in terms of who is sending to whom) gathered? (2) who computes the routes? and (3) how are the computed routes disseminated?

• *Geographic awareness.* Obviously, not all postal hops are equal in terms of their geographic distances and postal delays. The question is how to construct routing graphs that can account for these factors.

• *Integrating P-centers.* We would like to understand how to best integrate P-centers into our routing mechanism, incrementally if necessary, to improve service quality.

3.3 Solutions to the Routing Problems

We now answer each of the questions posed in the last section.

3.3.1 Static Routing Graphs

Although dynamic routing should undoubtedly out-perform the static approach, especially under light workloads, finding good static topologies is important for two reasons: (1) a good static routing graph may form the basis of a good dynamic routing algorithm; and (2) a good static routing graph may provide a performance upper-bound for a uniformly heavy workload, which may present few exploitable optimization opportunities for any dynamic approach. We examine two types of static routing graphs for use in the Postmanet: de Bruijn graphs [4] and random graphs.

An r-dimensional de Bruijn graph consists of 2^r nodes. Each node is associated with a distinct r-bit binary string, and a node identified by the binary string $b_1 b_2 \cdots b_r$ has directed edges leading towards the nodes identified by $b_2 \cdots b_r 0$ and $b_2 \cdots b_r 1$. (Figure 3 illustrates an 8-node de Bruijn graph.) Each node has both an indegree and outdegree of two. To route from a node $u_1 u_2 u_3 \cdots u_r$ to a node $v_1 v_2 v_3 \cdots v_r$, one simply routes through the intermediate nodes $u_2 u_3 \cdots u_r v_1$, $u_3 \cdots u_r v_1 v_2, \ldots, u_r v_1 v_2 \cdots v_{r-1}$, thereby resulting in a system with diameter of $\log N$. (Recall that diameter corresponds to maximum Postmanet latency expressed in postal hops, and node degree corresponds to maximum P-disks handled per site.) [1]

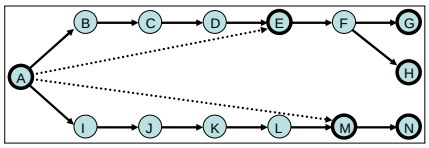

Figure 4: A dynamic routing example. The dark arrows are the edges in the underlying static routing graph. The dotted edges are the "short-cut" edges that A uses to directly forward messages to E and M.

Although random graphs can also achieve $O(log\ N)$ diameter with constant node degree, unlike de Bruijn graphs, the diameter bound is probabilistic. Furthermore, compared to de Bruijn graphs, random graphs tend to require a larger node degree constant to achieve a comparable diameter bound.

3.3.2 Disseminating Traffic and Routing Information

In traditional networks, implicit routing, wherein routing decisions are made locally without requiring elaborate knowledge of the global topology, can be very useful. In contrast, implicit routing may be of lesser importance in a Postmanet that has two "networks"—the LLLB Internet could be used for dispersing topology information or topology repairs, while bulk data traverses the HLHB channels.

In a similar vein, we can also use the LLLB channel to disseminate traffic information (in terms of who desires to send bulk data to whom). This makes the dynamic routing problem easier to solve. We may assume, for example, that traffic information is continuously being gathered at a centralized coordinator site over an LLLB channel. The coordinator uses the gathered information to compute the best dynamic routes, which the coordinator then disseminates to all the participating peer Postmanet sites, so by the time a postman arrives at a site to pick up outgoing P-disks, appropriate next-hop postal labels would have been generated at each site according to a global schedule and affixed to these outgoing P-disks. Furthermore, as much as 24 hours, for example, may elapse between successive postman visits, so the coordinator may have ample time computing the best dynamic routes. Multiple coordinators can be employed to improve reliability and performance.

3.3.3 Dynamic Routing

In Section 3.3.1, we have described a static routing strategy, where a message from a node s destined for a node t is always routed along the shortest path in the underlying de Bruijn graph. As pointed out in Section 3.2, this can be overly restrictive, especially when some nodes are lightly loaded. Consider the example in Figure 4. The figure shows a portion of the underlying static graph, where out- and in-degree of each node is constrained to be at most two. (Not all graph edges are shown here.) Assume that at some stage, node A only has messages destined for nodes E, G, H, M and N. In

[1] When the number of nodes involved is not an exact power of 2, a static routing graph can still be obtained by starting with a larger de Bruijn graph, and "routing through" non-existent nodes. Also note, choosing k as a "base" would result in a degree-k graph.

this case, instead of using the edges in the underlying graph, A may use the "short-cut" edges $A \rightarrow E$ and $A \rightarrow M$, and in a single step, send the messages destined for E, G and H directly to E, and those destined for M and N directly to M.

A good "dynamic routing algorithm" for Postmanet would make decisions of this kind in an optimal manner. Specifically, it would be an "on-line" algorithm that for each postman visit at each site, determines the next-hop destinations for the out-going P-disks, and also selects the set of messages to put on those P-disks. The goal is to make progress toward delivering messages to their respective destinations, while respecting the degree constraints on the nodes per postman visit. One way to measure incremental "progress" made by the system toward delivering a given message is to measure how close in the underlying static graph the message has reached to its eventual destination. A greedy optimization algorithm can then, at each step, try to choose the edges so as to quickly make as much global progress as possible.

In our proposed dynamic routing approach, an algorithm is run at the end of each step (or day) to determine the edges along which to ship P-disks on the next day. Our algorithm constructs a bipartite graph with vertex set $P \cup Q$, where each node of the system appears exactly once in both P and Q. Edge $p \rightarrow q$ is assigned a weight proportional to the progress that can be made by sending a P-disk from node p to q directly. The problem then reduces to choosing a set of edges so as to make as much total progress as possible. For this purpose, a maximum-weight matching algorithm is repeatedly invoked to find a set of matchings along which to ship P-disks. The weights on the edges can also take into account other factors such as message priorities, delivery deadlines and starvation. With a suitable choice of the progress metric, the dynamic routing algorithm would degenerate to the static routing algorithm under heavy message traffic. (Specific progress metrics will be discussed in a later section.) Thus, in the worst case, the performance of the dynamic algorithm would be no worse than that of the static algorithm, while under lighter load conditions, the dynamic algorithm would perform much better.

It is interesting to note that under this dynamic routing approach, the dynamically chosen routes are by no means obliged to follow any edges in the static underlying de Bruijn (or random) graph. The static graph's sole purpose is providing a means for the dynamic algorithms to gauge progress when greedily choosing next hops. In some sense, the static graph acts as a "traffic shaper," whose influence should be the strongest under extremely heavy workloads, which we conjecture would force the dynamically chosen routes to more closely conform to the shortest paths in the static graph.

Although the role of a static graph is only a traffic shaper, it is important for the static graph to have a node degree constraint that is identical to that of the dynamic routing graph, which should reflect the real-life limitation of how many P-disks a site handles. Had we chosen a static graph with a higher node degree and enforced a lower node degree only in the dynamic routing algorithm, the progress metric derived from the static graph could be overly optimistic, potentially resulting in too many messages being delivered to a site that cannot drain them quickly due to a low dynamic degree limit.

It is possible to model dynamic routing as a more precise optimization problem, and to try to achieve "theoretically-optimal" solutions. However, there seems to be little hope of finding such optimal solutions for two reasons. First, even the off-line version of our problem (where all of the requests are available at the beginning of the computation) appears to be NP-hard because of its relationship with the well-studied multi-commodity fixed-charge problems. Second, our real interest lies in devising an on-line algorithm that executes continuously and handles a multitude of events occurring in the system, and not all versions of our problem may be easily amenable to theoretically optimal solutions. Hence we use a greedy, heuristics-based approach.

3.3.4 Geographic Awareness

We use two techniques to make Postmanet routing geography-aware. First, we embed the static routing graph onto the set of participating nodes in a geography-aware fashion. This is achieved by using a Dijkstra-style greedy algorithm that tries to ensure that the postal system latencies along the graph edges are not too large. (Details of the algorithm are omitted due to space constraints.) Second, actual postal system latencies are taken into account when assigning weights to the edges in the bipartite graph used in the maximum-matching computation. We study the effects of both of these techniques in a later section.

3.3.5 Integrating P-Centers

We next consider how to integrate P-centers into our peer-to-peer routing infrastructure. P-centers, with their ability to provide two-hop connectivity between any pair of nodes, could be used to either service only some high-priority messages, possibly generated by paying customers who require predictable quality of service, or improve the latency of all messages by providing short-cuts in the routing infrastructure. The optimization problem, in either case, is to compute a set of source nodes and a possibly overlapping set of destination nodes for which a given P-center would serve as a hub on a given day in order to maximize the progress of the messages in the system. Each node is constrained to send to (or receive from) the P-centers at most one disk on any given day. A given P-center would not be statically bound to a fixed set of nodes, thereby allowing it to adapt to varying traffic conditions. Once again, theoretically optimal solutions for this problem are intractable even for the special case of augmenting the infrastructure with just one P-center, and we therefore resort to the following heuristic.

We determine the routing connectivity for one P-center at a time during each routing step. We begin by greedily picking a source node that would attain the greatest benefit from using the P-center to communicate its messages to at most d_{pc} destinations, where d_{pc} is the out-degree constraint on the P-center. We then pick the next source node based on a metric that takes into account both the amount of message traffic to some set of d_{pc} destinations and the amount of message traffic to only those destinations that are favored by the first selected node. We repeat this process, and at each step, we keep track of the most popular destinations corresponding to the current set of selected source nodes and pick the next

source node based on this information. Once we have picked all the source nodes, the most popular d_{pc} destinations are selected as the target nodes to which the P-center will send a disk. Based on the final selection of the target nodes, each source node will then compute what messages it will send to the P-center node, including on the P-disk any message that would make faster progress through the P-center than through the peer-to-peer infrastructure.

4 End Point Support

We have considered how P-disks are routed in the last section. We now consider support at the communication end points.

4.1 Postmanet Characteristics and Implications

The Postmanet has several unique characteristics, which require treatment different from that in conventional networks.

• *Datagram limitations.* The postal system represents a classic analogy of a datagram service: individual P-disks may be damaged, lost, delayed, or delivered out of order. Human users or individual applications should not have to cope with these complications if they desire better guarantees and abstractions.

• *Bursty arrival of large amounts of data.* A single Postmanet sender could have spent many hours writing to a P-disk, and data from multiple sources can arrive at a receiver per postman visit. Gigabytes or even terabytes of data could be involved. A (human or application) receiver would naturally desire to gain access to the newly arriving data as quickly as possible. A naive approach of forcing the receiver to wait until the system completes copying from incoming P-disks to local storage could add substantial delay. Instead, it is important to allow receiver applications quick access to summary or metadata information so that they can make flexible decisions before a large amount of data needs to be copied. This is an issue that does not arise in conventional networks that allow gradual and continuous data arrival.

• *Two networks.* The aid of an LLLB Internet connection makes the Postmanet more powerful and interesting. In addition to using the LLLB channel to carry small control messages such as acknowledgements, the sender system may choose between the LLLB Internet and an HLHB P-disk based on factors such as the amount of data to be sent and the desired arrival time. One may even choose to use both channels in parallel. For example, a Postmanet application may prepare multiple versions of an object (at different resolutions) for simultaneous transmission in the LLLB and HLHB channels.

• *Delayed action.* Conventional networks typically do not support, for example, an "unsend" operation, that allows a user to change his mind after a "send" operation is executed, because there is typically little time before actions are effected. In the Postmanet, however, there is ample opportunity for mind-changing: before the postman picks up the outgoing P-disk at the sender, as the P-disk is in transit in a P-center or in a peer's P-router, or even after the P-disk arrives at the destination but before the data is consumed by the receiver application. Even in absence of a mind-changing sender, a receiver application may discover that some of the newly arriving data is no longer needed due to application-specific reasons. In any case, the LLLB channel can be used to "shoot down" a message in any stage of transmission between the sender and the receiver end-points.

1. **`Entry.getName()`**
   name of this `Entry`.
2. **`Entry.getFullName()`**
   full path of this `Entry`.
3. **`Entry.create(name, type)`**
   create subentry. `type`: `File` *or* `Dir`.
4. **`Entry.delete(name)`**
   delete a subentry.
5. **`Entry.deleteAll(name)`**
   delete a subentry recursively.
6. **`Entry.get(name)`**
   returns the specified subentry.
7. **`Entry.list()`**
   list names of subentries of this `Entry`.
8. **`Entry.listEntries()`**
   returns list of subentries in this `Entry`.
9. **`Entry.search(filter)`**
   list subentries that match `filter`.
10. **`Entry.isFile()`**
    test whether this `Entry` *is a* `file`.
11. **`Entry.isDirectory()`**
    test whether this `Entry` *is a* `Dir`.
12. **`Entry.size()`**
    returns size of this `Entry`.
13. **`Entry.getFD()`**
    gets the "file descriptor" for this `Entry`.

(a) **`Entry`** interface (partial).

1. **`FD.lseek(offset, whence)`**
   set this pointer to specified `offset`.
2. **`FD.getOffset()`**
   returns current offset.
3. **`FD.length()`**
   returns size of this `File`.
4. **`FD.sync()`**
   flush changes to disk.
5. **`FD.read(bytes)`**
   returns next `bytes` *of data.*
6. **`FD.read(count,prefetch)`**
   similar. `prefetch`*: this is low priority.*
7. **`FD.write(data)`**
   write given data to this `File`.
8. **`FD.write(srcFD, offset, bytes)`**
   write `bytes` *starting from given source* `srcFD`*'s given* `offset`.

(b) **FD** interface.

1. **`Entry.getAttributes()`**
   returns all attributes of an this `Entry`.
2. **`Entry.getAttributes(nameSet)`**
   returns only specified attributes.
3. **`Entry.getAttribute(key)`**
   returns specified attribute.
4. **`Entry.modifyAttribute(key,value)`**
   modify/add specified attribute.

(c) **`Entry`** attribute interface.

Figure 5: Local storage interfaces (available to applications).

• *P-disk communication media.* Large-capacity P-disks play the role of wires. A P-disk may hold many messages, which require the data to be organized in a more structured fashion than that typically employed on a wire. A natural question is what type of structure we should use: for example, a database, a file system, or some other customized data structure? The physical organization of storage management is also a relevant issue. For example, a log-structured approach [16] may allow small message "sends" and certain types of receiver copying to execute efficiently.

4.2 API Overview

The most important means of addressing the unique Postmanet characteristics discussed above is well-defined APIs. Properly defined APIs should (1) abstract away unpleasant details (such as the datagram limitations of the postal services), (2) expose new capabilities (such as ways of using two networks), and (3) allow applications to circumvent performance difficulties (such as the problems associated with bursty arrival of large amounts of data). We give an overview of the interfaces, before we later explain how they address the Postmanet-specific characteristics.

There are three key sets of interfaces: (1) an interface that allows applications to manipulate data on P-disks (Figure 5), (2) an interface that controls sending and receiving of data (Figure 6), and (3) an internal interface used by P-routers that communicate with each other (but not visible to applications) (Figure 7).

An `Entry` is a basic P-disk-resident object that roughly corresponds to a Unix file or a directory (5(a)). Unlike conventional files, however, additional Postmanet-specific semantics and operations are built on top of `Entries`. `FD`s, similar (but not identical) to file descriptors, mainly allow data to be read/written to P-disks (5(b)). Beyond the file

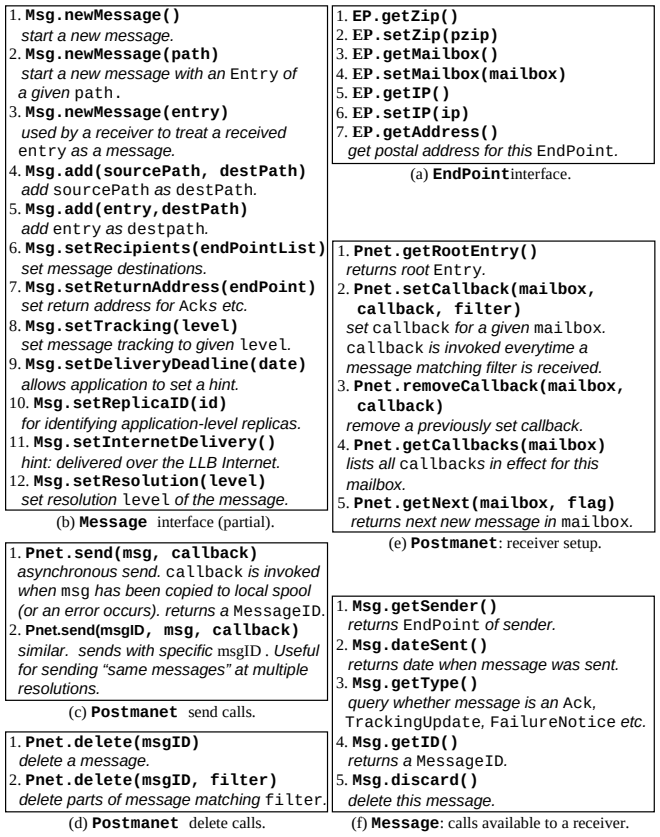

Figure 6: Communication interfaces (available to applications).

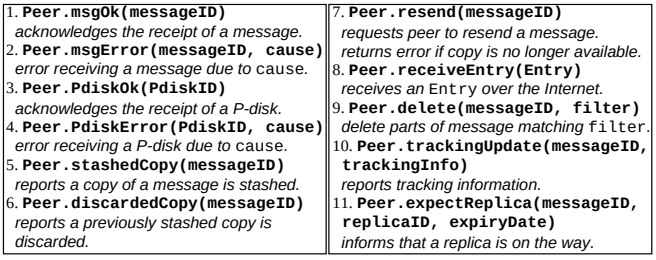

Figure 7: Peer interfaces (hidden from applications).

system-like operations, an `Entry` also has associated "attributes," or `(Key, Value)` pairs, which are used by both the system and applications (5(c)). (As examples, the intended recipient identity would be a system attribute of the outgoing data; and the URL of a web-based publication would be an application-specific attribute.)

A `Mailbox` is a directory `Entry` under which an application finds incoming data. To send data, one needs to specify a destination `EndPoint` (6(a)), which contains a `Pzip` that identifies the receiver machine, a `Mailbox`, and optionally, the IP address of the destination machine. (Such addressing information can be provided by a separate lookup service analogous to the DNS service of today. Lookups can leverage the LLLB Internet. We are using a simple local file in the current prototype.)

`Messages` are `Entries`. A sender manufactures `Messages` using the interface of 6(b). These calls essentially allow one to set a variety of attributes, which specify various delivery options. Of all these calls, only `setRecipients()` is necessary. Because `Messages` are `Entries`, all supported calls of `Entries` (5(a)) are also available for `Messages`. Once a message is created, it can be sent using the calls of 6(c), or "unsent" using those of 6(d) (if one were to change his mind). Postmanet provides "reliable" messaging but it currently does not guarantee in-order delivery. It allows one-to-many communication.

To receive messages, an application sets `Callbacks` on its `Mailbox` (6(e)). When a `Callback` is invoked, the `Entry` that resulted in its invocation is passed as an argument to the callback function. Note that the callback function would need to explicitly perform read operations using the interfaces given in Figures 5 and 6(f). The data being read, however, may or may not have been moved from an incoming P-disk to other local storage at the receiver by the system. This allows both good performance, for applications that desire to control their own data movement from a P-disk into application-specific local store, and convenience, for applications that do not want to be bothered with such low-level details.

The peer interfaces shown in Figure 7 allow peer P-routers to communicate with each other, mainly over the LLLB Internet channel. These interfaces are not visible to applications. They allow P-routers to manage control information such as acknowledgements, failure notifications, retransmission requests, message shoot-downs, replica management, and message tracking. Small data messages are also transmitted over the LLLB Internet using this interface.

4.3 Managing Postmanet-Specific Characteristics

We now discuss how the unique characteristics of the Postmanet (Section 4.1) are managed by (and behind) the interfaces given above (Section 4.2). These characteristics interact in interesting ways: the problems caused by some of these characteristics can be addressed by opportunities offered by other characteristics. For example, the datagram limitations and poor latencies can be improved if we judiciously exploit the availability of two networks and the excess capacity on P-disks.

• *P-disk organization.* The "messages" are organized in a hierarchal file system, with additional attributes and supported operations added to the file system-like objects. This arrangement makes the system easy to use for applications, many of which would find a file system-like interface natural. A sending application can prepare the outgoing data in a format that its receiving counterpart can readily integrate into its own persistent data structures. Minimum packing, unpacking, or conversion should be necessary. What we are seeing is a form of blurring the boundary between storage and networks. (The issues being explored here, however, as we explain in Section 7, are quite different from those seen in distributed storage and file systems.)

• *Two networks.* The LLLB Internet is made available for use to both the system and applications. The peer interfaces (Figure 7) allow peer P-routers to exchange various types of small control information. The `Message` interface allows applications to provide explicit or implicit hints (including

delivery deadlines, and whether a message is a low resolution version of a bigger P-disk-resident message) on whether and when to use the LLLB Internet channel. (See calls 9, 11, 12 of Figure 6(b).)

• *Bursty arrival of large amounts of data.* We have several potentially conflicting goals: (1) copying all the data out of an incoming P-disk to receiver local storage as quickly as possible; (2) allowing applications to make progress without having to wait for extensive copying to complete; and (3) minimizing application interference with each other, which could result from competing data copying activities.

The callback interface (Figure 6(c)) allows applications to read a minimum amount of summary information to get started, and then to read data strictly on a need-driven basis. Behind the callback interface, a key P-router component is a generic system-level *background copier* that copies data from an incoming P-disk to local storage. If an application chooses to discard some incoming data (for any application-specific reasons) before it is reached by the background copier, this data does not need to be copied at all. As the background copier proceeds, the system ensures that the `Entry` passed to the application callback function points to the correct storage location, which could be either the P-disk or the local storage. The background copier may be able to aggressively exploit sequentiality of the underlying storage organization. The background copier, however, needs to exercise care not to compete with applications for P-disk bandwidth. The background copier also provides a means for applications to avoid interfering with each other: a "well-behaved" application should always read only what is necessary and leave the rest to the background copier. (This is based on the assumption that the receiver applications fed by the same P-router are willing to be cooperative, in terms of performance.)

• *Datagram limitations.* The handling of damaged, lost, delayed, or system-replicated Postmanet messages is not visible to the application-visible interfaces of Figures 5 and 6. The peer P-router interface of Figure 7 allows the system to quickly deal with these anomalies using the LLLB Internet. Furthermore, as multiple P-disks are sent between a sender-receiver pair on successive days, the system may liberally replicate outgoing data of earlier days on outgoing P-disks sent on later days. In cases where a single P-disk is delayed or lost due to accidents in the postal system or uncooperative peers who were supposed to forward it but did not, the replicated data on a subsequently arriving P-disk is just a day away, so we can avoid unnecessary long end-to-end retransmission delays. This is another example of the consistent Postmanet theme of liberally "wasting" plentiful resources (storage capacity) to optimize for more difficult metrics (lower latency or better reliability).

• *Delayed action.* A message can be canceled at any point after it is sent and before it is consumed at a receiver. Such a cancel message may need to be buffered at a destination. Shoot downs can be useful for functionality or performance reasons. Applications use the interface in Figure 6(d) to initiate a shoot down, which is handled locally if the message has not left the sender, or generates one or more peer shoot down messages (call 9 of Figure 7) if the P-disk containing the message has departed. Shoot downs can also be initiated at the system level without application initiation. For example, extra system-level replicas, such as those described in the last paragraph, should be shot down, when it becomes apparent that the outstanding replicas are no longer needed.

Security concerns can be largely addressed using existing mechanisms. (We omit detailed discussions due to space constraints.) In summary, the Postmanet has a number of characteristics not seen in conventional networks. We believe that careful interface design is an important way of addressing these issues. We do not, however, claim that we have arrived at the ideal interfaces, and we are continuing to evolve and refine these interfaces.

4.4 Implementation

We have implemented a prototype P-router in Java. All communication between applications and a P-router, and between P-routers is done via Java's Remote Method Invocation (RMI). The implementation contains three main modules: the sender part, which handles send and delete requests, the receiver part, which handles integration of data on incoming P-disks (or over the Internet) with the rest of the system, and a local store, which provides the `Entry` abstraction. The attributes of entries, and the message tracking and management information used by the above modules, is stored by an LDAP [14] server, running over a BDB [20] back end, accessed via JNDI. On the DVD P-disks, data is written as an ISO file system with attributes stored in separate files. The ISO images are staged on a local disk but partial images can be incrementally appended to DVD+RW discs that we use in our prototype. The entire implementation is based on JDK v1.4.2_04 and is about 10,200 lines of java code.

4.5 Sample Applications

We briefly describe aspects of three simple Postmanet applications that we have developed. The first one is PwebCache, a web proxy that receives subscribed data from a Postmanet-aware web publisher. The publisher creates data in an entry hierarchy to correspond to the on-disk structure used by the cache. The URL of the page, and cache validation and control headers of the HTTP protocol, are stored as attributes of entries and the `deliveryDeadline()` of the message is set to the cache expiration date of the data, if any. When a P-disk arrives, the receiver cache program only needs to record the URL attributes of the top level entries to be able to start servicing client requests immediately, while leaving the general system background copier with the task of moving data out of the P-disk.

The second application is Pnapster, a Napster-like application. The file lookup and request issuing parts are performed over the Internet and are no different from existing Napster-like applications. Multiple peers who have copies of the requested content may receive requests, so the requester may enjoy the quickest reply. A peer request receiver sets the `replicaID` to a "request ID," which allows Postmanet to manage (and shoot down, when necessary) these application-level replicas. When a "preview" request is received, a peer that has the desired data generates small (low-resolution) versions of the bulk data, invokes `setResolution()`, and `setInternetDelivery()` to hint it be sent over the In-

ternet. A data requester may invoke `delete()` at any time to cancel a request.

The third application is Pemail, an email application. It uses `setTracking()` to acquire delivery status of outgoing messages. Pemail also generates small previews of bulk messages so the previews can be delivered over the Internet. A Pemail receiver may issue `delete()` calls with specific `filter` arguments to delete parts of messages before they are copied out of an incoming P-disk by the background copier.

5 Routing Simulation

5.1 Simulation Methodology

We have developed an event-driven simulator to study the various routing strategies described in Section 3. Our simulator allows us to systematically evaluate the performance and scaling properties of the various algorithms under different workloads, study the effects of using different kinds of static graphs, examine ways of mapping abstract graphs to real-life Postmanet configurations, and evaluate the benefit of integrating P-centers into the routing infrastructure.

The nodes used by our simulator correspond to randomly chosen USPS zip codes, located at real-life geographic coordinates. The simulator uses a latency matrix, enumerating latencies between all pairs of nodes. We examine two types of latency matrices. In one type, all latencies are equal to one day. This "uniform latency matrix" corresponds to a fast delivery service (such as FedEx). In another type, the latencies are set to be proportional to the geographical distances between nodes. At one day per 500 miles, with a maximum latency of eight days in the lower 48 states and a minimum of one day, and with the inclusion of nodes in Hawaii and Alaska, this second type of matrix represents a pessimistic assumption of the delivery service speed, a speed that is in fact worse than that experienced by DVDs delivered as first class USPS mails. We shall refer to this as the "USPS latency matrix." By choosing to use these two very different types of latency matrices, we hope to get some idea on the range of Postmanet latencies one might see in real-life.

For the experiments in this section, we use a parameterized, random workload, where each node generates λ new unit-sized messages each day destined for λ distinct, randomly-selected other nodes. The parameter λ is referred to as the "Average Message Load" of the workload, or simply as the "load." All workloads in our study contain 60 days worth of message traffic. To account for our conjecture that people either do not communicate or communicate with more than an average number of other parties, we have introduced "burstiness" into the workload.

5.2 Comparison of Routing Algorithms

We now compare static and dynamic routing algorithms, and also study the impact of using different progress metrics for the dynamic algorithms.

In the *Static* algorithm, each node only sends P-disks to its neighbors in the underlying static graph each day. The *Prefix* algorithm is a dynamic algorithm that chooses "short-cut" edges that correspond to multiple hops in the underlying static graph, as discussed in Section 3.3.3 and illustrated by Figure 4. The chosen short-cut edges are, however, constrained so that no message ever overshoots its destination. For example, in the topology of Figure 4, suppose A has 1 message each for E and M, and 1,000 messages each for G and N. Although edges $A \rightarrow G$ and $A \rightarrow N$ make greater incremental progress in terms of delivering messages, the *Prefix* algorithm is constrained not to overshoot E and M, thus choosing edges $A \rightarrow E$ and $A \rightarrow M$. The *Match-Hops* algorithm is a dynamic algorithm that is not hobbled by the above constraint, but instead uses a maximum-weight matching technique to maximize the sum progress of all the messages through the network. (See Section 3.3.3.) The progress metric associated with transmitting a message over an edge is simply how much closer the message is to its eventual destination in terms of number of hops in the static graph. The *Match-Lat* algorithm uses a different progress metric that takes into account the postal system latencies (and not just hop-counts) in determining how much closer the message is to its final destination in the static graph. Our implementation of these algorithms uses Goldberg's Network Optimization Library [7].

Figure 8(a) shows the performance of three routing algorithms for a network comprising of 1,024 nodes. A de Bruijn graph of degree two (referred to as "DB-2") is used as the underlying static graph, and the uniform latency-matrix is used to specify inter-node latencies.[2] We vary the "load," which is the average number of messages generated at each node on each day in the workload, and measure the average message latency (in days). (The latency in the figure is greater than one even when load is no greater than one because of the workload burstiness: the instantaneous load tends to be higher than average when a node communicates.) The following observations can be made from this graph. (1) *Static* uses only the edges in the static graph, and yields an average latency that is precisely the average distance between a pair of nodes in the underlying static graph, a value that does not vary with the load. (2) The two dynamic algorithms perform much better than *Static* when the network is lightly loaded. As the load increases, their performance gracefully degrades and approaches that of *Static*. (3) *Prefix* degenerates to *Static* as soon as the average load approaches the number of P-disks each node handles (two in this case), whereas *Match-Hops* out-performs *Static* for a much wider range of load values.

Figures 8(b) and 8(c) present results from similar executions, except that here the USPS latency-matrix is used instead of the uniform one. In Figure (b), a *geography-unaware,* random embedding is used to assign the physical nodes to the de Bruijn graph nodes, whereas in Figure (c) a *geography-aware* mapping generated via a Dijkstra-style greedy algorithm is used. Figure 8(d) plots the *Match-Hops* and *Match-Lat* curves from Figures 8(b) and 8(c) on the same scale to aid us in the task of comparing the different schemes. We begin by observing that *Static* performs much better in Figure (c) than in Figure (b), which is evidence that our geography-aware greedy algorithm produces a mapping that has significantly smaller postal latencies along the graph

[2] For the uniform latency-matrix, *Match-Lat* has the same behavior as *Match-Hops*, and is therefore omitted from the figure.

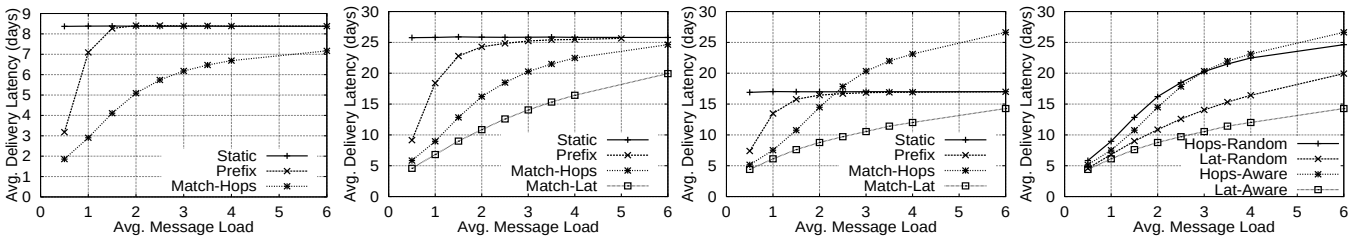

Figure 8: Comparison of different routing algorithms and geography-awareness techniques. The runs in (a) use the uniform latency-matrix, whereas those in (b) and (c) use the USPS latency-matrix. A random mapping of physical nodes to de Bruijn graph nodes is used in (b), while (c) uses a geography-aware mapping. (d) plots some curves from (b) and (c) on the same scale for comparison.

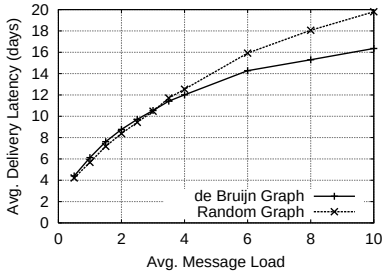

Figure 10: Comparing a degree 2 de Bruijn graph with a degree 2 random graph as the underlying static graph for the Match-Lat algorithm. We use the USPS latency matrix for this experiment.

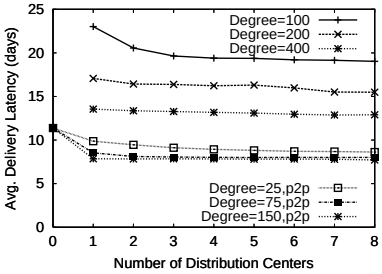

Figure 11: Impact of distribution centers on routing performance.

edges than a random mapping. We also observe that *Match-Lat* significantly outperforms *Match-Hops*, mainly because the latter algorithm uses a progress metric that is oblivious to the non-uniform nature of the postal latencies. This, combined with its greedy nature, makes *Match-Hops*'s performance even worse than that of *Prefix*, which benefits from its conservative approach of not overshooting its destinations and simply degrades to using the static graph edges in the worst case.

In Section 3.3.3, we conjectured that the use of a static underlying graph acts as a "traffic shaper" for the dynamic algorithms, especially under heavy load conditions. Figure 9 presents evidence to support this. Consider Figure 9(a) first. It describes executions of the geography-aware *Match-Lat* algorithm on four workloads of different average loads. In each execution, we count the number of times the algorithm picks a short-cut edge that spans k de Bruijn edges, and the curve is a *cumulative frequency distribution* of these counts. In other words, a data point (x, y) on a curve signifies that $y\%$ of the short-cut edges selected by the algorithm span x or fewer de Bruijn edges. For example, when 100% of the chosen edges span only one de Bruijn edge, we effectively have a traffic that entirely flows along the de Bruijn graph. Looking at the four curves, we observe that as load increases, a higher fraction of the chosen edges span only a small number of de Bruijn edges; that is, under high loads, the dynamic traffic more closely conforms to the underlying de Bruijn graph. Figures (a)-(d) show results from different de Bruijn graphs and different latency matrices, and they all support the same conclusion.

5.3 Comparing de Bruijn and Random Graphs

Figure 10 shows a comparison between using a de Bruijn graph or a random graph as the underlying static graph. The curves show the performance of the geography-aware *Match-Lat* algorithm where each of the 1,024 nodes has degree 2. As the graph shows, de Bruijn graph-based execution performs better than that based on a random graph under high message loads. Here we omit results that show that the difference between the two graphs is less pronounced when each node has degree 4 or more, or when the uniform latency matrix is used. The degree 2 case shown in the figure is realistic enough to make our use of the de Bruijn graph worth-while.

5.4 Integrating P-Centers

We next study the impact of integrating data distribution centers (or P-centers) into the peer-to-peer infrastructure. Figure 11 shows the performance of a dynamic routing algorithm (*Match-Lat*) on a 1,024-node network that uses a de Bruijn graph of degree two as the static underlying graph. The workload under consideration generates on average five messages per node each day. We vary the number and the degree capacity of the P-centers and evaluate the routing performance under the two following settings: (1) the P-centers serve all of the message load in the system (with the peer-to-peer infrastructure remaining unused), and (2) the P-centers share the routing load with the peer-to-peer infrastructure. (For instance, the curve with the legend "degree=150, p2p" corresponds to augmenting peer-to-peer routing with P-centers that can handle 150 incoming P-disks and generate 150 P-disks every day, while the top three curves in the figure correspond to using just P-centers for routing.) We note that a modest start of augmenting the peer-to-peer infrastructure with just one P-center causes a modest improvement on performance, but the marginal benefit of either increasing the

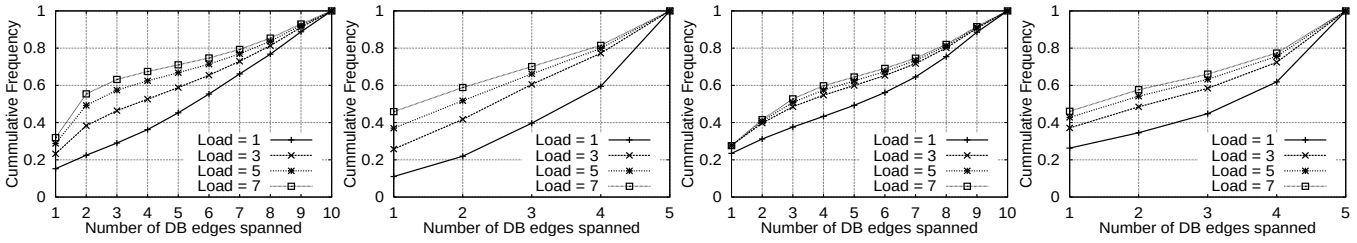

Figure 9: Analyzing the traffic generated by the geography-aware Match-Lat algorithm (a) on a DB-2 graph with the uniform latency matrix, (b) on a DB-4 graph with the uniform latency matrix, (c) on a DB-2 graph with the USPS latency matrix, (d) on a DB-4 graph with the USPS latency matrix.

DVD Writer	NEC ND2500A, 4× DVD-RW/+RW, 8× DVD-R/+R
DVD Media	Memorex 4× DVD+RW, 4.7 GB
OS	Linux 2.4.22 (Fedora Core 1)
Java	JDK 1.4.2_04
CPU	Pentium 3 800 MHz
Memory	128 MB
HDD	Maxtor 40 GB

Table 1: P-router machine characteristics.

Num. Mailboxes	8	10	10	10
Num. Messages	20	60	100	152
Data Size (MB)	80	366	966	2358
Case 1	34 s	172 s	458 s	1111 s
Case 2	134 ms	205 ms	253 ms	321 ms
Case 3	101 ms	103 ms	103 ms	103 ms

Table 2: Comparing startup times.

	news.yahoo.com	www.cnn.com
Naive	14.5 s	1047.9 s
Intelligent	6.6 s	256.3 s

Table 3: Exploiting knowledge of physical storage organization.

number of P-centers or their degree capacity is small. We also note that a single P-center with a large degree capacity appears to perform better than having many P-centers with a smaller degree capacity. We also observe that the peer-to-peer infrastructure, with no additional P-centers, appears to outperform many stand-alone P-centers of degree capacity 400. This behavior could, however, be an artifact of our heuristic algorithm for routing through P-centers, and more detailed analysis of P-centers is left for future work. The real value of well-run P-centers may lie in the predictability and stability of their service.

6 Measuring the P-Router Prototype

We set up an old desktop machine (see Table 1) to function as a P-router. Performance-wise, perhaps the most interesting aspects are about how we handle the bursty arrival of a large amount of data (as discussed in Section 4.3), and we mainly focus on these aspects in this section.

We experiment with the three applications that we have described in Section 4.5. The sending applications create an outgoing DVD P-disk that contains the following data. A Pemail mailbox contains 100 messages, varying in size between 10-20 MB. The messages contain high-resolution images, home videos, and movie trailers. The PwebCache mailbox contains three messages. (Recall a message or an entry can be of a directory type that includes more sub-entries.) One message contains 85 MB of CNN.com news data; one contains 9 MB of data from news.yahoo.com; and one contains a travelogue and a photo gallery that totals 139 MB. There are eight Pnapster mailboxes, which contain a total of 48 messages, including mp3 files (which average 4 MB each) and one 500 MB avi movie. In all, the P-disk contains 152 messages, for a total size of 2.35 GB.

• *Basic operations.* Our P-router appends bulk ISO image data to DVD+RW P-disks at 4.7 MB/s, and reads bulk data from them at 3.17 MB/s. Sending a small entry, which is only written to the local staging disk, takes about 3 ms on average, while reading a small entry from a DVD+RW P-disk costs about 40 ms.

• *Quick startup.* When a P-disk arrives, it is important that the receiver applications (which are all interactive, in the case of our three example applications) can quickly access summary information so they can make their application-specific decisions about what to do with the incoming data, without being forced to wait for time-consuming mandatory system-level data copying to complete. Table 2 compares three cases. In "Case 1," the applications are given access to the data only after a P-router copier copies all the data from the incoming P-disk to a local disk. In "Case 2," the P-router iterates through all the entry attributes, passing each attribute to a null application callback function. In "Case 3," the P-router only iterates through all the mailboxes, passing only the per-mailbox summary information to a null application callback function. This experiment shows the importance of structuring the P-router and its applications in a way that can avoid mandatory copying or scanning of large-capacity P-disks upon their arrival.

• *Exploiting knowledge of physical storage organization by the background copier.* Although it is important to allow applications to flexibly read from an incoming P-disk, a generic system-level P-disk copier may be able to function more efficiently by (1) exploiting knowledge of the physical storage organization (such as data locality) that applications are either unaware of or are unwilling to exploit due to complexity; and/or (2) performing more efficient scheduling across multiple applications. As applications dedicate the data movement tasks to such an efficient system-level background copier when possible, we may be able to drain data from incoming P-disks more quickly. In our prototype, the system background copier is able to exploit its knowledge of the ISO

	Pemail BW (MB/s)	Copier BW (MB/s)	DVD BW (MB/s)
Pemail alone	2.97		2.97
Pemail & dumb copier	0.17	0.17	0.34
Pemail & smart copier	2.73	2.34	2.66

Table 4: Potential impact of copier interference.

Applications	Time (s)
Pnapster	4.3
PwebCache1	6.6
PwebCache2	1.4
Pnapster & PwebCache1	100.3
Pnapster & PwebCache2	26.0

Table 5: Inter-application interference.

file system format, which clusters metadata in such a way that causes a naive recursive copier to suffer significant performance penalty. Table 3 shows an experiment of draining the PwebCache data from an incoming P-disk: the intelligent system-level P-router copier performs much more efficiently than a naive application-level recursive copier.

• *Cooperation between applications and the system-level copier.* We have argued above that both application-driven reads and system-level copiers are useful for efficiently draining incoming P-disks. Their co-existence requires their cooperation; and this cooperation takes two forms. First, when an application decides to discard incoming data without reading it, the system copier should (obviously) avoid copying it. An application that proactively "helps" the system in this way ends up improving the P-disk draining time of the entire system. Second, the system-level copier must exercise care not to compete against application-initiated reads. In the example of Table 4, as an email application retrieves a large attachment, the nature of the DVD media is such that an overzealous competing system copier ends up reducing the aggregate bandwidth by a factor of nearly 10.

• *Cooperation among applications.* We have discussed the interaction between an application and the system copier when processing an incoming P-disk. We now examine interactions among applications. Although the applications are given complete control of their reads from P-disks, as observed in Section 4.3, it is important that they read what is minimally necessary and leave the rest to the system copier. Overzealous applications that "prefetch" a large amount of data from a P-disk on their own, for example, may end up harming all applications, including themselves. We consider a simple example in Table 5. "Pnapster" retrieves a movie trailer (13 MB) from a DVD P-disk; "PwebCache1" retrieves the entire business subsection of CNN (321 entries, 8.8 MB); and "PwebCache2" recursively retrieves all the attributes under news.yahoo.com (341 entries). When multiple of these applications are active simultaneously, we consider the time it takes all of them to finish. Again, the nature of the DVD media is such that significant interference among applications may result if they are too eager reading P-disks. Each of these applications would have been better off only satisfying an interactive user's immediate needs and letting the system background copier move data out of the P-disk.

While the results in this section are based on DVDs, we believe they are generally important for two reasons. First, practically, DVD media is a very attractive P-disk candidate, and some of its fundamental characteristics (such as latency) are likely to be with us for some time. Second, even if we were to consider other types of movable media, such as IBM Microdrive-type disks, due to energy, noise, and size considerations, these storage devices are likely to share similar issues as DVDs, so the lessons that we have learned about getting the most of DVD P-disks may be more generally applicable.

7 Related Work

Gray and his colleagues have shipped via the postal system entire NFS servers filled with terabytes of astronomy data [8]. NFS servers are chosen as mobile storage devices to minimize the amount of manual configuration a data recipient would need to perform. This is a goal that we share. Our interest is in generalizing these tailor-made solutions for specialized applications into a generic communication mechanism that can benefit many applications. By itself, a local file system interface that grants application access to the mobile storage devices may be inadequate: for example, tasks such as recipients' sending back acknowledgements over the Internet should be automated away by a transport-level system. We also note that the applicability of the Postmanet approach is by no means limited to data-intensive scientific applications: we have discussed a variety of applications that can be useful for average users, especially those who fall on the wrong side of the digital divide.

Rover is a toolkit for constructing applications targeting weak and intermittent wireless networks [10]. A key element of the system is an asynchronous communication mechanism that allows applications running on mobile wireless clients to continue to function as communication with a remote server occurs in the background. The need of an asynchronous communication mechanism applies to the high-latency Postmanet. The characteristics of the postal system, however, are different from those of a weak wireless network: the postal system provides a high-latency high-bandwidth datagram-like service. By simultaneously exploiting an available low-latency low-bandwidth Internet connection and the excess capacity of movable storage media, we can provide better higher-level services.

Recent efforts on "Delay-Tolerant Networks" (DTNs) [6, 9,11,18] have started to examine the use of WiFi-enabled mobile elements (such as buses equipped with storage devices) to simulate "delayed" connectivity to places that have access to none today. While "postal classes of service" have been mentioned, to the best of our knowledge, the postal system has so far only been mentioned as an *analogy*—no existing DTN that we are aware of literally uses the postal system. There are several important differences between existing DTNs and the Postmanet. First, while existing DTNs are largely confined to relatively small regions or specialized environments, the postal system is a truly *global* "network" that reaches a far greater percentage of the world's *human* population without needing investment in exotic equipment. Ad hoc routing,

frequently a central focus of some DTNs, is not necessarily a top focus of the Postmanet. Instead, we are more concerned with somewhat less conventional routing metrics, such as the number of storage devices handled per site per postman visit.

Second, most existing DTNs are also frequently referred to as "challenged networks:" they may be limited by low bandwidth among mobile ad hoc elements, brief and/or intermittent contacts among these elements, small amounts of storage space on these nodes, and power consumption constraints. In contrast, the P-disks in the Postmanet are "dumb" and "dormant" during transit in the postal system. When they reach their destinations, they are "plugged in," quite possibly with high-bandwidth wired alternatives (such as USB2 or Firewire). Once such "contacts" are established, they may remain connected for extended periods of time. Instead of carefully conserving resources such as storage space and bandwidth, we may in fact strive to "waste" some of these abundant resources in order to gain other advantages. Another unique aspect of the Postmanet is the possible availability of a complimentary low-latency low-bandwidth Internet connection: the techniques involved in the *parallel* exploitation of multiple connectivity technologies are different from those involved in the *sequential* forwarding of data from one connectivity technology to another.

The PersonalRAID system leverages a single mobile storage device that always accompanies its owner to transport storage system differences across multiple computers for a single user [21]. The goal of these distributed mobile storage systems is to provide the illusion of a coherent *disk* or *file* system, while the goal of the Postmanet is to provide the illusion of a *network* connection—these are very different abstractions. The network abstraction is at a sufficiently low level that may allow potentially greater degree of application flexibility, while an important goal of typical distributed storage systems is to entirely abstract away device or machine identities. The question of how to build a distributed storage system on top of the Postmanet, however, is still an interesting one.

The de Bruijn interconnection topology has been used in parallel applications [3, 5, 17, 19] and distributed hash tables (DHTs) [12]. These DHT-based systems employ implicit routing wherein routing decisions are made locally without requiring elaborate knowledge of the global topology. We note, however, that implicit routing may be of limited value in Postmanet, where the control and data traffic can be conveyed on different networks—the LLLB Internet could be used for dispersing topology information or topology repairs, while bulk data is communicated over the HLHB channels. In absence of an LLLB channel, however, implicit routing may again become important. A problem that has not been considered by both the parallel computing and the DHT communities is how to construct a de Bruijn graph in a geography-aware fashion for systems where communication between different pairs of nodes incurs different amount of latencies. We have devised geography-aware de Bruijn topologies for use in the Postmanet.

8 Conclusion

In this paper, we have described how to turn storage media transported by the postal system into a generic high-bandwidth digital communication mechanism. We believe that this approach can enable a variety of interesting bandwidth-intensive applications; and it presents an unconventional but promising approach to addressing the digital divide.

References

[1] ANDERSON, R. 'Trusted Computing' Frequently Asked Questions. http://-www.cl.cam.ac.uk/~rja14/tcpa-faq.html, August 2003.

[2] AVRAMOPOULOS, I., KOBAYASHI, H., WANG, R., AND KRISHNAMURTHY, A. Highly secure and efficient routing. *Proc. IEEE INFOCOM* (March 2004).

[3] BERMOND, J.-C., AND FRAIGNIAUD, P. Broadcasting and Gossiping in de Bruijn Networks. *SIAM Journal on Computing 23*, 1 (1994), 212–225.

[4] DE BRUIJN, N. A Combinatorial Problem. In *Proc. Koninklijke Nederlandse Akademie van Wetenschappen* (1946), vol. 49, pp. 758–764.

[5] ESFAHANIAN, A., AND HAKIMI, S. Fault–Tolerant Routing in de Bruijn Communication Networks. *IEEE Trans on. Computers 34*, 9 (1985), 777–788.

[6] FALL, K. A delay tolerant networking architecture for challenged internets. In *Proc. of ACM SIGCOMM 2003* (August 2003).

[7] GOLDBERG, A. Network Optimization Library. http://www.avglab.com/andrew/soft.html.

[8] GRAY, J., AND PATTERSON, D. A Conversation with Jim Gray. *ACM Queue 1*, 4 (June 2003).

[9] HASSON, A. A., FLETCHER, R., AND PENTLAND, A. DakNet: A Road To Universal Broadband Connectivity. http://courses.media.mit.edu/2003fall/de/-DakNet-Case.pdf, 2003.

[10] JOSEPH, A. D., DELESPINASSE, A. F., TAUBER, J. A., GIFFORD, D. K., AND KAASHOEK, M. F. Rover: A Toolkit for Mobile Information Access. In *Proc. the 15th ACM Symposium on Operating Systems Principles* (December 1995), pp. 156–171.

[11] JUANG, P., OKI, H., WANG, Y., MARTONOSI, M., PEH, L.-S., AND RUBENSTEIN, D. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *The Tenth International Conference on Architectural Support for Programming Languages and Operating Systems* (October 2002).

[12] KAASHOEK, M. F., AND KARGER, D. R. Koorde: A simple degree-optimal distributed hash table. In *Proc. of Intl. Workshop on Peer-to-Peer Systems* (2003).

[13] LAI, J., ZISKIND, E., ZHENG, F., SHAO, Y., ZHANG, C., ZHANG, M., GARG, N., SOBTI, S., WANG, R., AND KRISHNAMURTHY, A. Distance learning technologies for basic education in disadvantaged areas. *The Eighth Global Chinese Conference on Computers in Education* (June 2004).

[14] OPENLDAP FOUNDATION. Openldap 2.1. http://www.openldap.org.

[15] The Postman Always Rings Twice. http://www.cs.princeton.edu/~rywang/-distance, 2004.

[16] ROSENBLUM, M., AND OUSTERHOUT, J. The Design and Implementation of a Log-Structured File System. In *Proc. of the 13th Symposium on Operating Systems Principles* (Oct. 1991), pp. 1–15.

[17] SAMATHAM, M., AND PRADHAN, D. The de Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI. *IEEE Trans on. Computers 38*, 4 (1989), 567–581.

[18] SHAH, R., ROY, S., JAIN, S., AND BRUNETTE, W. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop* (May 2003).

[19] SIVARAJAN, K., AND RAMASWAMI, R. Multihop Lightwave Networks based on de Bruijn Graphs. In *Proc. INFOCOMM* (1992), pp. 1001–1011.

[20] SLEEPYCAT SOFTWARE. Berkeley db 4.2. http://www.sleepycat.com.

[21] SOBTI, S., GARG, N., ZHANG, C., YU, X., KRISHNAMURTHY, A., AND WANG, R. Y. PersonalRAID: Mobile Storage for Distributed and Disconnected Computers. In *Proc. First Conference on File and Storage Technologies* (January 2002).

[22] WANG, R., LI, K., MARTONOSI, M., AND KRISHNAMURTHY, A. Distance Learning Technologies for Basic Education in Disadvantaged Areas. Tech. Rep. TR-685-03, Computer Science Department, Princeton University, November 2003.

[23] WANG, R. Y., GARG, N., SOBTI, S., LAI, J., ZISKIND, E., ZHENG, F., NAKAO, A., AND KRISHNAMURTHY, A. Postmanet: Turning the Postal System into a Generic Digital Communication Mechanism. In *Proc. ACM SIGCOMM 2004* (August 2004).

[24] WANG, R. Y., GARG, N., SOBTI, S., LAI, J., ZISKIND, E., ZHENG, F., NAKAO, A., AND KRISHNAMURTHY, A. Postmanet: Turning the Postal System into a Generic Digital Communication Mechanism. Tech. Rep. TR-691-04, Computer Science Department, Princeton University, February 2004.