A Delay-Tolerant Network Architecture for Challenged Internets

Kevin Fall

IRB-TR-03-003

February, 2003

Intel Research Berkeley

A Delay-Tolerant Network Architecture for Challenged Internets

[Kevin Fall -- Intel Research, Berkeley]

***Abstract*—The highly successful architecture and supporting protocols of today's Internet operate poorly when faced with operating environments characterized by very long delay paths and frequent network partitions. These problems are exacerbated by end nodes that have severe power or memory constraints. Often deployed in mobile and extreme environments lacking "always-on" infrastructure, many such networks have their own specialized protocols, and do not utilize IP. To achieve interoperability between them, we propose a network architecture and application interface structured around optionally-reliable asynchronous message forwarding, with limited expectations of end-to-end connectivity and node resources. The architecture operates as an overlay above the transport layers of the networks it interconnects, and provides key services such as in-network data storage and retransmission, interoperable naming, authenticated forwarding and a coarse-grained class of service.**

I. Introduction

The existing TCP/IP based Internet operates on a principle of providing end-to-end inter-process communication using a concatenation of potentially dissimilar link-layer technologies. The standardization of the IP protocol and its mapping into network-specific link-layer data frames at each router as required supports interoperability using a packet-switched model of service. Although often not explicitly stated, a number of key assumptions are made regarding the overall performance characteristics of the underlying links in order to achieve smooth operation: an end-to-end path exists between a data source and its peer(s), the maximum round-trip time between any node pairs in the network is not excessive, and the end-to-end packet drop probability is small. Unfortunately, a class of so-called *challenged networks*, which may violate one or more of the assumptions, are becoming important and may not be well served by the current end-to-end TCP/IP model.

Challenged networks arise primarily as a result of various forms of host and router mobility, but may also come into being as a result of disconnection due to power management or interference. Examples of such networks include:

- Terrestrial Mobile Networks: In many cases, these networks may become unexpectedly partitioned due to node mobility or RF interference. In other cases, the network may never have an end-to-end path and may be expected to be partitioned in a periodic, predictable manner. For example, imagine a commuter bus acting as a store and forward message switch with only limited RF communication capability. As it travels from place to place, it provides a form of message switching service to its nearby clients to communicate with distant parties it will visit in the future.
- Exotic Media Networks: Exotic communication media includes near-Earth satellite communications, very long-distance radio links (e.g. deep space RF communications with light propagation delays in the seconds or minutes), communication using acoustic modulation in air or water, and some free-space optical communications. These systems may be subject to high latencies with predictable interruption (e.g. due to planetary dynamics or the passing of a scheduled ship), may suffer outage due to environmental conditions (e.g. weather), or may provide a predictably-available store-and-forward network service that is only occasionally available (e.g. low-earth orbiting satellites that "pass" by one or more times each day).
- Military Ad-Hoc Networks: These systems may be expected to operate in hostile environments where mobile nodes, environmental factors, or intentional jamming may be cause for disconnection. In addition, data traffic on these networks may have to compete for bandwidth with other services at higher priority. As an example, data traffic may have to unexpectedly wait several seconds or more while high-priority voice traffic is carried on the same underlying links. Such systems also may also have especially strong infrastructure protection requirements.
- Sensor and Sensor/Actuator Networks: These networks are frequently characterized by extremely limited end-node power, memory, and CPU capability. In addition, they are envisioned to exist at tremendous scale, with possibly thousands or millions of nodes per network. Communication within these networks is often *scheduled* to conserve power, and sets of nodes are frequently named (or addressed) only in aggregate. They are often interfaced to other networks by way of one or more "proxy" nodes that provide protocol translation capabilities.

Given the large accumulated experience and number of systems compatible with the TCP/IP protocols, it is natural to apply the highly successful Internet architectural concepts to these new or unusual types of networks. While such an application is conceivable, the effects of very significant link delay, non-existence of end-to-end routing paths, and lack of continuous power or large memory at end nodes present substantial operational and performance challenges to such an approach.

In an effort to adapt Internet to unusual environments, one class of appoaches attempts to engineer problem links to appear more similar to the types of links for which TCP/IP was designed. In effect, these approaches, which we term *link-repair approaches*, "fool" the Internet protocols into believing they are operating over a comparatively well-performing physical infrastructure. They strive to maintain the end-to-end reliability and fate sharing model of Internet, and generally require the use of IP in all participating routers and end nodes.

Another common approach to deal with challenged networks is to attach them to the *edge* of the Internet only by means of a special proxy agent. This provides access to and

from challenged networks from the Internet, but does not provide a general way to use such networks for data *transit*. Without supporting transit, the full capabilities of these networks will go under utilized. Indeed, supporting transit is often of particular interest because remotely-deployed conventional networks (e.g. Intranets) may only be accessible through challenged intermediate networks.

In this paper, we argue that to achieve interoperability between some networks, especially those engineered for extreme environments or that often suffer from frequent network partitioning, link-repair approaches alone will not suffice and network-specific proxies are undesirable. Instead, we suggest a general purpose message-oriented reliable overlay architecture as the appropriate approach to tie together such networks, forming an "internetwork of challenged internets." The approach, which provides the service semantics of asynchronous message delivery, may be used in combination with TCP/IP where appropriate. Its design is influenced by the interoperability properties of the classical Internet design, the robust non-interactive delivery semantics of electronic mail, and a subset of the classes of service provided by the US Postal System. These networks have all evolved to become highly successful communication networks supporting millions of daily users.

II. Characteristics of Challenged Networks

Qualitatively, challenged internetworks are characterized by latency, bandwidth limitations, error probability, node longevity, or path stability that are substantially worse than is typical of today's Internet. We use the Internet's performance as a baseline due to its enormous scale and influence. This section explores the path properties, network architectures and end node resources found across the broad range of challenge networks introduced above and how they influence the design of a network architecture designed to accommodate them.

A. Path and Link Characteristics

High Latency, Low Data Rate: End-to-end path latency represents the sum of one-way delivery delays at each hop and comprises transmission/processing/propagation time across each link, plus any queuing delay experienced during forwarding. If we temporarily disregard processing and queuing delays (we return to queuing delays shortly), the transmission and propagation delays are directly affected by the underlying transmission medium. For challenged networks, transmission rates may be comparatively small (e.g. about 10kbps for underwater acoustic modems and low-power radios in sensor nodes) and latencies may be comparatively large (to about a second or two). Finally, data rates may be largely asymmetric (e.g. remote instruments may have a comparatively large return channel for relaying data but a small uplink used for device control). In some extreme cases, no return channel at all may be available (e.g. communication with some military assets requiring covert operation such as submarines).[1]

For efficient operation over networks with potentially high latency and low data rates, protocols should be as terse as possible. In particular, the fewest number of round-trip exchanges necessary to accomplish a protocol transaction should be chosen above "chatty" exchanges. Doing so helps to reduce turnaround time and conserve bandwidth. In addition, dynamic control functions within protocols (e.g. for sending rate adaptation) should be performed open-loop or hop-by-hop rather than end-to-end due to the lack of timely end-to-end feedback.

For networks which lack any return channel at all, any form of reliability will be at best be probabilistic. The most common techniques to deal with such systems include *data carousels* which periodically transmit copies of data to be sent. Recent developments in erasure coding [BLMR98] suggest that data carousels can efficiently utilize such codes so that receivers can reconstruct the entire message from any sufficient number of fragments, provided they were independently generated. This technique is especially useful for one-to-many applications. With respect to network architecture design, such experience suggests that this alternative form of reliable delivery should be contemplated when considering the mechanisms required to support an applications' request for reliable transfer.

Disconnection. In many challenged networks, end-to-end disconnection may be more common than connection. Generally speaking, disconnection may be broadly categorized as due to a fault or not. Faults have been studied extensively for conventional networks, and will not be further considered here. Non-faulty disconnections arise most frequently in wireless environments, from primarily two sources: motion and low-duty-cycle system operation. Disconnection due to motion may be highly predictable (e.g. satellite passes, busses that act as data routers, etc) or opportunistic (nodes arrive in communication range due to random walk) and may arise due to motion of either end-nodes, routers, or some other object or signal that obscures the communication. Disconnection due to low-duty-cycle operation is common among low-capability devices (e.g. sensor networks), and is often predictable. An exceptional condition requiring immediate attention (event response) can perturb the otherwise periodic low-duty-cycle operation.

Coping with disconnection requires the routing subsystem to understand that the lack of reachability between two or more nodes may be the result of a normal situation, and should not necessarily be considered an outage due to fault. In addition, the possibility that such outages may be known ahead of time with high accuracy suggests the routing system may be able to pre-schedule the times at which messages should be sent. Such scheduling could be more easily performed with some type of priority or class of service indicator on messages.

Long Queuing Times. For multi-hop paths in conventional packet networks with statistical multiplexing, the queuing time often dominates the transmission and propagation delays. Queuing time rarely exceeds a second (typically much less) and packets are discarded at routers if next-hop neighbors are not instantaneously reachable. In contrast, for challenged networks where disconnection may be common, the queuing time could be extremely large by comparison (hours, perhaps days). This suggests messages must be stored for potentially long periods of time at routing points, and also suggests that next-hop selection may need to be *revoked*. That is, messages should be able to be forwarded to alternative next hops if better routes are discovered prior to message transmission.

1. There are also some *very* extreme examples of long latency and low bit rates. The NASA Deep Space Network (DSN) continues to monitor Voyager 1, with a round-trip light time of about 23 hours. The US Navy's *Project ELF* utilizes extremely low frequency communications (in the 30-300Hz frequency range using 18-24 mile long antennas), achieving a purported bit rate of roughly 1 bit per minute. It is not likely, however, that such very extreme networks will be practical for internetworking in the foreseeable future.

B. Network Architectures

Interoperability Considerations In most challenged networks, the network "architectures" consist primarily of a link and media-access control protocol, and are not designed with interoperability (or very large scale) in mind. The reason for this is that in many cases, merely communicating *at all* over some links is still an active area for research, and the desire to use such links in an internetwork has not yet become a primary focus. Thus, these networks tend to be comparatively simple and local in scope, and may fail to provide even the baseline layering and abstractions that are well-matched for supporting Internet protocols. Implementations frequently "cut corners" when targeted for deployment on memory and power-limited devices, mixing together data from various system functional blocks into messages that are difficult to dis-aggregate.

Without interoperability considerations, networks designed for challenged environments may use application-specific framing formats, limited node addressing and naming capabilities, data packet size restrictions (because an assumed "best" size may have been chosen for some particular application), etc. They may also fail to implement reliability, congestion control, and security. In attempting to interoperate with and through such networks, a network design must make minimal assumptions of the underlying protocol stack capabilities, and yet be sufficiently modular to easily augment them where necessary to establish some baseline functionality.

Security. In challenged networks where communication media is exotic and possibly oversubscribed, link capacity is a precious resource and access to the "service" of data forwarding should be protected by some authentication and access control mechanism, at least at critical points in the topology. If multiple classes of service are available, some mechanism to enforce an access control matrix to CoS is also likely to be required.

For delay-tolerant networks with precious link resources, an end-to-end approach to security is not very attractive, stemming from two issues. First, end-to-end approaches typically require some end-to-end exchange of challenges or keys, which would be undesirable for reasons already discussed. Secondly, it is undesirable to carry unwanted traffic all the way to its destination before an authentication and access control check is performed. This problem has been (and remains) a significant one for the Internet, as can be seen by the rich collection of approaches that attempt to trace problem traffic all the way back to its source.

C. End System Characteristics

Limited Longevity. In some challenged networks, end nodes are placed in hostile environments. This is especially true for sensor networks, military networks, and networks of devices used by emergency response personnel. In such cases, network nodes may not be expected to last long, due to environmental dangers or power exhaustion. If such networks also remain disconnected for long periods of time, it is entirely possible that the round-trip or even one-way delivery time of a particular message may exceed the sending node's lifetime.

Clearly, in such cases it is useless to utilize conventional acknowledgment schemes to help verify delivery. Rather, the responsibility for reliable delivery should be *delegated* to some other party, and any notification of successful or unsuccessful delivery needs to be delivered to a delegate that remains functional.

Low Duty Cycle Operation. When nodes are deployed in areas lacking power infrastructure, they often run off batteries (which may be augmented with some charging capability such as solar cells). Even if charging is available, these systems typically attempt to save power by limiting their duty cycle. In some cases (e.g. battery powered sensors), duty cycles of well under 1% are desirable in order to achieve a reasonable longevity of the entire network. Such devices would typically collect data at some periodic rate, and report it at some (perhaps less frequent) rate.

For such systems, uptime (and data communication) is generally scheduled ahead of time. As such, it represents a form of disconnection that affects the operation of routing protocols as described above (i.e. protocols should understand periodic disconnection). Of course, discovering new nodes that join the network during an idle period remains a challenge, but several proposals address this problem, including special-purpose extremely low power *wake-up radios* [GZR01] that are able to monitor communications and wake up a primary radio when data arrives.

Limited Resources In several of the challenged network examples above, nodes with limited memory and processing capability are used. Consider, for example, an instrument with limited memory tasked with acquiring sensor readings of some random physical phenomena. It is undesirable to prohibit the instrument from collecting further samples because its memory is fully utilized with copies of in-transit data. In addition, the amount of time the end-nodes will need to keep retransmission buffers is at least the round-trip-time times the expected number of retransmissions (plus 1), which can be large for high latency and/or lossy paths. While the node may be able to implement a powered down mode of operation during this interval, provided it has nonvolatile storage, doing so considerably complicates the system design, particularly if other asynchronous messages may have to be received or unexpected physical events of interest occur. This example suggests that if reliability is to be incorporated into the network design, end nodes should be provided a way to empty their retransmission buffers comparatively quickly, and to not necessarily have to wait for an end-to-end acknowledgment.

III. Can We Use TCP/IP for Challenged Networks?

The Internet architecture achieves interoperability by mandating the mapping of certain fundamental objects (e.g. IP packet format, domain names, etc.) onto existing physical networks. Nodes linking two different types of networks must implement Internet packet forwarding, address mapping, and usually some Internet-standard routing protocol. The fundamental service provided is fully-connected best-effort unicast datagram transport, making the mapping relatively straightforward for most continuously-connected packet networks with a moderate packet size and reasonably small packet drop rate. Experience has shown that difficulties with the mapping arise when either the nature of the physical links are changed radically (e.g. wireless or ATM), or the Internet service offering is enhanced significantly (e.g. IP multicast, network-layer security, and Quality of Service).

The types of networks falling in the challenged category presented here are generally those with performance characteristics and network architectures that deviate significantly from the Internet. The most obvious options available to interconnect such networks include modification or enhancements to Internet protocols for direct use by challenged networks, or constructing a set of proxies to form an overlay that somehow utilizes Internet protocols where appropriate, but also supports the use of other protocol families more appropriately matched to challenged environments. The first question to answer is whether the popular Internet protocols and application interfaces (or modest modifications to them) could be used straightforwardly as the solution. We first examine the operation of several specific Internet protocols without enhancements, followed by effects of link-repair methods and the use of proxies.

A. The Internet's Most Common Protocols

The performance characteristics of challenged networks contribute to confound the efficient operation of the common Internet protocols. By the common Internet protocols we mean the system-oriented protocols: IP, TCP, UDP, BGP, common IGPs (RIP, OSPF, or EIGRP), DNS and the recently-specified SCTP, plus the user-oriented applications and supporting protocols: HTTP, various peer-to-peer sharing protocols, SMTP, POP, IMAP, Microsoft's Exchange Protocol and FTP. These protocols provide a wide range of services, including end-to-end datagram delivery, reliable two-party stream and message delivery, regional (aggregated) routing path discovery with policy, intra-domain path selection, distributed support for name resolution and data sharing. This section explores the impact of challenged networks' performance of several of these protocols.

Transport Layer (TCP, SCTP,UDP): High latency affects TCP directly by severely limiting its throughput performance or interfering with connection establishment [DMT96]. SCTP is affected at least as badly, because it adopts a TCP-friendly congestion response behavior and involves a connection establishment phase requiring more round-trip exchanges. Significant path loss also affects the TCP and SCTP transports strongly, causing them a number of problems. After multiple loss events they continue retrying with a backed-off retransmission timer until giving up on retransmission altogether and terminating a connection. Somewhat more moderate losses will contribute to problems invoking the fast retransmit and recovery algorithms, even in the presence of selective acknowledgements.

A less fundamental problem relates to the maximum segment lifetime (MSL). In the original TCP specification [RFC793], this is taken to be 2 minutes. After such a time, it is assumed no TCP segment could still exist within the Internet. Clearly, for networks that may be disconnected for periods of time, such an assumption could easily be violated.

Network Layer (IP): Although comparatively rare, IP performance can be affected by path loss if fragmentation is required. IP provides no mechanism for fragment retransmission, thereby causing the overall probability of successful datagram delivery to be further reduced if datagrams are fragmented [M95]. If comparatively small frame sizes are used (typical for lossy networks), fragmentation may be more frequent, causing the loss of fragments to become a larger problem. In addition, the IP protocol has a time-to-live field, which nominally is expressed in units of seconds with a maximum value of 255. Although this is now almost universally treated as a hop count due to the requirement that any forwarding router decrement an IP packet's TTL by at least one (and a queuing time of more than one second is essentially unheard of), in networks where end-to-end path delay could conceivably exceed 255 seconds, a problem of perpetual packet discarding could arise.

Application Layer (Routing): High latency adversely affects the proper operation of conventional routing protocols, as links will be incorrectly labeled as non-operating when soft-state refreshes are delayed too long (RIP, BGP), or a lack of response to link state "hello" messages (OSPF) is observed. Disconnection is completely at-odds with these protocols, as their objective is to compute paths in order to supply the abstraction of a fully connected graph at any end system.

Application Layer (Others): Any application-layer protocol using one of the reliable protocols as its underlying transport is negatively affected as mentioned above (e.g. BGP, DNS zone transfers, SMTP, etc.). While UDP is not sensitive to excessive latency because it does not contain timers that affect its operation, core application protocols that require it (DNS queries/responses and SNMP) often do, and may prematurely abort when faced with excessive delay, triggering failure (or the lack of ability to use DNS name/address mappings in the case of address-to-name queries). While application-level timeouts could conceivably be hand-tuned for communicating with particular challenged networks, this approach is fragile because in many circumstances the expected query-response time may exceed a node's (or application's) longevity, or the expected transaction round-trip time cannot be known with any degree of certainty in advance.

B. Application Development Practice

Developing network-based applications has always been a more challenging undertaking than developing applications for local execution. To lessen the burden on the programmer, network access is often "hidden" below some other common abstraction (e.g. remote procedure calls or method invocation, remote file system access, etc). These systems are designed with a certain general understanding of network performance in mind. Challenged networks stress these assumptions, and reveal a number of the problems detailed below. While one cannot fault these systems for not contemplating challenged networks, in order to operate over such networks, some change in programming methodology would be needed. These changes are not extremely radical, and carefully-designed application libraries could help to minimize the burden on the programmer.

The four most common problems found with applications (and application programming libraries) when deployed over challenged networks are: application-level timeouts mismatched to the actual experienced latency, lack of automatic failover, a programming style that assumes application process execution is relatively long compared to transaction duration, and use of "chatty" application protocols requiring a larger than necessary number of round-trip request/response protocol message exchanges. The first three issues can lead to complete failure, while the last one generally leads to degraded performance.

Timeouts. Application-level timeouts are typically used to initiate a transaction retry, or to inform a user that some

requested transaction cannot be completed (in the time provided). For example, web browsers and peer-to-peer agents will often time out during connection establishment after a minute or two (or less), awaiting a response from their peers. These applications make the assumption that a connection should be able to be established quickly, and that user-initiated retry is a fallback option on failure.

Lack of Failover. With the important exception of recent peer-to-peer systems, applications typically contain no automated routing function, so any attempt at failover requires manual reconfiguration or retry. For the peer-to-peer overlay systems which *do* contain routing capabilities, route selection may be far from optimal at the network layer due to separation of topology information in the overlay from topology at lower layers. While this problem is being addressed, most of these systems are constructed to provide a distributed searching or storage capability, and are not focused on interoperability or providing data transit to pairs of communicating parties. A notable exception is the Resilient Overlay Networks [ABKM01] project, which does provide a form of message delivery capability between users, but using an Internet-like message delivery semantic with fully-connected path abstraction.

Synchronous Programming Style. The assumption that connection establishment and duration will be relatively short-lived compared to the execution time of an application is implicit in the design of the most common network programming interface— sockets. In particular, the port binding functions provided by the `bind()` call are only persistent for at most the duration of the application. In addition, most of the supporting functions (connect, accept, etc.) are synchronous calls by default, and similarly persist at most as long as the application is in execution. While asynchronous primitives or extensions also exist, they are less frequently used by programmers and do not generally contemplate network transactions that may last longer than the uptime of the requesting application or host system.

Chatty Application Protocols. When end-to-end round-trip times are assumed to be small, using multiple client/server exchanges to establish a communication association presents few problems. When the client/server round-trip-time is large, however, end-systems must be concerned with buffering data for long periods of time and network bandwidth may be inefficiently used. As an example, using FTP will typically require at least 6 round-trip times: two for TCP connection establishment and tear-down, one for the server to prompt for and receive a name, one for a password, one to change transfer mode, and one to request a file put or get operation.

To encourage the development of applications capable of operating over challenged networks, a combination of developer education and a carefully-constructed network API are required. A properly-constructed network library would free the programmer from constructing application level timeouts by providing applications the ability to be long lived: simple methods for snapshot, shutdown and restart. It would also encourage the use of asynchronous messaging semantics by requiring registration of callbacks to handle incoming messages. Finally, such an interface would easily take advantage of DTN's late binding of name tuples (see SectionIV.B.), allowing a name to be mapped to a different physical destination on failure. Chatty application protocols could still be created by programmers, and avoiding them would be a subject of education regarding application development methods.

C. Use of Proxies and "Protocol Boosters"

To combat the various problems with the Internet protocols over challenged networks (or to enhance their performance over subnetworks with special features), several types of in-network entities have been developed that modify protocol behavior. Investigations of link repair approaches, primarily for satellite and terrestrial wireless Internet access via TCP/IP, have resulted in the development of Performance Enhancing Proxies (PEPs) [RFC3135] and so-called *protocol boosters* [FMS98]. These agents, which actively modify the end-to-end data stream, in effect "fool" TCP/IP-based end stations into operating more efficiently over paths involving links with poor or unusual performance characteristics. They generally operate at the transport or application layers (or some combination of the two).

These approaches most frequently elicit particular end station behavior by actively modifying the TCP data stream contents or timing relationship. Approaches range from local connection termination, modification of the TCP ACK stream, retransmitting lost TCP packets without end-system interaction, and general data stream modification (e.g. compression or encryption). The approaches differ primarily in their respective levels of transparency. That is, to what extent they modify end-to-end semantics. Generally, more transparency is preferred to less, with the overall trade-off being between transparency and degree of performance improvement.

Use of PEPs is discouraged (by the IETF) except for particular environments where they are necessary for reasonable performance. This restriction is due to their fragility. In particular, they may contain state which is necessary for connection operation (thereby violating the Internet fate sharing principles which suggest connection state should reside only in end stations), they confound end-to-end diagnostics and reliability by (partially) changing the communicating endpoint, they significantly increase system complexity if mobility is frequent (due to the need to migrate state when end-nodes move), many require both directions of data to flow through the PEP, and thus fail in the face of asymmetric routing. They also pose a significant challenge for end-to-end security mechanisms implemented below the transport layer such as IPSEC[RFC3135]. While protocol boosters are conceived with the idea that they are entirely transparent to end-protocols, this assumption limits their overall ability to improve performance when subnet conditions are especially bad (e.g. disconnected).

An alternative to boosters and PEPs involves application-layer proxies that provide a specialized Internet-to-special network name mapping and protocol translation. Proxies are generally used at the edge of such special networks, and allow interoperability with the Internet without requiring IP routers to exist inside. This approach is important, as there is often significant reluctance to deploy IP protocols inside these challenged networks due to its overhead and mismatch with the performance of the physical network links.

The disadvantage of the proxy approach is in its specificity. Proxies usually use one of two approaches: they respond to a specialized set of commands, or act merely as raw data connectors. The first approach limits the ability to re-use the proxies

for different networks; the second method fails to take advantage of any special resources the proxy node may have to offer (such as memory or processing capabilities), and requires applications communicating with the proxy to employ specialized code on a per-network basis. It would be generally more attractive to standardize on a set of proxy-based services which provide I/O to and *through* the challenged network using a common set of methods.

D. Can We Use Electronic Mail?

Electronic mail, an asynchronous type of message delivery system, comes close to addressing the problems posed by the need for delay tolerance and flexible name/addressing semantics, and had operated over a rich set of network technologies (especially prior to the widespread use of the Internet). Email falls short most fundamentally due to its lack of dynamic routing and weakly-defined delivery semantics. Email generally, and the Internet's SMTP protocol particularly, makes use of a statically-defined set of mail exchanger pointers that provide a limited method for using alternative mail drops to deliver mail. Furthermore, the mail architecture as currently conceived does not accommodate a set of dynamically-discovered intermediate mail redistribution points (message routers). Thus, if an especially lossy path (or no path) exists between a sender's SMTP agent and a receivers mail spool, TCP will be unable to make any significant progress in delivering e-mail toward its destination.

Another particular problem with the SMTP protocol is its startup sequence; it is a chatty protocol. At the beginning of an SMTP-based e-mail exchange, each peer must mutually identify itself prior to actually exchanging the mail payload, even though accomplishing this name exchange early is not fundamental to the process of delivering e-mail. At least six round-trips are generally required: 2 for TCP open and close, one for the HELO exchange, one for the RCPT TO exchange, one for the MAIL FROM exchange, and one for the actual data transfer. Clearly, such a protocol could be made more efficient by collecting together the meta-data at connection establishment.

The delivery semantics of electronic mail appear to be *mostly reliable* delivery with *likely* failure notification. Electronic mail can fail to be delivered due to mis-addressing, persistent lack of intermediate or end-node storage, failure of underlying transport protocols or enforcement of policies on content (e.g. content filtering or size restrictions). When delivery succeeds, end-to-end acknowledgments are generally not provided. Upon failure, the original message and accumulated errors are generally returned to the sender, possibly with additional information supplied to a third party. While this diagnostic information is extremely useful, the end-user typically has little ability to influence to whom it is directed.

Although un-enhanced electronic mail fails to completely solve connectivity and interoperability problems across challenged networks, it does provide a significant number of useful features. In particular, flexible naming, asynchronous message-based operation, and in-band error reporting are particularly useful.

E. Discussion

In consideration of using the existing Internet protocols as a basis for interconnecting challenged networks, we encounter problems due to a mismatch of the fundamental abstraction provided by the Internet architecture, in combination with common application development practices. In particular, certain properties of the operational environment in which these networks exist seem insurmountable. Disconnected paths, limited-capability/longevity end devices with potentially specialized protocol stacks, and unusual routing (including predictable or periodic connections) appears to preclude the use of conventional reliable transports such as TCP which depend on end-to-end retransmissions for reliability because a complete forward or reverse may not ever be available. Even with PEPs or boosters, which can improve performance for various circumstances, the end-to-end continuous connectivity Internet service model seems poorly matched to the types of operating environments discussed here, and suggests some alternatives should be explored.

Most of the problems outlined above are a consequence of a reliable data delivery scheme based on end-to-end retransmission, which is tied closely to Internet's idea of *fate sharing*. Fate sharing suggests that per-connection state should remain only in end-stations, because a failure of one of them would presumably render the data connection essentially useless. In many challenged environments, some of the base operating assumptions are different. For example, it may be quite useful to allow a node to "hand off" its end-node connection state to another node if it has other tasks to accomplish, particularly if it is power or memory limited. Doing so would not violate fate sharing entirely (per connection state would not be required in all intermediate nodes), but would represent a somewhat different fate sharing behavior than is implemented in the current Internet.

Using the proxy approach, but without a common protocol to interconnect them, introduces a number of disadvantages as discussed above. In addition, deploying proxies that lack a routing function fails to provide a general-purpose network communication capability and is unlikely to support transit through challenged networks. In particular, without routing support, proxies only provide protocol translation and their existence must be discovered out of band. Without some common architecture for them, no communication service can be built consistently, as no common set of underlying system capabilities can be assumed. For these reasons, we believe a new standard architecture and set of protocol features, able to operate under poor conditions such as frequent disconnection and long latencies, is required for any approach based on proxies.

Given the assumptions, the most desirable type of framework for supporting challenged networks would appear to be a network service providing a sort of least common denominator interface: non-interactive messaging. Based on experience with the Internet, we conclude the system should combine some overlay routing capability such as is present in peer-to-peer systems with the delay-tolerant and disconnection-tolerant properties of electronic mail. If implemented at the application layer (in the form of a proxy), such a system could conceivably provide a gateway function between dissimilar networks. These together motivate the articulation of a new architecture, which we now discuss.

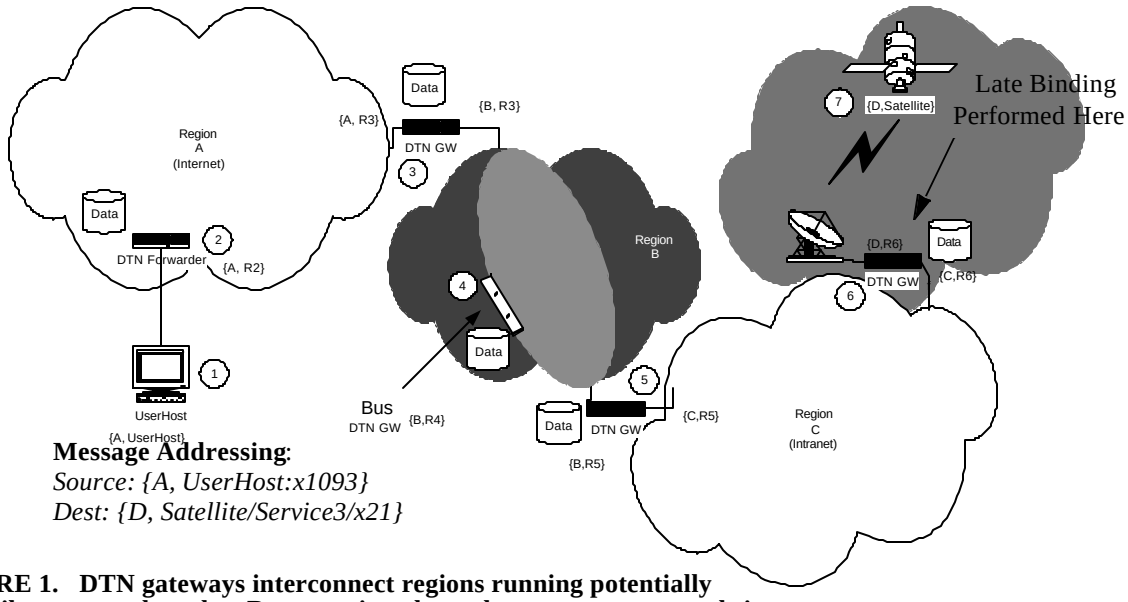

FIGURE 1. DTN gateways interconnect regions running potentially dissimilar protocol stacks. By operating above the transport protocols in use on the incident networks, they provide virtual message switching, in-network retransmission, and name mapping, allowing the use of globally-interoperable names to be mapped to region-local names as required by the adjacent regions' delivery semantics.

IV. A Delay Tolerant Message-Oriented Overlay Architecture

The architecture proposed here for interoperability between and among challenged networks is called the *delay tolerant networking* architecture (DTN), and is based on an abstraction of message switching. Message aggregates are known as "bundles" and are adopted from [IPN01]. The routers that handle them are called "bundle forwarders" or DTN gateways.

The abstraction of moderate-length message delivery for non-interactive traffic can provide benefits over packet switching from the network management point of view because it allows the network's path selection and scheduling functions to have *a-priori* knowledge about the size and performance requirements of requested data transfers. Although virtual circuits also share some of these properties, the need to pre-establish network state end-to-end in such systems is considered an inappropriate match for the link characteristics assumed here. While optimal solutions to these scheduling/routing problems are not easily computed for networks with intermittent connectivity, some heuristics are known and can likely be taken advantage of given some knowledge of the traffic request matrix and available network capacity. Without a message-oriented network architecture, communication capacity requirements cannot generally be known in advance (i.e. if data communications is accomplished by streams that lack message boundaries).

A message-based application interface also tends to favor a split-phase programming style using asynchronous I/O, thus limiting an application's expectation of short request/response interaction times typical of blocking API calls (e.g. in remote procedure calls or method invocations). This influence on program structuring is beneficial to enhance the overall system's robustness to high delay; programs can continue performing other useful tasks while awaiting lengthy client/server interactions.

As an "overlay" architecture, DTN is intended to operate above the existing protocol stacks in various network architectures and provide a gateway function between them when a node physically touches two or more dissimilar networks. For example, within the Internet the overlay may operate over TCP/IP or SCTP/IP, but may provide a gateway service to CFDP [CFDP02] for space links, or to some yet-to-be-standardized sensor transport protocol for sensor/actuator networks. Each of these networking environments have their own specialized protocol stacks and naming semantics developed for their particular application domain. Achieving interoperability between them is accomplished by special DTN gateways located at their interconnection points.

A. Regions and DTN Gateways

The DTN architecture includes the concepts of *regions* and *DTN gateways*, as illustrated in Figure 1. In this example, four regions are illustrated (A, B, C, D). Region B includes a DTN gateway resident on a commuter bus that cycles between DTN gateways 3 and 5. Region D includes a low-earth-orbiting satellite link (LEO) that also provides periodic connectivity (albeit perhaps more regular than the bus which may be subject to vehicular congestion or other delays).

Region boundaries are used as interconnection points between dissimilar network protocol and addressing families. More formally, two nodes are in the same region if they can communication without using DTN gateways (generally using existing protocols local to the containing region). We expect a small number of *region types* (e.g. Internet-like, ad-hoc mobile, periodic disconnected, etc.) may evolve and each instance of the same type will implement a similar stack of underlying protocols. DTN gateways correspond to both the Metanet "waypoint" concept in [W97] and also to the definition of gateways described in the original ARPANET design [CK74]. The waypoint concept describes a point through which data must pass in order to gain entry to a region. This point can serve as a basis for both translation (between region-

Option Name	Mailing Receipt	Delivery Record	Air Delivery	Recipient Pays	Moves Money	Delivery Confirm	Return Receipt	Careful Handling	Insurance	Restricted Delivery	Signature Confirm
Cert. Of Mailing (RM)	Y		(w/PAL)					(w/SH)			
ParcelAirLift (PAL)			Y								
Special Handling(SH)			(w/PAL)	(w/COD)		(w/DC)	(w/RR)	Y	(w/IM)		(w/SC)
Certified Mail(CM)	Y	Y					(w/RR)			(w/RD)	
COD	(w/RM)	Y		Y		(w/DC)	(w/RR)	(w/SH)	(w/RM)	(w/RD)	(w/SC)
Delivery Confirm(DC)				(w/COD)		Y	(w/RM)	(w/SH)	(w/IM or w/RM)		
Insured Mail(IM)			(w/PAL)			(w/DC)		(w/SH)	Y		(w/SC)
Money Order					Y						
Return Receipt(RR)	Y	Y	(w/PAL)			(w/DC)	Y	(w/SH)		(w/RD)	(w/SC)
Registered Mail(RM)	Y	Y		(w/COD)		(w/DC)	(w/RR)		Y	(w/RD)	(w/SC)
Restricted Delivery(RD)			(w/PAL)			(w/DC)	(w/RR)	(w/SH)		Y	(w/SC)
Sig. Confirm		Y				Y					Y

() - indicates the names of options required for specific services
Y - indicates the option provides the service directly

TABLE 1. The US Postal Service Class of Service Offerings. As a basis for a network CoS offering, priorities (low, medium, high), return-receipt, and delivery record are compelling capabilities. The service of careful handling, conceptually similar to reliable delivery, represents another very useful service.

specific encodings) as well as a point to enforce policy and control.

A DTN gateway spanning two regions consists logically of two "halves," each half in one of the adjacent regions above their corresponding transport protocols, analogous to ARPANET-style gateways structured above specific link layer protocols. In operating above the transport layer, however, DTN gateways differ from ARPANET gateways and are instead focused on reliable message routing instead of best-effort packet switching. DTN gateways are responsible for storing messages in nonvolatile storage when reliable delivery is required and mapping between differing transports by resolving globally-significant *name tuples* to locally-resolvable names for traffic destined internally to an adjacent region (see following section). They also perform security checks on arriving traffic to ensure forwarding is to be allowed.

B. Name Tuples

For routing of DTN messages, we elect to use identifiers for objects or groups of objects called *name tuples* comprising two variable length portions. In Figure1, the DTN name tuple(s) for each end point and each router "half" is illustrated in curly braces in the form {`Region Name, Entity Name`}. The first portion is a globally-unique, hierarchically structured region name. It has topological significance: it is interpreted by DTN forwarders to find the path(s) to one or more DTN gateways at the edge of the specified region. It is populated into DTN forwarding tables either statically (by a network administrator), or by one or more dynamic DTN-layer routing protocols (which could be computed centrally for a region, for example). A region name's hierarchical structure provides the ability to reduce the size of DTN forwarding tables in a fashion similar to the Internet's route aggregation in CIDR [RFC2519], yet allows for additional flexibility due to the variable-length substrings allowed between the hierarchy delimiters. Note that despite their similar appearance to DNS names, region names need not necessarily be resolved to any form of address or resolved in a distributed hierarchy as DNS names are.

The second portion identifies a name resolvable within the specified region and need not be unique outside the region. As illustrated in the figure, it may be of arbitrary structure and may contain special indications resolvable in the origin or destination regions. In the case of the Internet, for example, we could have the following tuple:

```
{internet.icann.int,  "http://www.ietf.org/oview.html"}
```

This tuple would refer to the Internet region (in some yet-to-be-defined region hierarchy), along with an Internet-specific local identifier (in this case, a Universal Resource Identifier or *URI*; see [RFC3305] for more details).

As a message transits across a (potentially long and heterogeneous collection of regions), only its region identifier is used for routing. Upon reaching the edge of the destination region, the entity name information is locally-interpreted, and translated if necessary, into a protocol-standard name (or address) appropriate to the containing region. This method of resolving names results in a form of *late binding* for tuples in which only the portion of the tuple immediately needed for message forwarding (the region portion) is used by DTN forwarders. By not imposing any particular fixed structure on the second portion of a tuple, any reasonable naming scheme can be easily accommodated, even unusual ones (e.g. treating only sensor aggregates as endpoints as in [H01]). The concept of late binding has been used in other systems. For example in [WSBL99], it is used primarily for supporting anycast where a location-independent service discovery operation is desired.

Late binding of tuples in DTN differs from the DNS-style Internet naming and addressing which requires one or more DNS transactions to complete prior to the start of an Internet end-to-end conversation. For challenged networks, the need to consult a name-to-address mapping that may be resident only in the destination region seems impractical given potentially large end-to-end delays. While it could be argued the DNS *naming structure* can possibly be separated from its implementation (thereby eliminating the request/response round-trip required to execute the DNS distributed database access), the

DNS structure does not have any explicit method for handling completely opaque naming data or late binding.

The choice of adopting names rather than addresses as the basis for identifying objects derives from an observation of recent trends in the operation of the Internet. The Internet design makes frequent reference to resource sharing as enabled by a (distributed) interprocess control mechanism. Addressing is used for routing and referring to a computational resource (i.e. server), and naming is applied to make the addressing easier for humans. Today's Internet includes objects such as search engines and page caches which are used extensively. In many cases, a name (in the form of a URL or URI web address) effectively refers to a query for data rather than identification of a particular end-system computational resource.

Although the DNS/addressing approach results in greater forwarding efficiency at IP routers (no per-hop name resolution is required), we do not consider the need to perform a per-overlay-hop string lookup detrimental to our approach, as the entire design is not focused on high speed. Indeed, the tuple structure could imply more than two indirect lookups to ultimately determine an endpoint: one in order to resolve the region identifier to a valid local next-hop, and a second lookup to resolve the region-specific data to a valid next-hop or aggregate set address within the specified region. Importantly, however, such queries should be local to a router (given a reasonable DTN routing protocol), and not require a complete end-to-end name resolution transaction.

C. A Postal Class of Service

The notion of a challenged network inherently implies a limitation on various resources. Priority-based resource allocation is therefore important to adopt in the overall model, but care must be taken to avoid so burdensome a class of service architecture as to have it be unimplementable or confusing to users in many cases. The approach taken here is to adopt a subset of the types of services provided by the US Postal Service. This system has evolved to meet the needs of millions of users exchanging non-interactive traffic and has the added benefit of already being reasonably familiar to most users. As such, it seems a highly compelling starting point for considering the classes of service to be offered by a primarily non-interactive networking architecture.

Over its roughly 230 year history, the Post Office Department (and the modern US Postal System of the last half-century) has developed a remarkable class of service offering associated with the seemingly straightforward service of mail delivery. In addition to the basic delivery categories of first-class, priority, express mail, parcel post and "bound printed matter", Table1 above indicates the various special delivery options and the nature of the services they are designed to provide. In this table, the first column indicates the name of the option and possibly its abbreviation, and the first row indicates the intended service. The entries in the matrix indicate if the service is directly supported by the option (indicated as "Y"), or whether it is available only in combination with some other option (indicated parenthetically). Empty entries in the matrix indicate a lack of support for the service using the corresponding option.

As can be seen from the table, some combinations of options are not supported, whereas other options have mutual interdependence. The complexity of this system seems too great as a basis for a network class of service offering, as several of the options are not directly applicable to a data network (e.g. air delivery) or are tied to financial considerations that are considered to be out of scope for the DTN design (e.g. insurance or "moves money"). In a distilled form, however, the following core services seem to be attractive due to their coarse granularity and intuitive character: low, ordinary, and high priority delivery; notifications of mailing, delivery to the receiver (return receipt), and route taken (delivery record). The model is extended with the option of reliable delivery (somewhat akin to careful handling), and messages requiring this service are handled somewhat differently by the routing system in that they require persistent storage and a *custody transfer* at each routing hop (see SectionIV.E.).

Class of Service indications are used within DTN for the allocation of buffering, link capacity, and possibly processing time or power. They are also used in resolving network congestion (see section SectionIV.I.)

D. Path Selection and Scheduling

The DTN architecture is targeted at networks where an end-to-end routing path cannot be assumed to exist. Rather, routes are comprised of a cascade of time-dependent *contacts* (communication opportunities) used to move messages from their origins toward their destinations. Contacts are parameterized by their start and end times (relative to the source), capacity, latency, endpoints, and direction. In addition, a measure of a contact's predictability can help to choose next-hop forwarders for message routing as well as select the next message to be sent. The predictability of a route exists on a continuum ranging from completely predictable (e.g. wired connection or a periodic connection whose phase and frequency are well-known) to completely unpredicted (an "opportunistic" contact in which a mobile message router has come into communication range with another DTN node). Note that the measure of a contact's predictability is sensitive to its direction. For example, a dial-up connection may be completely predictable from the initiator's point of view while being completely unpredicted from the callee's point of view.

Given the message-oriented nature of the system, it is conceivable to ascertain, prior to transport, the set of routing requirements and link capacities that will be available and solve a multicommodity flow optimization problem to obtain the optimal assignment of messages to paths and transmission times. This problem, however, is made more difficult than the traditional multicommodity flow problem due to the temporal nature of the topology graph (edges come and go according to schedules that may be known in advance) and the non-negligible edge transit times. Problems of this kind are nearly universally NP-hard (they fall into a class of *flows over time* optimization problems [KMS02]), but some approximation techniques are known. In any case, by matching pending messages to network capacity, we believe more sophisticated path selection algorithms beyond shortest path may allow for the approximately-optimal use of multiple delivery paths simultaneously.

The particular details of path selection and message scheduling are expected to be heavily influenced by region-specific routing protocols and algorithms. At this relatively early stage of DTN development, several challenging problems have been identified: determination of the existence and predictability of contacts, obtaining knowledge of the state of pending messages

given assumptions of high delay, and the problem of efficiently assigning messages to contacts and determining their transmission order. While very simple (e.g. greedy) heuristics for these problems can be implemented without excessive problems, each issue represents a significant challenge and remains as future work.

E. Custody Transfer and Reliability Semantics

The DTN architecture includes two distinct types of message routing nodes: persistent (P) and non-persistent (NP). P nodes are assumed to contain nontrivial amounts of persistent message store, and NP nodes are not. Unless they are unable or unwilling to store a particular message, P nodes generally participate in *custody transfer* using the appropriate transport protocol(s) of the containing region. A custody transfer refers to the acknowledged delivery of a message from one DTN hop to the next and the corresponding passing of reliable delivery responsibility. Custody transfer is akin to delegating responsibility for delivering postal mail to a person or service who promises to do so.

The custody transfer concept is fundamental to the architecture in order to combat potentially high loss rates and to relieve potentially resource-poor end nodes from responsibilities related to maintaining end-to-end connection state. In particular, end-nodes do not ordinarily need to keep a copy of data that has been custodially transferred to a DTN next hop. For end nodes *insisting* on an end-to-end acknowledgment, a "delivery confirmation" may be optionally requested, although what to do with this indication is left to the requesting application. It may employ its own retransmission techniques if desired.

In contemplating a change from end-to-end reliable delivery semantics to a hop-by-hop reliability approach (with end-to-end notification), we may ask whether a qualitatively different type of reliability is being provided. We believe that the custody transfer mechanism is no less reliable than using typical end-to-end reliability, and that the provision of the end-to-end optional acknowledgment is consistent with the *end-to-end principle*---that only the applications truly know what they require. Indeed, custody transfer can be viewed as a performance optimization for end-to-end reliability, again consistent with the end-to-end principle.

F. Protocol Translation and Convergence Layers

The facilities provided by the transport protocols in use within the regions containing a DTN P node may vary significantly. For example, any transport protocol may or may not offer the following: reliable delivery, connections (with indications of connection failure), flow control, congestion control, and message boundaries. As the bundle forwarding function assumes an underlying reliable delivery capability with message boundaries when performing custody transfer, transport protocols lacking these features must be appropriately augmented. Figure2 illustrates the implementation structure for a bundle forwarder, including a number of transport-protocol-specific *convergence layers* used to add reliability, message boundaries, and other features above those transport protocols requiring augmentation. (Note that TCP in the Internet requires augmentation due to its lack of message boundaries; a convergence layer for SCTP would likely be minimal and not require such augmentation). The design and implementation details of convergence layers is specific to the transport protocols being augmented, and are therefore beyond the scope of this paper.

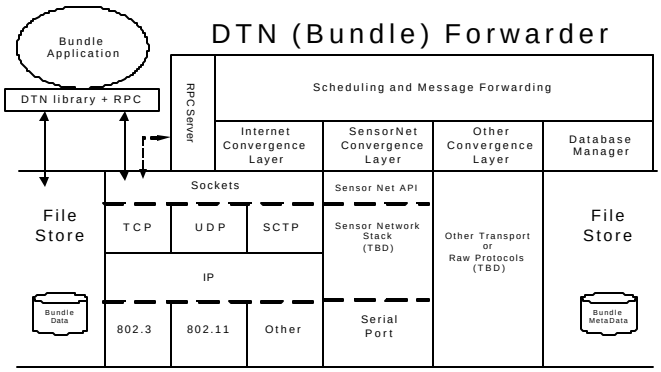

FIGURE 2. Structure of a DTN forwarder. Multiple *convergence layers*, one per protocol stack, provide a common interface to the message scheduler/forwarder.

In cases where reliable delivery is provided by an underlying transport, a bundle forwarder need only manage connection state and initiate restarts if a connection is lost. In the case of connection-oriented protocols, detection of a lost connection is generally provided through the application interface (via signals or other errors using the socket interface, for example). In cases where no direct support is provided for detecting failures, the bundle forwarding function may set a coarse-grained timer to re-start message transfers should it be concluded they have failed. This is designed as a fallback measure in cases of underlying communication failure, and is not expected to be an especially efficient mechanism for initiating retransmission.

Setting the coarse-grain retransmission timer will vary depending on the details of the containing region, and thus represents a certain form of layer violation in which the overlay "network" layer is able to be sensitive to underlying "physical" layer properties. In challenged networks, knowledge of some path properties at the forwarding layer appears to be very useful in selecting error control policy. In space communications, for example, communication opportunities and approximate path delays may be known ahead of time due to planetary dynamics. In other cases, alternative means (e.g. based on geographic location coupled with knowledge of the intermediate communications media) may give reasonable loose bounds on what values to use for the timer. Note that given the often-disconnected state of the network, we cannot generally rely on TCP-like estimates of the current end-to-end round-trip time as a basis for setting retransmission timeouts.

G. Time Synchronization

The DTN architecture requires a level of time synchronization between communicating parties. For the purposes of simple forwarding, time synchronization may be loose, as the time is used only to garbage collect messages which have exceeded their source-specified expiration times. For most circumstances, however, there are several benefits from imposing a more stringent constraint on time synchronization---on the order of one millisecond. The requirement stems from the observation that synchronized timing is needed by many distributed applications used in challenged environments and is

required by the DTN's approach to scheduling, path selection, and congestion management. Although more burdensome than time synchronization requirements on the Internet (which are essentially nonexistent), we believe the problem of time synchronization is not so difficult to solve as to make optional. Protocols such as NTP [NTP3] have provided 1ms accurate time synchronization (or better) within the Internet for years, and most existing networks for extreme environments already provide some (often out-of-band) means for obtaining accurate time.

The need for time synchronization is based on several features common to many challenged environments. First, challenged networks are often used to communicate with devices deployed in remote and possibly hostile locations. Such instruments often collect data and/or position as a function of time and need to be controlled. While the DTN architecture does not strive to support real-time control loops, it does aim to provide a mechanism to deliver pre-programmed control instructions to be executed at reasonably precise future points in time. Without time synchronization, *post-facto* data analysis and pre-programmed control is made considerably more difficult and unreliable. For bundle routing, synchronized time is used to remove pending messages from the delivery system when they expire. This feature does not require especially accurate synchronized time, but deviations of more than a few minutes could prove to be problematic.

H. Security

The security model for the DTN architecture differs somewhat from traditional network security models in that the set of participants includes the network routers (i.e. DTN forwarders) themselves in addition to the principals. Most security approaches involve the mutual authentication and private exchange of data between two network users, leaving the intervening network as a non-participant. In the DTN case, we are more interested in verifiable access to the carriage of traffic at a particular class of service and want to avoid carrying traffic potentially long distances that is later found to be prohibited.

To implement the security model, each message includes an immutable "postage stamp" (a type of capability) containing a verifiable identity of the sender (or role), an approval (and approving authority) of the requested class of service (CoS) associated with the message, and other conventional cryptographic material to verify accuracy of the message content. Routers check credentials at each DTN hop, and discard traffic as early as possible if authentication fails. This approach has the associated benefit of making denial-of-service attacks considerably harder to mount as compared with conventional Internet routers.

The current approach uses public key cryptography as a starting point for keying. Routers and principals are issued public/private keypairs, and a principal sending a message must obtain a signed copy of its public key from a certificate authority known to DTN forwarders. (All routers are assumed to be pre-equipped with copies of one or more certificate authority public keys and their own public/private key pairs). A user then presents her signed public key along with a message to be carried signed using her private key. At the first DTN router, the signed public key is used to validate the sender and requested CoS against an access control list stored in the router. Accepted messages are then re-signed in the key of the router for transit. Using this approach, only first-hop routers need cache per-user certificates, and then only for adjacent users. Non-edge "core" routers can rely on the authentication of upstream routers to verify the authenticity of messages. We believe this approach will help to improve the scalability of key management for these networks, as it will limit the number of cached public key certificates to a function of the number of adjacent routers rather than the number of end-users. This should provide both the obvious advantage of space savings, but also an improvement to system management as router keys are expected to be changed less frequently than end-user keys. As DTN routers are likely to be deployed in remote areas, re-keying operations may be a comparatively burdensome system management tasks, so limiting the number and frequency of certificate updates should provide additional savings.

The current approach is partially susceptible to compromised routers. If an otherwise-legitimate router is compromised, it would be able to utilize network resources at an arbitrary CoS setting and send traffic purportedly originating from any user who's identity is known to the router. However, if the message signature is carried end-to-end (an option for DTN security), a legitimate user could repudiate the origin of any traffic generated in this manner. Thus, we believe a reasonable trade-off is to admit the possibility that a compromised router could launch a denial-of-service attack in order to gain the scalability benefits of not checking end-user credentials at every hop.

I. Congestion and Flow Control

As a form of hop-by-hop architecture, flow control and congestion control for DTN are closely related. Flow control in this context refers to limiting the sending rate of a DTN forwarder to its next hop. Congestion control refers to the handling of contention for the persistent storage at a DTN forwarder. It is especially difficult when messages request custody transfer, as accepting custody of a message corresponds to a promise to ensure its delivery with high probability.

For implementation of flow control, a DTN forwarder will attempt to take advantage of whatever flow control mechanism is present in the underlying region-local transport protocols. For most mature networks, some such mechanism exists already (e.g. TCP, X.25, RTS/CTS, XON/XOFF, explicit admission/rate control, etc). For other networks where such mechanisms are still being developed, region-specific mechanisms may be constructed in the DTN forwarders' convergence layers. Doing so is (naturally) region-specific, and is beyond the scope of this paper. In any case, the uppermost functions of a DTN forwarder generally assume the existence of flow control, so some such mechanism must be present to ensure reliable message delivery.

DTN congestion control is especially difficult to implement as compared to other aspects of the architecture because of two features: contacts may not arrive for some time in the future (so accumulated data may not have an opportunity to drain in the immediate future), and received messages for which custody has been accepted cannot be discarded except under extreme circumstances or on expiration. The current approach uses a priority queue for allocating custody storage. First, messages that are too large are denied custody transfer. Next, messages are spooled FCFS based on priority. Two potential problems that arise include a form of priority invesion (arriving higher-priority messages may not have custody storage available if lower-priority messages arriving earlier have been cus-

todially received) and head-of-line blocking. The blocking can arise when a DTN forwarder accepts custody for messages that are outgoing on a contact that has not yet begun, but is also the next hop for messages to a currently-available contact that does not require custody transfer. In such a case, the persistent storage in the node may be completely consumed by the pending messages, thereby preventing the non-custody messages from transiting.

The mechanisms available to deal with congestion may be proactive or reactive. Proactive methods generally involve some form of admission control, to avoid the onset of congestion in the first place. In many cases, a single region may be under the administrative control of a single entity, and this approach may be practical. In addition, any messages who have met their expiration times can be safely discarded. If proactive methods are insufficient or unavailable, reactive means must be used which usually result in degraded performance. For DTN, the possibilities include reserving buffer space as a function of CoS, rejecting incoming connections for new messages when buffer space is full, arranging for custody transfers to other potential custodians that may not be the most desirable next hop (a form of hot potato routing) and discarding non-custody bundles in favor of any bundles requiring custody transfers. In unusual and dire circumstances, a facility for removing bundles requiring custody may be available, but removing such information is to be avoided if at all possible, as deleting reliable bundles would be considered a system fault.

V. Application Interface

As described, the DTN architecture is built as an overlay network using messages as the primary unit of data interchange. Applications making use of the architecture must be careful not to expect timely responses and must generally be capable of operating in a regime where a request/response turnaround time exceeds the expected longevity of the client and server processes. In addition, applications must be prepared to handle the creation and manipulation of name tuples and their registrations (for demultiplexing received messages), class of service specifiers, and authentication information.

Bundle forwarders must implement the application interface, along with bundle routing and forwarding functionality, persistent message storage facilities, time synchronization and transport protocol convergence layers. The application interface is non-blocking, and involves registration and callback functions between bundle-based applications and the local bundle forwarding agent. Registration data is persistent, and bundle meta-data is handled with database semantics. Generally speaking, all bundle applications should be structured to operate without trouble in the face of reboots or network partitioning as much as possible.

VI. Implementation Experience

A prototype DTN system has been developed under the Linux operating system, which implements the application interface, rudimentary bundle forwarding across scheduled and "always on" connections, detection of new and lost contacts, and two convergence layers (for TCP/IP as well as a bundle-based proxy to the Berkeley *mote* network [H00]). The DTN bundle forwarder, as illustrated in Figure2 above, comprises about 22k lines of C program code, and currently makes use of the Sun RPC library to interface with applications, and Sleepycat™ data management software for persistent storage of message meta-data. [2] The prototype has been used as a proof-of-concept of the overall architecture, and also to show the general utility of the non-interactive reliable messaging service it provides.

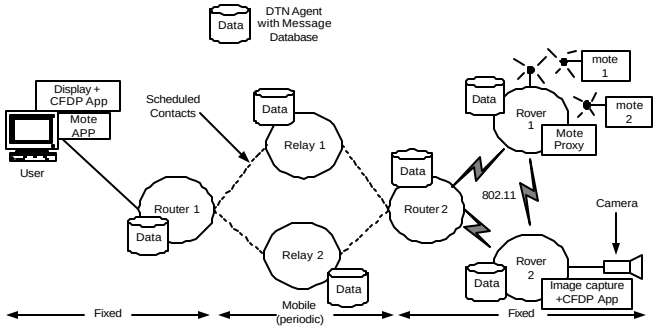

FIGURE 3. Illustration of the prototype test environment. No end-to-end path ever exists between the display applications and data collectors (motes and camera). Relays know schedule of contact opportunities and route messages reliable toward user, splitting the message transfer across contacts if necessary..

Figure3 illustrates the topology used in several experiments. The solid-line links are always connected, and the dotted lines indicate intermittent links. The intermittent links are periodic and out of phase with respect to each other so no end-to-end path between the rovers and workstation is ever present. Data is sourced either from the *motes* near Rover 1 or from the camera attached to Rover 2. The motes are small battery-powered computers equipped with a radio and interface connector supporting a wide array of sensors. In this case, light level readings are sent to Rover 1, where a DTN agent encapsulates the readings into DTN messages (using a mote proxy located in the agent's convergence layer) and delivers them through the DTN network to a mote application on the user's workstation. This configuration allows for sensor readings to accumulate without being lost even during periods of disconnection.

The camera is driven by a simple image capture application in combination with a DTN-compatible implementation of the CFDP file transfer program. CFDP is a standardized file transfer system used by the space community for moving files between and among Earth and space assets. In the conventional version of CFDP, all responsibility for reliable delivery and routing is implemented within CFDP itself, and these facilities are not made available to any other co-located applications. In this example, where the CFDP application is able to use the underlying DTN facilities, its implementation is made significantly less complex, and the file transfer facilities are made common to any applications requesting it.

The results from the experiments to date are primarily qualitative, and tend to support our expectations of the capabilities of the architecture. In the configuration of Figure3, the primary contributing factor to end-to-end latency is the amount of time the relays must wait for contacts to become available (as

2. RPC is may be (optionally) used between an application host and a nearby bundle forwarder if end-node resources are limited and the path between application and forwarder is of reasonable performance.

expected). When the contacts are allowed to be always available, the primary latency arises from the file system access when writing meta-data (using SleepyCat) or message data directly. In the example, all routers are single-board systems where the persistent storage is implemented in comparatively slow flash memory. Thus for small-sized messages (typical of the mote application and CFDP in its default configuration), comparatively high overhead is experienced. The problem is ameliorated by opting for larger message sizes, or utilizing a type of persistent memory with a lower I/O fixed cost than flash (e.g. compact flash form factor hard drives).

VII. Related Work

The DTN architecture is based most closely on work that originated with the Interplanetary Internet [IPN01] design. It represents a significant evolution from the previous work in the following areas: the types of network paths/connections has been generalized to include unpredictable contacts, the formalization of convergence layers for different network stacks has been created, the security model has been more carefully defined, the class of service model has been simplified and the notion of some payment unit for transport has been removed. Furthermore, any "space environment"-dependent features of the architecture have been excised in favor of more general mechanisms.

With respect to routing in frequently-disconnected networks, a number of recent efforts have arisen. In ZebraNet [JOW02], wireless sensor nodes (attached to animals) collect location data and opportunistically report their histories when they come in radio range of base stations. They explore the case of mobile base stations and sensor devices and the use of two flooding-based routing protocols. In DataMules [SRJB03], low-power sensor nodes can save power if periodically visited by a "mule" that periodically travels among them and provides a non-interactive message store-and-forward service. In these two efforts plus that of Vahdat [VB00], device (animal) mobility models are employed to predict the ability of partially connected networks to deliver data eventually. A large body of earlier work [P01] focuses on running IP over mobile ad-hoc networks, relying primarily on *reactive* protocols which supply routes only after a message is sent. DTN capabilities could be used in conjunction with such facilities during periods of disconnection.

The use of late binding for names in DTN is shared with, although not directly based upon, the work on Intentional Naming [WSBL99]. Here, names represent a form of query and are used specifically for anycast in order to locate nearby network services. Routing based on names is shared, to some degree, with Internet Content Routing [GC01]. This work focuses on using routing on names to provide a content distribution facility for the Internet, addressing its scalability and performance. It does not use two separate name components as in DTN, but does suggest the viability of the name-based routing mechanism. The generality of the entity portion of names is influenced by [H01], where database-like queries are effectively used as addresses for groups of sensor nodes.

The architectural thinking regarding interoperability and layering is guided by principles of the ARPANET/Internet [CK74,C88]. DTN gateways operate in many ways similar to Internet routers, but are adapted to use in high-delay and disconnected environments.

VIII. Conclusion

The DTN architecture aims to address the desire to provide interoperable communications between and among a wide range of networks which may have exceptionally poor and disparate performance characteristics. The design embraces the notions of message switching with in-network retransmission, late-binding of names, and routing tolerant of network partitioning to construct a system better suited to operations in challenged environments than most other existing network architectures, particularly today's TCP/IP based Internet.

The architecture represents a generalization of the Interplanetary Internet architecture to challenged networks other than space. The previous work was closely tied to issues of deep space communications in particular, but contributed many key ideas toward the development of a networking architecture applicable for challenged internetworks more generally. The design also derives in part from some interesting trends in the Internet: a move toward content-based naming, creation of administrative "regions", and alternative routing structures (e.g. network overlays).

The proposed DTN architecture advocates a change to the basic service model and system interface most Internet-style applications have become accustomed to, motivated by the exceptionally poor performance present in some networks. This is a comparatively radical approach; other approaches aim to "repair" underlying link performance problems or alter limited portions of the Internet architecture, such as routing, with additional protocols in an effort to keep the current service model and existing TCP/IP based protocols constant. Because it provides a different type of network service than Internet, the DTN design makes a different set of choices in the architectural design space: messages versus packets, a form of hop-by-hop reliability and security versus end-to-end, name based routing versus address based routing, and a routing abstraction of partially-connected rather than fully-connected network graph. Interestingly, DTN can be overlaid upon the TCP/IP based Internet easily, and therefore remains compatible. This is not the most interesting case, however, as its strength lies in its ability to tie together dramatically different types of networks with unusual connectivity properties. As such, in some ways it makes more limited assumptions on the underlying protocol layers than IP does upon its underlying link layers.

Only time will tell what application interfaces and service semantics will most appropriately match applications to challenged networks, but we believe the DTN architecture puts forth several design decisions worthy of consideration. In addition, we believe it is timely to consider a very broad range of network characteristics in formulating a new network architecture, as it appears likely an ever increasing number of these features will have to be dealt with.

IX. Acknowledgment

The author wishes to thank the members of the Interplanetary Internet Research Group for their previous work on the initial definitions of bundling and naming, without which this architecture would not exist. Members of this group include Vint Cerf (MCI/WorldCom), Adrian Hooke and Scott Burleigh (NASA/JPL), Bob Durst and Keith Scott (the MITRE Corporation), and Howard Weiss (SPARTA). The author is especially indebted to Bob Durst and Scott Burleigh for an ongoing col-

laboration regarding the DTN design. Versions of the manuscript benefited from the comments of David Culler and Sylvia Ratnasamy.

X. References

[ABKM01] D. Andersen, H. Balakrishnan, F. Kaashoek, R. Morris, “Resilient Overlay Networks”, Proc. SOSP 2001.

[BLMR98] J. Byers, M. Luby, M. Mitzenmacher, A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data”, Proc. SIGCOMM 1998.

[C88] D. Clark, “The Design Philosophy of the DARPA Internet Protocols”, Proc. SIGCOMM 1988

[CFDP02] CCSDS File Delivery Protocol (CFDP), CCSDS 727.0-B-1, January 2002, available from http://www.ccsds.org

[CK74] V. Cerf, R. Kahn, “A Protocol for Packet Network Intercommunication”, IEEE Trans. on Comm., COM-22(5), May 1974

[DMT96] R. Durst, G. Miller, E. Travis, “TCP Extensions for Space Communications”, Proc. MobiCOM 1996

[DSN] NASA’s Deep Space Network, See information page at http://deepspace.jpl.nasa.gov/dsn

[FMS98] D. Feldmeier, A. McAuley, J. Smith, D. Bakin, W. Marcus, T. Raleigh, “Protocol Boosters”, IEEE JSAC, Apr 1998

[GC01] M. Gritter, D. Cheriton, “An Architecture for Content Routing Support in the Internet”, Proc. Usenix USITS, March 2001.

[GZR01] C. Guo, L. Zhong, J. Rabaey, “Low Power Distributed MAC for Ad Hoc Sensor Radio Networks”, Proc. IEEE Globecomm 2001.

[H00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister. “System architecture directions for network sensors”. Proc. ASPLOS 2000.

[H01] J. Heidemann et. al., “Building Efficient Wireless Sensor Networks with Low-Level Naming”, Proc. SOSP, Oct 2001, pp 146-159

[IPN01] V. Cerf et. al., “Interplanetary Internet (IPN): Architectural Definition”, Available as http://www.ipnsig.org/reports/memo-ipnrg-arch-00.pdf

[JOW02] P. Juang, H. Oki, Y. Wang, M. Maronosi, L. Peh, D. Rubenstein, “Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet”, Proc. ASPLOS 2002, Oct. 2002

[KMS02] E. Kohler, R. Mohring, M. Skutella, “Traffic Networks and Flows Over Time”, Technical Report 752-2002, TU-Berlin, 2002

[M95] J. Mogul, “Fragmentation Considered Harmful”, Proc. SIGCOMM 1995.

[NTP3] David Mills, “Network Time Protocol (Version 3) Specification, Implementation and Analysis”, RFC1305

[P01] C. Perkins, ed., *Ad Hoc Networking*, Addison Wesley, 2001

[RFC793] “Transmission Control Protocol”, Internet Request for Comments 793, Sep. 1981

[RFC3135] J. Border et. al., “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations”, Internet Request for Comments RFC3135, June 2001

[RFC2519] E. Chen, J. Stewart, “A Framework for Inter-Domain Route Aggregation, Internet Request for Comments RFC2519, Feb. 1999

[RFC3305] M. Mealling, R. Denenbers, eds., “Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations”, Internet Request for Comments RFC 3305, Aug 2002

[SRJB03] R. Shah, S. Roy, S. Jain, W. Brunette, “Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks”, to appear, IEEE SNPA Workshop, May 2003

[VB00] A. Vahdat, D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks,", Duke Technical Report CS-2000-06, July 2000. http://www.cs.duke.edu/~vahdat/ps/epidemic.pdf

[W97] John Wroclawski, “The MetaNet: White Paper - Workshop on Research Directions for the Next Generation Internet”, Available from http://www.cra.org/Policy/NGI/grouppapers.html, May 1997

[WSBL99] W. Adgie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, “The Design and Implementation of an Intentional Naming System”, Proc. SOSP, Dec 1999