A Bundle of Problems

Lloyd Wood
Global Government Solutions Group
Cisco Systems
London, United Kingdom.
lwood@cisco.com

Wesley M. Eddy
Federal Network Systems
Verizon Business
Cleveland, Ohio, USA.
wesley.eddy@verizonbusiness.com

Peter Holliday
Global Government Solutions Group
Cisco Systems
Brisbane, Australia.
phollida@cisco.com

Abstract—Delay-Tolerant Networking" (DTN) is a neologism used for a new store-and-forward architecture and protocol suite intended for disrupted networks where there is intermittent or ad-hoc connectivity. This has been proposed as one approach to supporting delay-tolerant networks. Work in the late 1990s on the "Interplanetary Internet" forms the basis for current DTN protocols and architecture. That early work considered transport protocols robust to the hours-long propagation delays of deep-space communications. DTN is also known, primarily in military circles, as Disruption-Tolerant Networking, due to the dynamic links and outages in the military tactical environment, rather than long-delay links. In both cases, DTN technologies are well-suited to applications that are mostly asynchronous and insensitive to large variations in delivery conditions. DTN networks differ sufficiently from traditional terrestrial networks in their characteristics and connectivity that link, network and transport protocols must be carefully considered and chosen to cope with these different characteristics, or new protocols can be designed that are suited for the problems that these DTN network conditions impose. The "Bundle Protocol" exists within the DTN architecture, which sends bundles over subnet-specific transport protocols, called "convergence layers." "Bundling" has undergone a large amount of shared development and design over a period of years as a research effort. We examine the Bundle Protocol and its related architecture closely, and discuss areas where we have found that the current Bundle approach is not well-suited to many of the operational concepts that it was intended to support. Problems with the Bundle Protocol and its convergence layers exist in mechanisms for error detection and overall reliability. This weakens the Bundle Protocol's suitability to disrupted and error-prone networks. We show that these reliability issues can lead to performance problems in DTN networks, requiring mitigation. Open research and development areas also exist with design choices in handling timing information, in determining necessary and sufficient security mechanisms, in its Quality of Service capabilities, and in other aspects of application or content identification. We show that the existing DTN bundling architecture has a number of open real-world deployment issues that can be addressed. We suggest possible remediation strategies for these weak areas of the bundle protocol that we have been working on. We also look at alternate approaches to DTN networking. Rather than only providing criticism, this paper identifies open issues, where work on modifying the Bundle Protocol is encouraged and approaches to address its various problems are suggested.

TABLE OF CONTENTS

1. INTRODUCTION

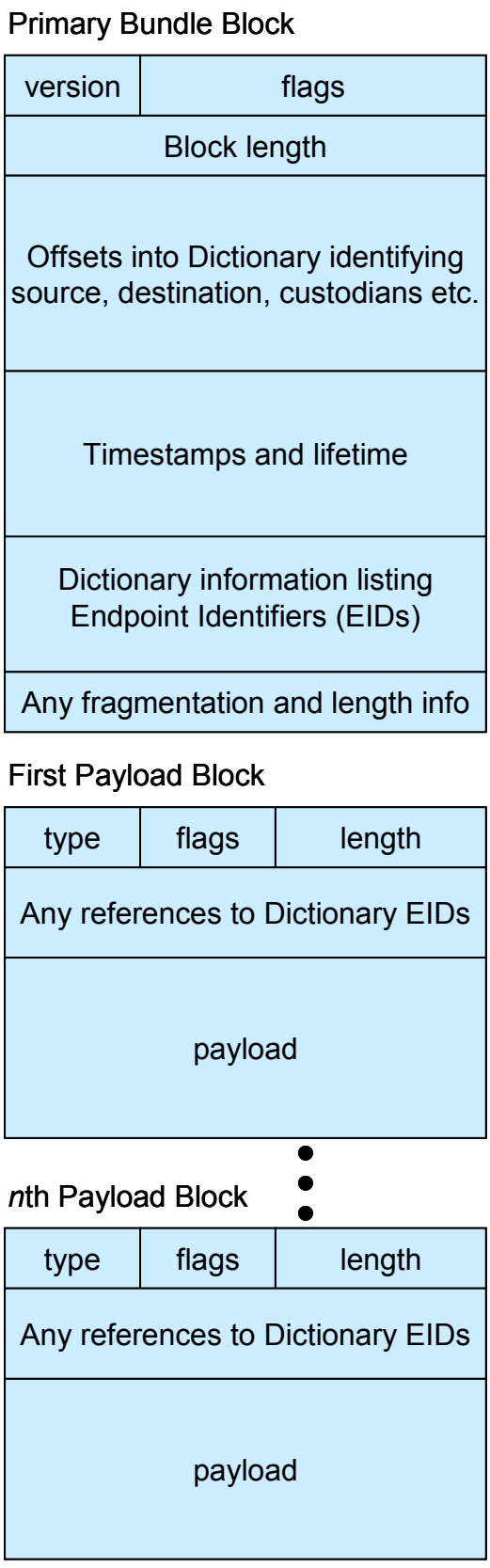
The term "Delay-Tolerant Networking" stems from Fall's seminal 2003 SIGCOMM paper, which introduced an architecture generalised from design work for the Interplanetary Internet [1]. The Delay Tolerant Networking Research Group (DTNRG) in the Internet Research Task Force (IRTF) began working on development of this architecture [2] and its Bundle Protocol [3]. The DTNRG community has enjoyed wide participation from experts in a variety of disciplines, with knowledge of a variety of diverse operating environments.

The Bundle Protocol design is intended to meet the needs of widely divergent classes of networks including deep-space exploration, undersea networking, tactical military networks, ad-hoc networks, sensor networks, and other so-called 'challenged networks' [4]. A bundle consists of a number of concatenated blocks, including common shared metadata (the 'bundle header' or 'primary bundle block') followed by a number of other payload blocks for varying purposes, as shown in figure 1.

By design, the bundle specification that has been developed is mostly focused on the logical layout of the bundle format, rather than specifying the operations and interactions between the protocol entities, called "bundle agents". It is in many ways more similar to a complex file format specification than to an internetworking protocol in this regard. This was purposefully done so that the details for a specific implementation or deployment scenario could be worked out later. For example, the operations for forming bundle protocol addresses in Endpoint Identifiers (EIDs) and forwarding bundles based on them are not yet defined.

1

[1] 978-1-4244-2622-5/09/$25.00 ©2009 IEEE.
[2] IEEEAC paper #1023, Version 10 final, updated December 23, 2008



Many fields (including EIDs and fields using SDNVs) are variable-length.

Figure 1 – Basic bundle and block structure

Nor are Quality of Service mechanisms, means for key exchange and establishment of security associations, or network management and monitoring protocols.

This Bundle Protocol is intended to provide a general solution to networking for ad-hoc, delay-tolerant and disrupted networks. These networks can be said to have one characteristic in common: they are unlike terrestrial fixed networks in their link conditions and connectivity. They can vary in many other ways. This makes it very difficult to create a generalized networking solution that is suitable for a wide range of delay-tolerant networks. The Bundle Protocol alone does not solve the problems of networking in any of these environments. Rather, it is intended to provide a common format for store-and-forward networking messages and proposes that the availability of in-network storage in bundle agents will allow the challenges of these networks to be overcome. Many of the innovations that enable and support delay-tolerant network services should be understood as existing outside the basic Bundle Protocol itself, and as being largely independent of the Bundle Protocol. The Bundle Protocol requires significant supporting infrastructure to function, in convergence layer adapters, and in enhancements or additions for specific implementations or deployments.

The Bundle Protocol always sits upon a local transport 'convergence layer' whose design matches local network conditions. Bundle Protocol identifiers for routing, the Endpoint Identifiers (EIDs), are somehow mapped to local routing addresses in the local subnetwork via late binding.

The Bundle Protocol itself only supports internetworking indirectly, by permitting multiple different convergence layers to be used, or multiple diverse naming schemes for EIDs to be used. Controlling the mappings, propagation of routing information, and discovery of node and application identifiers, is left as orthogonal to the Bundle Protocol, and may be accomplished by entirely incompatible methods in different networks. This implies that there will not be a single universal DTN like the shared and widely-understood Internet, but rather many independent and incompatible DTNs. This implication should be carefully considered against the goals of network-centric communications infrastructure. An internetwork DTN architecture based on the Bundle Protocol may be undesirably similar to the beads-on-a-string model [5] with translation gateways between neighbouring networks, should the local enhancements to the Bundle Protocol become too divergent to permit high-level interoperability between user applications.

This indirection in implementing specific features separate from the Bundle Protocol itself does not prevent the Bundle Protocol's performance from being affected by local networking conditions. Our experiments with the Bundle Protocol in operational environments with deployed DTN networks have clearly demonstrated that the Bundle

Protocol is affected by real-world concerns and physical effects, such as error-detection, reliability and timing. These physical effects act on the logical Bundle Protocol format even through the local network and the local convergence layer.

2. OUR USE OF THE BUNDLE PROTOCOL

Authors of this paper participated in the team that is the first to use the Bundle Protocol from space. The Bundle Protocol was tested by transferring image data from the UK-DMC remote-sensing satellite [6][7]. We implemented bundling over the *Saratoga* UDP-based transfer protocol developed at Surrey Satellite Technology Ltd (SSTL). *Saratoga* is designed for scheduled private links where high utilization is required by a single flow, and where loss recovery presumes that congestion is not present.

These private links are the wireless links from a remote-sensing satellite in low Earth orbit to a ground station, downloading image data during a pass [8]. *Saratoga* was designed to work as a standalone protocol, and was later also adopted to carry bundles as a convergence layer for Bundle Protocol use [9]. Proactive fragmentation of files delivered in bundle fragments across multiple passes over a ground station was demonstrated. This fragmentation is not found to be currently of benefit to SSTL's operational imaging scenario, because planned satellite contacts are rarely interrupted, image files are sized to be transferable in a single pass, and images are not stored on the satellite for long periods if not downloaded, but can be replaced *in situ* by newer images. However, the support for fragmentation may have utility in future systems where more than one ground station is used to upload or download large volumes of data, where data, *e.g.* large code executables, are uploaded to the satellite across multiple passes, or where more automated distribution and processing of data is desired.

We also used the Bundle Protocol over TCP as a convergence layer, for carrying downloaded remote sensing data across the shared terrestrial Internet. This was implemented with existing DTN2 *dtnd* bundle agent software [10][11]. The DTN bundles for these experiments were carried by and fully utilized existing IP networking throughout.

In working with the Bundle Protocol and multiple convergence layers in both concept and practice, we have become familiar with its development and design choices – and with its current architectural weaknesses and areas needing work. As it may be possible to remedy many of those weaknesses, we discuss them at length in this paper.

As popular media coverage has somewhat sensationalized the benefits and features of the Bundle Protocol beyond those present in its current incarnation, these weaknesses deserve to be more widely known, so that any potential user of the Bundle Protocol is aware of what benefits the Bundle Protocol does currently bring to a user's application – and what it doesn't yet bring.

3. BASIC BUNDLE PROTOCOL ARCHITECTURE

From an architectural standpoint, bundling differs from traditional Internet paradigms, even at the highest level of addressing and the definition of endpoints. In the Internet's layered model, applications and transport protocols run end-to-end, with the Internet Protocol (IP) mediating link vagaries at each hop between source and destination. Internet applications send data in different ways (e.g. bytestreams versus datagrams) and that data is broken into relatively small IP packets with well-defined minimal and maximal sizes by the transport and packetization sub-layers.

In the DTN bundling architecture, end applications "register" with bundle agents. The applications then pass data to the bundle agents, which then create bundles and perform the end-to-end transmission functions behalf of the applications. However, in contrast to the packetization of IP data by transport layers using IP, the "bundleization" rules for DTN data are not yet well-understood.

Internet applications typically select transport protocol configurations that fit their needs for reliability, and based on that configuration which they control, can clearly understand what levels of retransmission and error-detection are in-effect for their transmissions. Reliability of bundle transmissions is intended to be provided through the optional "custody transfer" mechanism in the bundle protocol. The custody transfer is essentially a bundle agent acknowledgement that a particular bundle has been received by some later bundle agent – the custodian – which has taken on the responsibility for getting the bundle to its final destination. This is what allows the bundle protocol to provide some in-network retransmission service to avoid the expense or even impossibility of end-to-end retransmission. However, as currently defined, the applications still have to provide many reliability mechanisms of their own to detect or prevent corruption of data, misdelivery of data, replication of data, *etc.*

While the definition of Bundle Protocol reliability was apparently discussed at an early phase [4], we could not find published records of those discussions. How reliability and error detection is handled is not explained in the architecture or bundle protocol definition RFCs, and the details and implications of this have not been known or well-understood until recently. We discuss reliability at greater length later in this paper.

IP networks have well-understood means for locating services and resolving host and service identifiers into locators, as well as updating the bindings in the mapping

systems – the Domain Name System (DNS), Session Initiation Protocol (SIP) servers, and others. The Bundle Protocol currently relies on late-binding of all identifiers and does not yet distinguish between host and service identifiers. This use of late binding is what enables a DTN to function in the absence of well-connected infrastructure that possesses the low latency of communications required to perform global lookups and multiple recursive exchanges in sequence – a necessity for DNS. However, the localization of the mappings raises the issue of how to securely update them. There are not yet means for bundle agents to police registrations, and secure registration and efficient registration state management may itself require additional protocol machines that are not yet defined. There are interesting cases possible, such as a bundle agent accepting custody of a bundle for an application who it thought was registered, even though that application endpoint has moved on to another bundle agent. Handling errors due to stale local state and gracefully recovering becomes necessary due to the inability to coordinate global state. More work is required to prevent or at least mitigate this.

IP network engineers are familiar with the creed "IP over anything", referring to the ability of IP to work over and bridge across many types of underlying subnetworks. For example, specifications exist for IP over many types of wired and wireless links, and demonstrations have even been done of IP over challenging transmission media with large delays, such as carrier pigeons, as a lark.

The Bundle Protocol is similarly intended to work over any type of subnetwork, and accomplishes this through "convergence layer adapters" tailored for each subnetwork. Convergence layer adapters play a significant role in the reliability of the bundle protocol, as discussed later. Specifications of IP over different link types and DTN convergence layers differ in their scope, as IP specifications define how to configure addresses, discover routers, maintain neighbor reachability information, and other basic operations that are not included in the definition of scope of a DTN convergence layer adapter.

4. PROBLEMS AND ISSUES WITH BUNDLING

4.1 Reliability, error detection, checksums and performance

The current base Bundle Protocol specification does not attempt to detect errored bundles, in that it has no checksum support for error detection and rejection of corrupted bundles – either to detect corrupted header information (metadata) that the Bundle Protocol uses for its own needs, or to detect corruption in payloads that it carries. It cannot be determined if the bundle information received by a bundle destination is error-free or not.

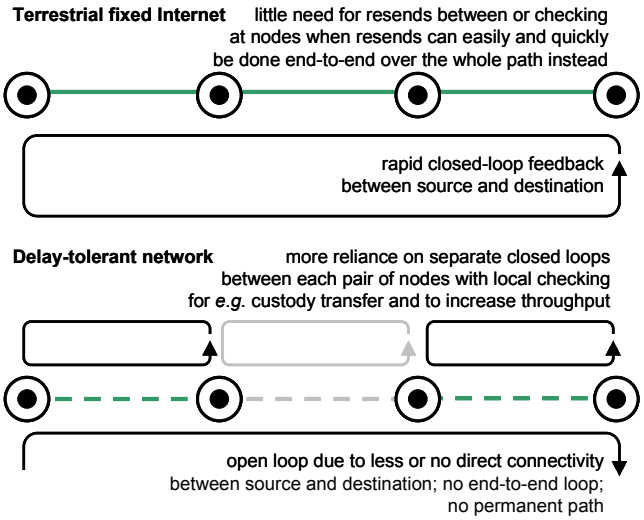


Figure 2 – comparison of control loops

Error detection is a very basic networking concept that was apparently purposefully left out of the Bundle Protocol design, for both its header (metadata) and payload data.

Without useful error detection, the Bundle Protocol's custody transfer mechanism cannot guarantee that a custodian bundle agent taking responsibility for final delivery of a bundle has actually received an uncorrupted copy of that bundle to send on. We demonstrate that this is a severe performance and resource usage issue in DTNs for applications relying on reliably-transmitted and uncorrupted payload data. This also becomes an issue when bundle headers, or *metadata*, are unexpectedly corrupted in transit, leading to misdelivery, misinterpretation of data fields, or other mishandling.

The rationale for the omission of error detection has been described as based on the fact that not all applications require error-detection or data integrity, and that applications can provide these features themselves. That rationale ignores the need for any protocol to guard and verify the reliability of and test its own header information, which the sending application using the protocol does not know about – in this case, the Bundle Protocol's own metadata. The Bundle Protocol's own headers and metadata should be verified by using checksums covering each block's header in order to achieve this verification and protection against corruption.

Leaving error recovery up to the applications is only possible when the applications are tightly coupled across the network, with a tight control loop for resends of errored data. DTN networks, by their ad-hoc nature, are loosely coupled, and there may not be any direct communication or control loop between applications at end nodes, requiring increased assistance from the network to improve performance. This is shown conceptually in figure 2.

The well-known *end-to-end principle* [13] has been used to justify both leaving everything to the applications, and the

alternative of involving the network in recovery and resends, as the principle both implies that each application must always provide its own sanity-checking fit for purpose as the ultimate fallback, and at the same time that if significant performance gains are obtainable, then the lower layers should also implement a version of these features to improve overall performance across the network. To quote [13]: "Clearly, some effort at the lower levels to improve network reliability can have a significant effect on application performance."

Optional security extensions to the Bundle Protocol [14] have been proposed as a way to enable error detection. It is true that some error detection can be accomplished through security protocols that provide keyed checksums across their payloads, as possible in the Bundle Security Protocol. Using the Bundle Security Protocol to implement error-detection has some drawbacks, in that different responses are likely required between legitimate errors and attacks, and checking the reliability of secured payloads is not possible at intermediate agents that have the necessary keys withheld from them.

The *mutable canonicalization* rules of the Bundle Protocol mean that coverage of metadata and the primary and payload blocks can vary considerably depending on the rules used. Using a ciphersuite that requires a private key to decode or check header fields that are included via canonicalization means that protection of those headers against corruption is is only available where that key is available.

We have proposed a workaround to add reliability into the existing protocol infrastructure. This approach uses the bundle security specification and 'wraps' the bundle payloads and non-mutable header data in the primary bundle block using an insecure "null-keyed" reliability-only ciphersuite using a well-known key, rather than an actual security ciphersuite that only provides a reliability check as a side-effect of security [15]. The null-keyed ciphersuites provide integrity protection against non-hostile causes of errors, such as bugs in reassembling fragments or accessing memory, corruption of memory due to radiation events, etc, rather than providing security services such as non-repudiation or sender authentication.

Like the Bundle Protocol, the Licklider Transfer Protocol (LTP) [16], a bundle convergence layer developed to carry bundling over private Consultative Committee for Space Data Systems (CCSDS) space links, did not include inherent error detection, but did define optional security extensions. The idea of using a shared well-known key to repurpose security for a reliability function originated here [17]. This makes implementing the optional security suite mandatory for reliability purposes. LTP is able to use this successfully because its purpose is to provide point-to-point communication without other nodes in the middle. We show that reuse of LTP's approach to reliability for the bundle protocol is weak, because checking the reliability of bundles at bundle agents in the middle of the network, between source and destination, is needed to increase performance across challenging networks. This also ignores the possibility of errors in storage, processing or reassembly internal to a bundle agent and host.

These reliability and performance problems are illustrated in figure 3, where two bundles travel across a DTN network with intermittent link connectivity indicated by the dashed lines. The dark bundle contains a Payload Integrity Block (PIB), signed with a security ciphersuite of key known only to the source and destination. The light bundle has a PIB using an insecure null-keyed reliability-only ciphersuite, that always uses the well-known key contained within its specification, and which can be checked by any node knowing that well-known shared key.

It does not matter whether the dark bundle is confidential with a Payload Confidentiality Block (PCB) or merely a PIB authenticated only at endnodes; what matters is how widely its key is known.

In steps *a*. and *b*., custody transfer takes place. The bundle agent at the second node takes ownership of the bundles, but can only check the light reliability-only bundle for receipt without errors. Without knowing the key for the other dark bundle's security association, the reliability of the payload cannot be checked.

The custody transfer acknowledgement sent back to the source as an administrative record takes it on faith that no errors were introduced, and that the bundle was delivered entirely successfully by the convergence layer and any hop-by-hop security, if implemented. That custody agent then transfers the bundles onwards to the next node and bundle agent in *c*. In our example, that node is hit with memory corruption during long-term storage of the bundles in *d*.

The agent on that node is unable to check the integrity of the encrypted payload in the dark secure bundle, and so forwards it not knowing of its corruption. However, the agent has the ability to check the payload of the light reliability-only bundle before sending it on, and so can use the reliability check to detect errors, before sending that reliability-only bundle onwards. This error detection can be used to cause a request to be sent to the custody transfer node for an uncorrupted copy of the reliable bundle.

The corrupted but security-protected dark encrypted bundle travels onwards in steps *e*. and *f*., and is only discarded once it reaches the destination application, which decrypts the bundle payload and attempts to use it. At that point, the application at the endpoint must detect the corruption, which is indistinguishable from an attack, and then re-request the encrypted bundle from the distant custody transfer node in *g*.

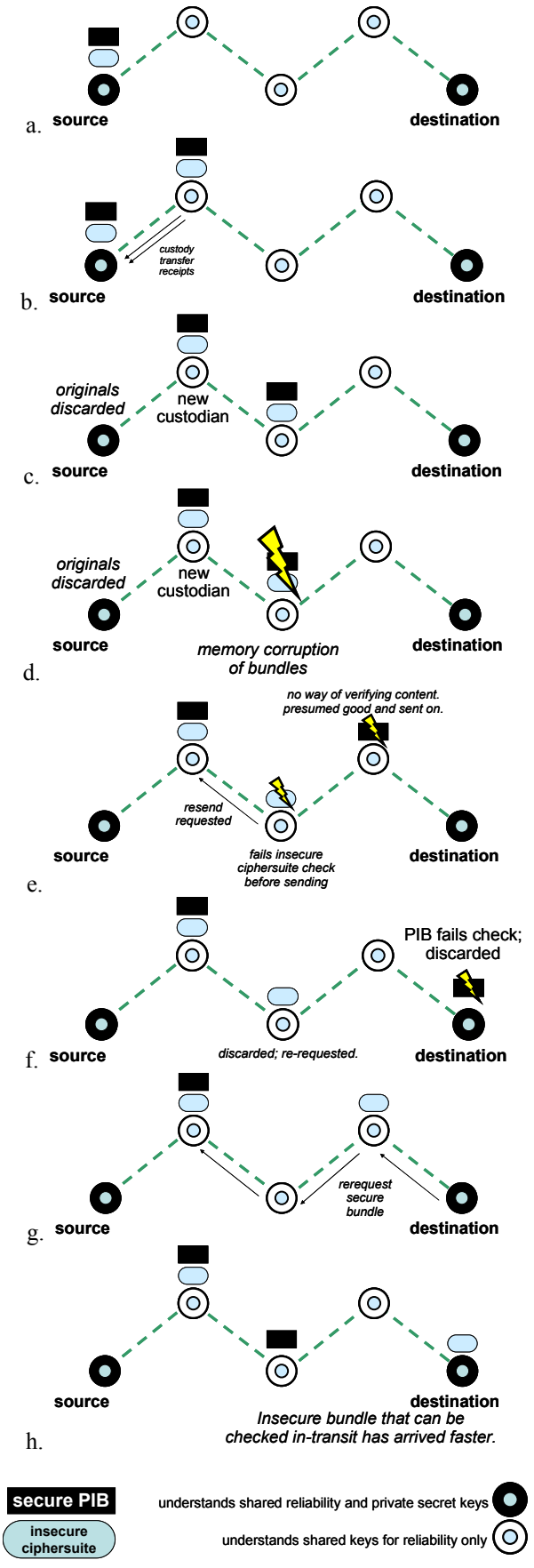


Figure 3 – performance problems with bundling

Meanwhile, the light insecure reliable bundle has already been resent, and a useful copy of that insecure bundle arrives at the destination, before a resent dark secure bundle, in *h*.

This is an example of the parts of the network helping performance of traffic, in accordance with the end-to-end principle, by using separate tighter control loops than application-to-application end-to-end. The applications' control loop is open when there is no direct connectivity between source and destination in an ad-hoc delay-tolerant network. We use the example of memory corruption because it is something that 'reliable' convergence layers or hop-by-hop security protocols cannot address, as it happens outside the links and subnets that they cover. This can happen in memory and storage subsystems internal to a host, and studies have indicated that it is not uncommon [18][19]. (However, it should be noted that requesting retransmission of a bundle may be unacceptable from a security standpoint, as it would be possible for an attacker to create an amplification and redirection attack by sending spoofed bundles, causing retransmissions that consume network resources. That would be addressed by an authentication framework. We discuss this later in the section on security.)

Performance for encrypted payloads can be improved in this scenario by having secured bundles 'wrapped' within an outer reliability checksum block that is applied to the bundle after payload blocks are encrypted. This gives the best of both worlds, providing the benefits of an end-to-end reliability check, error detection throughout the bundle's travels that makes custody transfer meaningful, and the performance boost that comes from tighter control loops for resends within the network that do not involve the application endpoints, while at the same time implementing security by also using the Bundle Protocol's security mechanism. For secured payloads, it would be necessary to nest a secured payload within an outer reliability check, much as an IPSec packet can nestle in an Ethernet frame with a strong Cyclic Redundancy Check (CRC) across the entire packet and frame, so that third-party nodes lacking keys to content can check that they have reliably received and are reliably relaying unknown content. The security mechanisms are completely applied before the reliability check is computed, so that there is no security risk – just as there is no security risk from computing an Ethernet CRC for a frame containing an IPSec packet. Figure 4 shows how existing ciphersuites can be reused to achieve this.

This concept of nesting or 'wrapping' already exists in the Bundle Protocol, for security gateways. Here, we are suggesting that every node is also its own 'reliability gateway, and that whenever a security block with private keys is applied, a reliability block using shared keys should be applied afterwards, covering the security block, so that the robustness of the bundle can be checked in transit in the network. Alternatively, another approach to gain the

performance of reliability with end-to-end security is for applications to implement their own end-to-end-security, tunnelling their own encrypted application payloads through the Bundle Protocol, using the Bundle Protocol's security mechanisms only for reliability, and not nesting security blocks. Not relying on the security mechanisms offered by the Bundle Protocol can increase overall performance in this case.

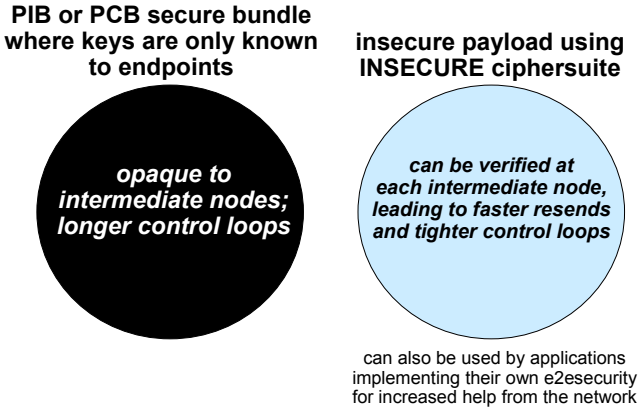
A reliability block using an insecure cipher, but with a zero-length payload, will still cover and protect several non-mutable header fields in the primary payload block. This is a complex way of implementing a header-only checksum.

In [4] it is claimed that reliability is best left up to the application. In an ad-hoc network, there is unlikely to be a direct end-to-end connection between applications, leading to an open loop rather than the closed loop expected in terrestrial networks, and poor resulting performance, as shown in Figure 2. [4] does not distinguish between the need to always protect headers (metadata) and optionally protect payloads if the payloads cannot tolerate errors. It would be useful if custody transfer provided that facility, confirming that metadata was received correctly, and optionally that bundle payloads were received correctly, too, if protection against payload errors is required. We expect that protection against payload errors is desirable and will be more common than the alternative. Protection against errors in metadata is clearly very desirable, given the expense of mishandling, and the need to use the DTN's limited connectivity efficiently.

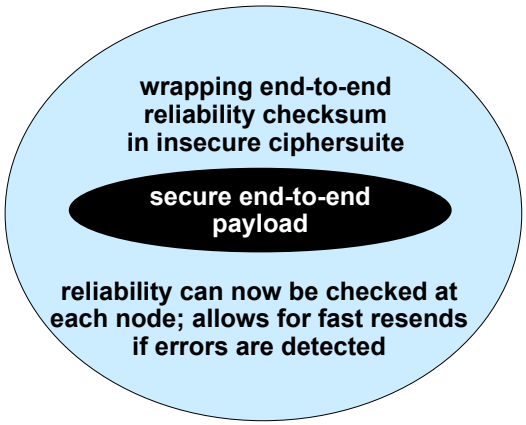
If leaving reliability up to the application is reasonable to do for DTNs, as [4] claims, without assistance from the network to increase performance, then by that same logic security is best left solely to the end-to-end applications as well, and should not be contributed to by the network. And, for the existing Bundle Security Protocol and our reliability work leveraging that, applications implementing their own security and only using reliability ciphers in the bundle network can get increased performance from the bundle network, as we have outlined in our scenario.

[4] makes an analogy with IP removing the header checksum when going to IPv6, leaving error-detection to higher layers checking the IP pseudo-header checksum. That is presented as a justification for leaving error checking out of the Bundle Protocol, but can easily be argued against. First, IP in a terrestrial network exists on an end-to-end path where there is a tight control loop, as shown in figure 2. If an IP packet is received errored at the destination, the request and resend can happen quickly thanks to that tight, closed, end-to-end control loop.

The IPv4 header checksum was removed from the IPv6 design to prevent repeated computation of the checksum due to changing fields such as the time-to-live (TTL) count.



a. Existing use of secure and insecure ciphersuites



b. proposal to combine wrapping of ciphersuites for increased performance from the network

Figure 4 – Comparison of approaches to reliability giving different performance from the network

It would have been possible for an IPv6 header checksum to be designed to only cover 'non-mutable' fields. It is possible for routers to check the pseudo-header checksum to verify the reliability of each IP packet, but this is not done in practice due to the following reasons:

- emphasis on processing speed,
- the carriage of unfragmented IPv6 packets in outer framing such as Ethernet, with strong cyclic redundancy checks (CRCs),
- the lack of explicit fragmentation of IP meaning that reassembly errors are normally not a concern for higher layers, and
- awareness of the tight control loops that make resends easy once the pseudo-header is checked and found wanting at the endpoints.

DTN networks do not have the tight end-to-end control loops of terrestrial IP networks. In a DTN there may be no direct connectivity between endpoint applications at all.

DTN networks are expected to have significant amounts of fragmentation of bundles, both at the sending nodes (called *proactive fragmentation*) and within the network (called *reactive fragmentation*). There is a need to check

reassembly of fragmented bundles. By contrast, fragmentation of IPv6 packets is only possible at the sending endpoint, and *en-route* IPv6 packets can not be fragmented once in the network.

Sending data in many DTN networks can be expected to be more expensive in terms of slow rates, power use, etc., leading, as our example has shown, to a distinct performance advantage to nodes able to use a reliability check on each bundle. Network architectures with different designs for different conditions have a different set of tradeoffs. Appealing to a design for a terrestrial IP network to justify design choices for a different network, where control loops and tradeoffs are entirely different, is not justifiable.

In our experiments with an operational DTN satellite sensor network, we have implemented an optional MD5 checksum for the *Saratoga* protocol to provide a measure of reliability checking. This can be used to compare hash values of files before and after downloading. This provides protection in the convergence layer over the most error-prone hop between bundle agents, but not end-to-end between the application source and sink. On the relatively low-end processors typical of spacecraft or embedded devices, that MD5 computation can take several minutes to run over a large file, so it is likely to be used sparingly onboard. Given that image data is often downloaded in 'one shot' before being deleted to make room for new images, and postprocessed heavily with human inspection, the need to resend image files with slight corruption is minor, although knowing where that corruption may lie in the image data would be useful. However, overall reliability checking becomes very important when *e.g.* uploading code to be executed.

4.2 Time synchronization problems

The Bundle Protocol assumes that all communicating bundle nodes share a common, simultaneous, synchronised, conception of UTC time so that its timestamps can be interpreted and handled correctly. There are three primary goals for timestamps in the Bundle Protocol. These goals relate to network and agent resource protection and efficiency:

- the Lifetime keeps bundles from looping continuously throughout the network due to routing loops, similar to IP's Time-To-Live (TTL) counter;
- the Lifetime spare the network from storing and propagating bundles after a time that the sending application has designated the data as no longer useful.
- The Creation timestamp information can uniquely identify the bundle, necessary for assembly of bundle fragments. This is a similar role to the Identification field in the IPv4 header.

During our initial testing in terrestrial networks, it became clear that network time synchronization is critical. That is probably not a reasonable requirement for many DTN networks, as nodes in many DTN networks will be isolated and disconnected for long periods of time.

Furthermore, the Bundle Protocol is a network overlay, and one that may be running on top of ad-hoc networks in highly stressed environments, effectively at the application layer as far as the ad-hoc network is concerned, The requirement that all DTN networks running the Bundle Protocol must be synchronized to enable interoperation is not necessarily one that is either practical or deployable.

Our clock synchronization problem with bundling was experienced during initial terrestrial use of the Bundle Protocol. All of our DTN bundle agents were originally configured and tested at NASA GRC in Cleveland, Ohio. One bundle agent was sent to Guildford, England. A second was sent to Universal Space Networks (USN) in Alaska. When performing initial bundle transfers from SSTL to GRC to USN across the public Internet, it was noticed that the machine clocks had drifted sufficiently enough during shipping to result in the bundle creation time stamps being out of synchronization. Bundles generated by the "dtnping" application used for configuration testing were therefore rejected due to lack of tight time synchronization between system clocks, leading to unexpected early expiry of the bundles. Once the machines were resynchronized with a common clock reference, bundle transfers operated correctly.

With scheduled low Earth orbit passes over a ground station, it is necessary to know what the time is to support the pass opportunity. However, in our initial testing of the Cisco router in Low Earth Orbit [20], nodes in the field at Vandenberg were still able to operate with clocks set several minutes adrift; the loosely-coupled architecture used there tolerated this.

It appears possible to reduce problems related to time synchronization, that we discuss in this and following sections. This would be done by making small modifications to the primary bundle block format and/or changing the semantics of the creation timestamp and lifetime field to more directly implement their three goals. However, proposals to do so have not been fully fleshed out or evaluated by the community.

4.3 Problems in learning the current time

Expecting DTN nodes with loosely-coupled ad-hoc connectivity to be tightly coupled with respect to their understanding of clock time has interesting ramifications.

A side effect of requiring synchronized clocks is that it would not be possible for a node to learn the correct time using the Bundle Protocol, as its bundles sent asking for the time are likely to be judged expired or invalid and be discarded. Another protocol would be required to do clock

'housekeeping'. One approach to implementing this other protocol would be to have nodes assign clock confidence levels to themselves. A node that has rebooted would have a clock time of earlier than a starting date of *e.g.* 1 January 2008, and a confidence level of zero. Nodes with their own internal clocks would have higher confidence levels, depending on the accuracy of the clock. Nodes with low confidence levels can receive and accept current time from authenticated nodes that they trust with higher confidence levels.

Furthermore, for network-centric operations involving diverse organizations, it may not be possible from a security standpoint to accept time reference data from nodes operated by a different organization, even though data communications with that organization are deemed acceptable. Authentication of the provenance of the time information received over some protocol other than the Bundle Protocol, and outside of the Bundle Protocol's authentication mechanisms, is an interesting problem.

Time synchronization for interplanetary use was noted as a problem by Vint Cerf [21]. Others have noted similar problems with synchronization [22]. Problems caused by differences in clock times due to Einstein's relativity are unlikely to be noticed in any near-term deployment.

443 The selected time standard

The Bundle Protocol uses Coordinated Universal Time (UTC), where leap seconds are added at irregular, unpredictable, intervals to reflect slowing of the Earth's rotation. For nodes 'in the field' for a long time (decades), some way of communicating newly-decided UTC leap seconds will be required to prevent clock drift over long time scales that would eventually lead to bundles expiring before delivery. This is most likely to be a significant issue for real-time traffic with very short bundle lifetimes.

4.5 The roles of convergence layer adapters

Direct TCP and UDP convergence layers are in already in experimental use for carrying bundles across the terrestrial Internet, though these convergence layers are not yet agreed on by implementers and are still in the process of being documented [23][24]. Different UDP implementations are currently only compatible when sending bundles that fit within single UDP datagrams without convergence-layer fragmentation.

The simple UDP convergence layer implemented in existing DTN software, including DTN2 *dtnd*, is unreliable, providing a bundle agent with performance that differs considerably from a convergence layer supporting reliability, such as TCP, *Saratoga* or LTP. The community has yet to determine whether, and how, the UDP convergence layer should comply with recommendations for UDP-based protocols [25].

The more well-defined convergence layers have differing intended operating environments and underlying stack dependencies. *Saratoga* relies on UDP, while LTP is intended to be used over CCSDS protocols. LTP is supported over UDP as well for testing; LTP would not support a pseudo-header check, discouraging carrying LTP directly over IP. UDP and IP can also be carried by CCSDS protocols in a variety of ways, but have not traditionally been used by most CCSDS-based space missions. Other ways of building upon UDP are also possible, e.g. the Uni-DTN convergence layer for unidirectional links, which relies on the FLUTE erasure coding UDP multicast protocol [25].

To complicate matters, convergence layer adapters have been proposed and written that run directly over data links, *e.g.* Ethernet or Bluetooth, as well as over file-based portable storage media like USB "thumb-drives".

These convergence layers obviously have widely varying expectations and properties that the bundle agents can depend or rely on, leading to differing behaviour at the Bundle Protocol level. (The potential for use with long-term storage media, rather than with store-and-forward network protocols, was one motivation for including timestamps using a universal clock within the bundle protocol.)

The possibilities for carrying the Bundle Protocol over a variety of convergence layer adapters are shown in figure 5.

A further challenge and complication to the scoping of a convergence layer adapter's role is that some convergence layer adapters contain their own store-and-forward capabilities in addition to those of the bundle agent. *Saratoga* is one example, with file storage actions. LTP is another such example. An LTP-T variant has even been proposed as a way to add source-routing capabilities to LTP which can blur the need for a bundle agent in some deployment scenarios [27].

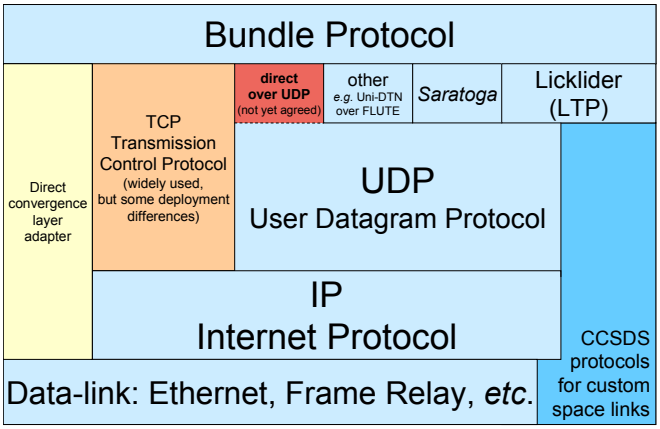


Figure 5 – Major convergence layer choices

This is similar to the concept of "extended operations" in the CCSDS File Delivery Protocol (CFDP) [28], which itself bears several similarities to the capabilities touted for a bundle agent in providing reliable store-and-forward services. A space probe using CFDP would not need bundling, and *vice versa* – unless there is a desire to use one or the other's programming interfaces or interoperate with other systems based on either CFDP or bundling.

It is interesting that, although bundling is intended to work over a wide range of networks and protocols via convergence layers, most of the use and development of the Bundle Protocol has been over IP. IP provides a shared working base that is popular, widely implemented, and well-understood. Furthermore, since "IP over everything" is already well-accepted and numerous research and development projects are continuously extending the breadth of IP networks, it makes for IP-based convergence layer adapters to be specified and documented and for bundle agents to bootstrap configuration parameters from the IP-based convergence layer protocols whenever feasible.

Using IP-based convergence layers, rather than attempting to map the bundle protocol directly onto link protocols, makes sense architecturally, because it limits the number of bundle protocol specifications for mapping that have to be created and documented. The engineering and development costs are contained. Since IP-to-datalink mappings are either already written, or are typically concurrently written with the development of new datalink technologies, the availability of standard IP-based DTN convergence layer adapters leads to "DTN over everything" by proxy, with very little additional systems integration or expense required. Literally hundreds of documents could be written over the years for carrying the Bundle Protocol over a variety of different data links with different convergence layer adapters. This is avoided entirely if IP-based convergence layers, capable of working in a broad set of environments, can be adopted.

Configuration information, e.g. TCP port numbers, IP addresses, *etc.*, is needed for bundle agents to work with convergence layer adapters, along with an indication of what bundle agents can be reached through the convergence layer adapters. This information would be provided and exchanged as part of a routing protocol, but is currently entered manually in experiments with DTN software. Methods for bundle agents to autoconfigure and autodiscover each other in IP networks have not been pursued by the DTNRG, though some groups have experimented with different techniques. It is not clear that aiding in configuration belongs within the role of the convergence layer adapters or not.

4.6 Maximum transmission sizes and fragmentation

There is currently no method for advertising or negotiating the maximum size of a bundle that can be accepted by a bundle agent for storage and delivery, so that bundles that are too large can be rejected. This can also lead to fragmentation of bundles in the network. Handling reactive fragmentation, with delivery and reassembly of bundles split in the network, whose fragments take different routes to the destination, is an open problem with reliability implications.

The interactions and effects of fragmentation on other parts of the DTN architecture are not yet well explored. For instance, reactive fragmentation is known to cause complications with the proposed bundle security protocol [14] and with custody transfer. A solution for allowing the security mechanisms to function in conjunction with reactive fragmentation has not yet been selected, though there are proposals for either sending signatures covering fragments at a later time, or for proactively including multiple authentication codes across pieces of the bundle (called the "toilet paper" scheme [29]). These have advantages and disadvantages that must be weighed. Custody transfer of fragments has been discussed [10], but is not part of the Bundle Protocol specification.

We believe that useful bundles will be large, and have experimented with sending bundles of sensor data of over one hundred megabytes in size from space. We expect large fragments to be common and useful. Schemes to implement reactive fragmentation should be carefully considered.

4.7 Agreement on naming schemes

Different Bundle Protocol implementations are currently supporting multiple different naming schemes for Bundle Protocol Endpoint Identifiers (EIDs), with different rules for forming and interpreting EIDs. This is not necessarily a bad thing, as different operational environments may have different requirements for naming and addressing, and it avoids the difficulty of trying to create a one-size-fits-all scheme. However, it does lead to some interesting open research questions.

The built-in naming flexibility gained by using a generic Uniform Resource Identifier (URI) format for Bundle Protocol EIDs remains to be put to its full use. The URI scheme is intended to indicate how the remainder of the EID string should be parsed. However, the DTNRG has not yet rigorously specified or adopted any common EID schemes, and additionally, rules for what to do with unrecognized EID schemes have not been defined.

A basic scheme that facilitates initial testing and implementation would be helpful, and would provide a common base for which multiple implementations could be expected to interoperate regardless of their support for other EID schemes. As routing to destinations is meant to be based on EIDs, a common EID format becomes a prerequisite for routing between different DTN networks.

Since the EID is the sole basis to identify the upper-layer protocol machines above the bundle protocol, a common property of all EID schemes seems to be the need to mux and demux traffic between different protocol machines registered to the same bundle agent. This would be accomplished via a mechanism similar to the next-protocol field in IP packets, or the port numbers and service codes in Internet transport protocols. If common EID schemes can not be defined for some reason, then at least defining a standard way to include mux/demux tokens within an EID would be useful.

An EID specifies a group of bundle agents, rather than a single agent. How bundles are delivered successfully to this group, so that the equivalent of multicast can be carried out and applications can rely on sending to group EIDs, is an open question.

4.8 Standardization of routing methods

The need for common routing protocols and address resolution techniques is related to the issue of common EID schemes for naming of destinations. An EID is essentially an identifier with the late-binding operation that determines the route a bundle takes. Resolving a destination EID into one of a local registration, a next hop bundle agent, a convergence layer adapter, or even a particular convergence layer adapter address are all possible paths within a bundle agent's processing. As no resolution protocols comparable to DNS, ARP, or SIP, and no routing protocols with features similar to RIP, OSPF, or BGP have yet been defined, there is much fruitful work to be done in this area of the DTN bundling architecture. A large challenge would be in defining mechanisms that are independent of a particular convergence layer adapter or operational environment. If this is not possible, then it makes "DTN over everything" more difficult, as routines for such operations have to be created per-deployment.

Forwarding without any automated routing or resolution protocol is possible through several means:

- if static routes are configured at each node, which is the antithesis of the ad-hoc DTN networks that bundling is intended for.
- if source routing is used, perhaps as a new bundle option.
- if the EID scheme itself implies forwarding rules somehow through clear use of hierarchy, which can be thought of as a form of source routing.

Automated routing protocols increase scalability, reduce operations and management overhead, and enable operations in completely ad-hoc settings.

It seems likely to us that a number of subnet-specific routing protocols will be needed in order to enable the Bundle Protocol to perform well across the highly diverse range of environments that it is envisioned for. (The Bundle Protocol is already relying upon IP routing protocols to run across the terrestrial Internet.) This again represents the utility in clearly specifying and documenting IP-based convergence layer protocols, because it means that existing IP routing and resolution mechanisms can be employed without re-inventing the wheel and re-learning painful lessons in routing for DTN.

Interconnecting different DTN networks poses problems with gateways and sharing of routing information, possibly leading to the separate internal and external routing models used by the Internet – which is vastly complicated by the late binding to addresses of EIDs. With late binding, mapping EIDs to individual subnetworks can be problematic and even dangerous depending on the properties of interconnection between subnetworks and the mobility of EID owners between subnetworks.

The concept of using "regions" as one component of an EID was proposed as a way to distinguish particular subnetworks in an EID. However, this is very problematic in the general sense because it leads to difficulties in multihoming by naming an interface (or set of interfaces) rather than a host (or application process, or set of application processes), as was known already during the early Internet build-up [5]. There has been question of whether a region relates to either a common field of control, a high probability of interconnection, or a spatial area. In some discussions, the notion of regions has been attempted to be supplemented by the notion of "domains", but there is still a lack of clarity in what these terms imply and do not imply, and not yet agreement on how to handle inter-domain, inter-region, inter-EID-scheme, or other types of routing decisions.

Agreement on a very basic routing protocol that simply aids in testing and debugging but is not expected or required to perform optimally (similar to RIP for IP), would be useful in these early phases of DTN test and development. Methods for auto-discovery of bundle agents have been proposed and tested, but not yet fully adopted in the DTNRG. Building on auto-discovery, methods of distributing advertisements of routes and predicted contacts would greatly increase the capabilities of the Bundle Protocol and bring it closer to the state needed for operational benefit.

4.9 Network management

DTN nodes currently have no support for remote management, which is common in IP networks.

For an operational DTN, it would be very useful to have some type of network management capability, similar to the features of the Simple Network Management Protocol (SNMP) in IP networks. This capability could be used to report on node health, storage issues, undeliverable bundles,

performance data, and so on. It could be used to remotely (re-)configure a bundle agent through sending network management bundles to conditionally fetch and set configuration parameters.

A powerful network management protocol might even be able to share functionalities with a DTN routing protocol, as it could be used to add/remove and enable/disable routes on the bundle agents under control.

However, the long delays and link disruptions that increase or break end-to-end control loops in DTN networks also make network management more difficult. It is likely that network management would be subnet-specific, using a subnet-specific protocol, *e.g.* SNMP over IP, rather than the Bundle Protocol itself. Management of the overlay running the Bundle Protocol is distinct from management of the underlying network. Combining these into a single system has a number of technical and other tradeoffs.

Little work has yet been done on DTN network management, though it seems to be essential in some proposed scenarios where DTN bundle agents are to be operated as long-term infrastructure elements. Our experiences with problems in time synchronization suggest that DTN network management based on the Bundle Protocol will be somewhat complicated, as bad configurations may be impossible to fix without using bundles with very long lifetimes and/or spoofed creation timestamps – which would be undesirable in the network.

4.10 Quality of Service

IP network infrastructure supports an array of Quality of Service (QoS) mechanisms and mappings between different subnetwork QoS mechanisms and IP QoS mechanisms. The bundle protocol has defined some bits to indicate one of three priority levels in a bundle, but the semantics of these levels are as yet undefined, so they cannot be effectively used unless the handling behaviour is locally agreed on and supported by all bundle agents.

Mechanisms for identifying, labeling, policing, and otherwise providing QoS for bundle flows are not yet part of the DTN architecture. The aspects of DTN QoS differ from IP QoS as DTN makes heavier use of in-network storage and has more complex routing decisions. The interactions between late binding and QoS reservations may also be complex.

One simple proposal is to consider a bundle flow to consist of all bundles between the same two EIDs, and then use the bundle priority field in per-flow priority queuing. This would guarantee certain forwarding behaviours within applications' data flows, but still leaves the more difficult problem of scheduling contention between flows as a matter of local policy.

4.11 Efficiency of protocol overhead

The Bundle Protocol can be efficient in terms of metadata overhead if bundles can be made large, but this is not natural for all applications.

Our experiments from orbit transferred a 150MB image file as a fragmented payload. There, the added overheads of the bundle format were trivial, and were outweighed by the overheads of *Saratoga*/UDP/IP packet headers in the convergence layer.

However, some convergence layers, *e.g.* raw UDP implementations, cannot handle large bundles, expecting a bundle to fit into a UDP packet, and be less than 64KB in size.

Recent discussions have questioned the suitability of DTN for "small payload" transmissions such as real-time voice codecs. This still appears to be an open issue.

4.12 Complexity and performance

The complexity of the Bundle Protocol's design is shown by its variety of optional fields, structures, and novel binary formats [30].

The use of variable-length, rather than fixed-length, fields with Self-Delimiting Numeric Values (SDNVs) and Endpoint Identifiers, and the referencing back to the dictionary in the primary block from later payload blocks can make the Bundle Protocol more susceptible to parsing errors from corruption than a more robust fixed-length format. In a fixed-field format, an error in a field may affects only that field. In variable-length SDNVs, a single bit of each byte indicates whether the SDNV continues for another byte. An error introduced into any of these bits of an SDNV not covered by a canonicalization will affect interpretation of that and everything that follows.

This complexity, including concepts such as the mutable canonicalization rules used by security (and thus inherited by reliability), can be considered as a hurdle for implementation, interoperability, and adoption – especially for those pieces of the design that have not yet been fleshed out or agreed.

It would be difficult to be as ambitious and all-encompassing as the Bundle Protocol and not be complicated. However, much of that complexity lies in the security protocols, on which much time and effort has been spent.

4.13 Security

The bundle security mechanisms are not yet finalized as we write, and some aspects of them remain to be completed. As these are not yet finished, or fully evaluated in practice, it is speculative for writers to make strong claims about the

overall security properties of the bundling architecture. An overview of those properties is given in [31]. There are currently drafts defining methods to protect the integrity of bundles between two hops and bundle payloads end-to-end, as well as the ability to make a bundle block confidential. These all rely on an assumption of shared keying material. No method for automatically sharing this material is yet defined within the bundle architecture, or proposed in any DTNRG draft. Key management has been recognised as an open problem [32].

The difficulties of applying typical IP security techniques, such as established certificate systems, in DTNs are well-known, with several research issues open and work to be done if capabilities similar to those provided by IKE in the Internet are to be available in DTNs.

Another interesting aspect of current Bundle Protocol implementations is the lack of any features similar to the firewalling or access control lists of typical IP networking devices. In operational practice, these are absolutely essential in contemporary IP networks, even though they often result in frustrations when deploying new services and mobile devices. In DTNs with the emphasis on late-binding it seems that this may be even more problematic, especially since filter rules depend heavily on address formats and DTN EID schemes are currently not fully-specified and the architecture permits (and even encourages) a proliferation of differing formats. There is interesting work to be done in this area of how filter implementations can accommodate diverse EID schemes.

The logic for building some means to authenticate bundles has been justified as necessary to protect against denial-of-service attacks against a bundle agent's resources. This has been shown to not be as large a threat in practice as originally assumed [33].

We believe that a larger threat consuming network resources exists in inadvertently forwarding errored, unchecked, bundles that can only be checked and discarded at the destination application. This problem is prevented by the reliability checks discussed earlier, and by introducing a check across bundle contents before acknowledging a custody transfer or on reassembly of fragments or retransmission.

4.14 Content identification

The Bundle Protocol does not identify the content it carries to select an application to hand the content off to. There is no notion of something similar to an IP port number or protocol ID, or type field, that can be used to pass bundles to higher-layer protocols or applications. This can lead to each EID scheme also supporting some way of indicating applications through the EID, with every application appearing as its own bundle node in the EID space – a problem reminiscent of creation of all the vanity domain names for webservers in the Internet's DNS.

It can be argued that the web and email have become successful at delivering content partly because it is so easy to determine what application should be invoked to receive a delivered file, due to their universal adoption of MIME [34]. It is reasonable to expect the Bundle Protocol to adopt MIME as well.

5. OTHER APPROACHES TO DTN NETWORKING

Just as *an internet* consists of multiple connected networks, and *the Internet* has special meaning implying certain protocols used in a certain way, delay-tolerant networking environments are larger in scope than the single DTN architecture that has been proposed for them by the DTNRG. The Bundle Protocol is just one single approach to addressing the problems posed by delay-tolerant networking. Other approaches to delay-tolerant networking, that do not require the Bundle Protocol, are possible.

We suggest a simple approach, leveraging existing standards, using the Hypertext Transfer Protocol (HTTP) as a transport-layer-independent 'session layer' between each two communicating DTN nodes, hop-by-hop, as shown in figure 6 [35]. HTTP is well-understood by applications in the same way that IP is well-understood, easing adoption. New *Content-Source:* and *Content-Destination*: headers are added, which provide routing information end-to-end. Content- headers are treated specially by HTTP: HTTP servers must reject transfers with unknown Content- headers. Adding these two new headers creates a separate DTN network that will not interact with or affect existing traditional web use of HTTP. Reuse and implementation of HTTP in this way to create HTTP-DTN appears straightforward.

Fitting HTTP to *Saratoga* for long-delay or private networks is possible; HTTP does not contain timers, and with persistence, pipelining and unidirectional PUTs, can work over long distances. HTTP is already widely used over TCP across the shared, congested, Internet, and has been proposed for use over SCTP, separating HTTP from the underlying transport [37].

The two bundle hops used in our operational sensor satellite scenario – transport of the bundle over *Saratoga* from the UK-DMC's SSDR computer to the bundle agent in a computer in the ground station, then transport of the bundle over TCP across the Internet to NASA Glenn – would be replaced by two HTTP-DTN hops: HTTP-DTN transfer of the image file over *Saratoga* between satellite and ground station, then an HTTP-DTN transfer over TCP between ground station and NASA Glenn's computer.

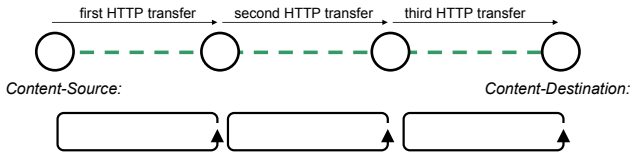


Figure 6 – HTTP-DTN transfers end-to-end

The *Content-Destination:* header would be set to indicate the domain name of the destination NASA Glenn computer. In applying HTTP-DTN to our test scenario, the destination is resolved with late binding on the last HTTP-DTN hop, which is across a subnet that understands that name using DNS. (A static route is used on the wireless first hop for traffic from satellite to ground station; everything goes down the downlink.)

HTTP provides the ability to easily transfer content identified by MIME. This provides the necessary content identification that we have identified as missing from the Bundle Protocol. Payload reliability can be addressed by using the existing *Content-MD5*: header. HTTP has a number of existing security protocols that could also be evaluated for suitability for reuse in unusual DTN conditions, on a case-by-case basis.

HTTP-DTN shares some of the same problems that we have articulated for the Bundle Protocol – particularly the assumption of synchronized time across nodes that makes timestamp and *Expires:* headers useful, and problems with ensuring end-to-end reliability of headers and indicating the maximum size of MIME payloads that can be accepted. However, by leveraging existing well-implemented technologies, HTTP-DTN is far less work to specify, understand, and implement than the Bundle Protocol, which is a significant benefit.

We have given HTTP-DTN as an example of an alternative to the Bundle Protocol because we are familiar with it and details have been worked out. Other alternatives, considering concepts implemented by uucp, netnews, message-oriented middleware, or Fidonet (where "bundles" were proposed [37]), could be fleshed out to be similarly viable solutions for delay-tolerant networking scenarios.

CONCLUSIONS

Our practical experience gained with implementing and operating the Bundle Protocol from space over an IP-based sensor network enables us to consider aspects of the Bundle Protocol's design.

The lack of integrity checksums for reliability checks in the Bundle Protocol and the need for network time synchronization were found to be real deployment issues during our first tests, and we are investigating new checksum mechanisms to increase the performance and reliability of the Bundle Protocol.

We have described a bundle of problems with the Bundle Protocol as it currently stands. Though it is far easier to criticise than to create, and easier to analyse something that is being fleshed out in detail than it is to synthesize something entirely new, it is hard to avoid recognising some problems with the Bundle Protocol and its current design.

We hope that recounting those problems here will lead to them being addressed in later versions of the DTN architecture and Bundle Protocol specifications, as research into delay-tolerant networking continues and these research ideas are matured. Once these are addressed, the Bundle Protocol may one day be ready for operational use.

ACKNOWLEDGEMENTS

We thank our colleagues for thoughtful discussion during the gestation of the ideas presented here.

REFERENCES

[1] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," SIGCOMM, August 2003.

[2] V. Cerf *et al*., "Delay-Tolerant Network Architecture," IETF RFC 4838, informational, April 2007.

[3] K. Scott and S. Burleigh, "Bundle Protocol Specification," IETF RFC5050, experimental, November 2007.

[4] K. Fall and S. Farrell, "DTN: an architectural retrospective," Journal of Selected Areas in Communications, vol. 26 no. 5, pp. 828- 836, June 2008.

[5] J. Day, *Patterns in Network Architecture*, Prentice Hall, 2007, ISBN 0-13-225242-2.

[6] L. Wood, W. Ivancic, W. M. Eddy, D. Stewart, J. Northam, C. Jackson and A. da Silva Curiel, "Use of the Delay-Tolerant Networking Bundle Protocol from space," paper IAC-08-B2.3.10, 59th International Astronautical Congress, Glasgow, September 2008.

[7] "UK-DMC satellite first to transfer sensor data from space using 'bundle' protocol," Surrey Satellite Technology Ltd press release, 11 September 2008.

[8] L. Wood, J. McKim, W. M. Eddy, W. Ivancic and C. Jackson, "Saratoga: A Scalable File Transfer Protocol," work in progress as an internet-draft, draft-wood-tsvwg-saratoga-02, October 2008.

[9] L. Wood, J. McKim, W. M. Eddy, W. Ivancic and C. Jackson,, "Using Saratoga with a Bundle Agent as a Convergence Layer for Delay-Tolerant Networking," work in progress as an internet-draft, draft-wood-dtnrg-saratoga-04, October 2008.

[10] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho and R. Patra, "Implementing Delay Tolerant Networking," Technical Report IRB-TR-04-020, Intel Research Berkeley, December 2004.
[11] DTN reference implementation, October 2007 release, available from http://www.dtnrg.org/wiki/Code
[12] D. Waitzman, "IP over Avian Carriers with Quality of Service," RFC 2549, 1 April 1999.
[13] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277-288, November 1984.
[14] S. Symington, S. Farrell, H. Weiss and P. Lovell, "Bundle Security Protocol Specification," work in progress as an internet-draft, draft-irtf-dtnrg-bundle-security-06, November 2008.
[15] W. M. Eddy, L. Wood and W. Ivancic, "Checksum Ciphersuites for the Bundle Protocol," work in progress as an internet-draft, draft-irtf-dtnrg-bundle-checksum-03, October 2008.
[16] S. Burleigh, M. Ramadas and S. Farrell, "Licklider Transmission Protocol – Motivation," RFC 5325, September 2008.
[17] S. Farrell, M. Ramadas and S. Burleigh, "Licklider Transmission Protocol - Security Extensions," RFC 5327, September 2008.
[18] J. Stone, M. Greenwald, J. Hughes, and C. Partridge, "Performance of checksums and CRCs over real data," IEEE Transactions on Networks, vol. 6 issue 5, pp. 529-543, October 1998.
[19] J. Stone and C. Partridge, "When the CRC and TCP checksum disagree," Proceedings of ACM Sigcomm, pp. 309-319, September 2000.
[20] L. Wood, W. Ivancic, D. Hodgson, E. Miller, B. Conner, S. Lynch, C. Jackson, A. da Silva Curiel, D. Shell, J. Walke and D. Stewart, "Using Internet nodes and routers onboard satellites," *International Journal of Satellite Communications and Networking*, vol. 25 issue 2, pp. 195-216, March/April 2007.
[21] J. Jackson, "The Interplanetary Internet," IEEE Spectrum, vol. 42, no. 8, p. 30, August 2005.
[22] W. M. Eddy, "DTN Time Sync Issues," email to the IRTF dtn-interest mailing list, 1 April 2008, and subsequent discussion.
[23] M. Demmer and J. Ott, "Delay Tolerant Networking TCP Convergence Layer Protocol," work in progress as an internet-draft, draft-irtf-dtnrg-tcp-clayer-02, November 2008.
[24] H. Kruse and S. Ostermann, "UDP Convergence Layers for the DTN Bundle and LTP Protocols," work in progress as an internet-draft, draft-irtf-dtnrg-udp-clayer-00, November 2008.
[25] L. Eggert and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers," RFC5405, November 2008.
[26] D. Kutscher, K. Loos and J. Greifenberg, "Uni-DTN: A DTN Convergence Layer Protocol for Unidirectional Transport," work in progress as an internet-draft, draft-kutscher-dtnrg-uni-clayer-00, April 2007.
[27] S. Farrell and V. Cahill, "Evaluating LTP-T: A DTN-Friendly Transport Protocol," Third International Workshop on Satellite and Space Communications (IWSSC '07), September 2007.
[28] CCSDS File Delivery Protocol (CFDP), Consultative Committee for Space Data Systems Blue Book, Issue 4, January 2007.
[29] C. Partridge, "Authentication for fragments," ACM SIGCOMM HotNets- IV workshop, 2005.
[30] W. M. Eddy, "Using Self-Delimiting Numeric Values in Protocols," work in progress as an internet-draft, draft-irtf-dtnrg-sdnv-00, September 2007.
[31] S. Farrell, S. Symington, H. Weiss and P. Lovell, "Delay-Tolerant Networking Security Overview," work in progress as an internet-draft, draft-irtf-dtnrg-sec-overview-05, November 2008.
[32] S. Farrell, "DTN Key Management Requirements," work in progress as an internet-draft, draft-farrell-dtnrg-km-00, September 2007.
[33] J. Burgess, G. Bissias, M. Corner and B. Levine, "Surviving Attacks on Disruption-Tolerant Networks without Authentication", Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), Montreal 2007, pp. 61-70.
[34] N. Freed and N. Borenstein, "Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," IETF RFC 2045, November 1996.
[35] L. Wood and P. Holliday, "Using HTTP for delivery in Delay/Disruption-Tolerant Networks," work in progress as an internet-draft, draft-wood-dtnrg-http-dtn-delivery-02, October 2008.
[36] P. Natarajan, P. Amer, J. Leighton and F. Baker, "Using SCTP as a Transport Layer Protocol for HTTP," work in progress as an internet-draft, draft-natarajan-httpbis-sctp-00, October 2008.
[37] W. Wagner, "A Bundle Proposal," FSC-0014, January 1988.

BIOGRAPHIES

Lloyd Wood *is a Chartered Engineer who serves as a space initiatives manager and technical leader in the Global Government Solutions Group within Cisco Systems. There he has had responsibility for CLEO, the Cisco router in Low Earth Orbit, for over five years. With colleagues from NASA Glenn Research Center and Surrey Satellite Technology Ltd, Lloyd achieved the first tests from space of IPv6 and of the delay-tolerant networking Bundle Protocol. Lloyd has contributed to the Internet Engineering Task Force and modified IOS, Cisco's router software... software now in space on CLEO. Lloyd gained his PhD from the Centre for Communication Systems Research at the University of Surrey, where he researched internetworking and satellite constellations.*

Wesley M. Eddy *is a network architect with Verizon, contracted to NASA's Glenn Research Center (GRC). He supports systems engineering and architecture development for NASA communications systems and technology development for network-centric operations using space and aeronautical mobile networks. He wrote the bundling code demonstrated on the UK-DMC satellite. He is co-chair of the IETF TCPM Working Group and of the IRTF's Internet Congestion Control Research Group (ICCRG). He holds an M.S. degree from Ohio University.*

Peter Holliday *is a technical leader with Cisco Systems' Global Government Solutions Group. He is a retired Army Officer who has served over thirteen years in the Royal Australian Corps of Signals. He received his BSc from Queensland University of Technology, and his MEngSc at the Australian Defence Force Academy from the University of New South Wales, where he is currently undertaking a PhD in military mobile ad-hoc and disrupted networks.*