MP W

MITRE PRODUCT

Secure Multidestination Delivery in DTN

March 6, 2005

Susan Symington

MITRE
Center for Air Force C2 Systems

Table of Contents

List of Figures

Section 1

Introduction

There are currently no provisions in the Bundle Protocol for multidestination delivery of bundle payloads. Ideally, however, a multicast delivery capability inherent to DTN will eventually be defined, complete with a DTN multicast group management protocol to enable the formation of DTN multicast groups, the association of each group with a DTN multicast group address, and the ability to add and remove DTN endpoints from these multicast groups. A native DTN multicast capability will also require the definition of a DTN multicast routing protocol to enable each DTN router to determine on which interfaces a bundle destined for a multicast group address should be forwarded. While such a native DTN multicast capability is desirable, its definition does not seem imminent. In the meantime, before all the components necessary for such a full-scale native DTN multicast capability are in place, it has been proposed that the DTN Bundle Protocol be modified with the optional ability to provide a modified form of multicast, which we will call "multidestination delivery". Multidestination delivery can conserve DTN bandwidth by ensuring that a given payload sent from a single source traverses each link of the DTN at most once, no matter how many destinations to which the payload is addressed.

The multidestination delivery capability being proposed is to allow a single bundle to contain multiple destination fields and thereby be explicitly addressed to multiple destinations. Under this proposal, a source wishing to send a bundle to *n* destinations need only include each of those *n* destination addresses in the destination fields of the bundle and send the bundle once, and the DTN routers along the paths from that source to each of the destinations would, upon receipt of the bundle, look at the destination addresses to determine on which interfaces (one or more) the bundle should be forwarded. The routers would forward the bundle onto these interfaces so that each instantiation of the bundle can reach its intended destinations along a tree-shaped delivery path, which has as its root the source of the bundle and as its leaves each of the destinations. No special multicast group addresses would be used. In fact, the notion of a group is completely absent from this proposal. The source of the bundle is assumed to be magically in possession of all of the destination addresses to which the bundle should be sent. If these destinations are part of a group, that group was formed and information regarding its members was provided to the source out of band. Similarly, no special multicast routing protocol is required to enable DTN routers to determine on which interfaces to forward bundles using this proposal. The routers consider each individual destination address in turn and route to it using a unicast routing protocol.

Consider, for example, the following figure, in which the source application node on the left sends a single bundle with two destination endpoint addresses: the address of D1 and the address of D2. This single bundle would be forwarded to router A, and router A would

forward this bundle out onto two different interfaces, so that a copy of the bundle will reach destination D1 and another copy of it will reach destination D2.

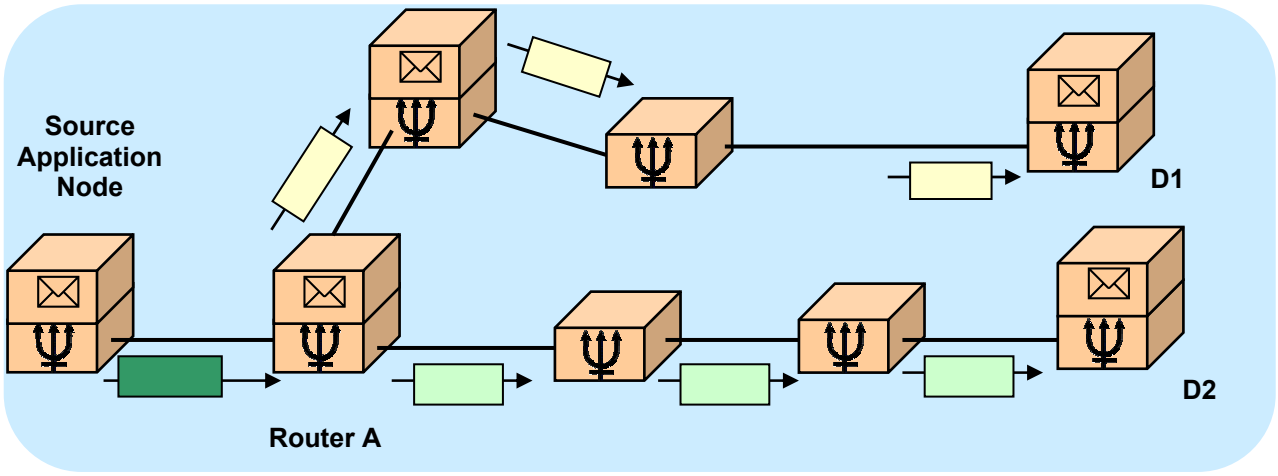


Figure 1-1. A Bundle Delivered to Two Destinations Via Multidestination Delivery

This figure depicts how multidestination delivery would work at a high level. In actuality, several areas of the Bundle Protocol would need to be modified to enable them to support multidestination delivery. A modification to the bundle delivery paradigm to enable the receipt of a single bundle at any given DTN router to cause that router to forward the bundle onto more than one of its interface would have ramifications for the way that bundle class of service options work, such as custodial transfer, retransmission, and release, and it would be complicated by the possibility that a given bundle could be fragmented differently on different downstream paths. Multidestination delivery would also have ramifications for bundle security, which is what we will focus on here.

1.1 Purpose

This paper examines the ramifications that multidestination delivery would have for bundle security; it examines the changes that would be required to the bundle protocol to enable secure multidestination delivery. In particular, it discusses multidestination delivery loops and their avoidance (section 2), examines the impact that providing secure multidestination delivery would have on each area of DTN security (section 3), walks through an example of how each of the available DTN security mechanisms would be applied to a bundle addressed to multiple destinations (section 4), summarizes the security ramifications of multidestination delivery (section 5), and then contrasts the security ramifications of DTN multidestination delivery with the security ramifications of the eventual ideal of real DTN multicast (section 6).

Section 2

Multidestination Delivery Redundancies and Their Avoidance

The figure below shows the undesirable situation in which a bundle that is destined for multiple addresses suffers from a delivery redundancy. The Source Application Node on the left sends a single bundle to two destinations: D1 and D2, on the right.

Router A creates two instantiations of the bundle that it receives, one of which is depicted

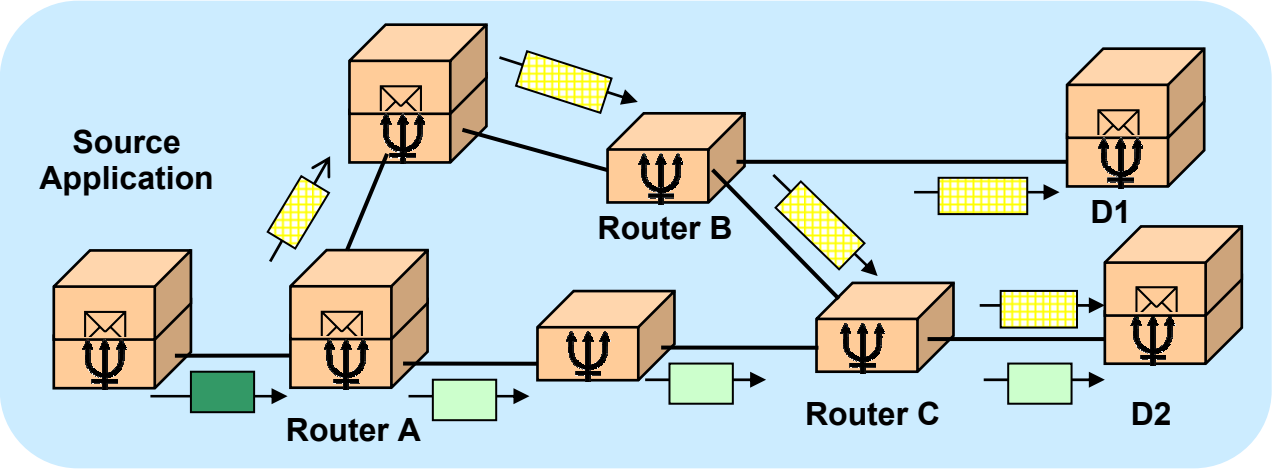


Figure 2-1. A Multidestination Delivery Redundancy

as patterned and one of which is not, and forwards each of these instantiations on a separate interface. The patterned bundle is destined for D1 and the solid bundle is destined for D2.

Router B erroneously creates 2 instantiations of the bundle, forwarding one to D1 and one to D2.

D2 incorrectly receives the bundle twice.

One way to correct this multidestination delivery redundancy would be to enable router C to detect that it has received the bundle twice and to discard the second copy of the bundle it receives. DTN, however, does not include any functionality for arbitrary nodes to detect the receipt of duplicate bundles. Another way to correct this delivery loop would be to enable Router B to "know" that it is only responsible for forwarding to D1, but not to D2. In this case, Router B would only forward the bundle to D1, but not to Router C. In the context of

DTN multidestination delivery, enabling router B to "know" that it is only responsible for forwarding to D1 but not to D2 could be accomplished in one of two ways:

- At each router at which the bundle's path forks, the forwarding router must mark the "relevant" addresses" on each bundle instantiation. This informs all downstream routers as to which destination addresses they are and are not responsible for forwarding the bundle. We refer to this as "marking". Or,
- At each router at which a bundle's path forks, the forwarding router must strip irrelevant addresses from each bundle instantiation. Downstream routers are responsible for forwarding only to those destination addresses that remain in the bundle. We refer to this as "stripping".

In the above example, these two choices would mean either:

- Marking: router A marks destination address D1, but not destination address D2 in the patterned copy of the bundle that it forwards; and it marks destination address D2, but not D1, in the solid copy of the bundle that it forwards. This way, when Router B receives the bundle, it knows that it is only responsible for forwarding the bundle so that it can reach destination D1, but not destination D2, or
- Stripping: router A strips destination address D2 from the patterned copy of the bundle that it forwards, and it strips destination address D1 from the solid copy of the bundle that it forwards. This way, when Router B receives the bundle, it is not even aware that D2 was originally a destination for this bundle, so it will only forward the bundle to reach destination D1.

Given the importance of avoiding multidestination delivery loops and the fact that arbitrary DTN nodes will not be equipped to detect and discard duplicates, our discussion of multidestination delivery must keep in mind that destination addresses will have to be either marked or stripped as a multidestination delivery bundle progresses through the network and its path forks. The choice of whether to mark or strip destination addresses will have ramifications for the degree of end-to-end security with which the bundle can be provided, as will be discussed in sections 3.3 and 3.4. The choice of whether to mark or strip destination addresses will also have ramifications for the amount of bandwidth used by the multidestination bundle, as will be discussed next.

With regard to the relative merits of marking versus stripping, stripping is much more desirable in terms of bandwidth conservation. Destination endpoint addresses can be very long, so stripping, which enables all destination addresses that are no longer relevant to be removed from the bundle, provides a useful mechanism for enabling the size of each bundle to be reduced as its path through the network forks en route to its multiple destinations. With stripping, an address only appears in an instantiation of a bundle if it is needed for delivery of that instantiation of the bundle to the addressed destination. Marking, on the other hand, is

very wasteful of bandwidth insofar as it results in destination endpoint addresses appearing in instantiations of bundles in which the addresses are not needed. Marking does, however, preserve all of the original destination information that was present when the bundle was originally sent, so that a destination that receives a bundle that has been marked as opposed to stripped receives information regarding all destinations to which the bundle was sent. A destination that receives a bundle that has been stripped as opposed to marked, on the other hand, has no way to distinguish this bundle from a bundle that was unicast from the source. It is not clear if preserving all of the bundle destination addresses in all instantiations of the bundle, however, provides any benefit and, if it does, whether the benefit is worth the additional bandwidth that is required to support this feature.

With regard to the bandwidth-savings merits of multidestination delivery, it should be noted that when the addresses of multidestination bundles are marked (as opposed to stripped), more cumulative bandwidth is required to carry the destination endpoint addresses on all instantiations of a given bundle than would be required to carry all the destination endpoint addresses in all bundles if the payload were instead unicast separately to each endpoint destination. When the addresses of multidestination bundles are stripped, the amount of cumulative bandwidth that is required to carry the destination endpoint address portion of all instantiations of a given multidestination bundle is about the same as the amount of cumulative bandwidth that would be used if the payload were instead unicast separately to each endpoint destination. (The cumulative amount of bandwidth used for the destination addresses in the multidestination case versus the unicast case are exactly the same if all destination addresses are specified completely in the multidestination bundle. If some addresses share the same region portion, however, then using the data dictionary to capitalize on this similarity could enable the multidestination case to conserve bandwidth over the unicast case.) So, regarding the bandwidth saved in the bundle header, multidestination delivery, when used with address stripping, may provide some, but not a significant bandwidth savings over unicast delivery. For the most part, the potential bandwidth saving benefits of multidestination delivery over unicast delivery are realized on the payload portion of the bundle. That is, when distributing a payload to n destinations using a multidestination delivery bundle, the amount of bandwidth used by the payload would be at most the size of the payload on each link of the distribution tree, whereas the cumulative amount of bandwidth used by the payload in the unicast case would be n times that amount. Therefore, multidestination delivery is not significantly advantageous over unicast delivery in the case in which the bundle payload is small relative to the endpoint destination addresses, but it can be significantly advantageous in the case in which the bundle payload is large, and its benefit increases as the size of the payload increases.

Section 3

The "Secure" Part of Secure Multidestination Delivery?

DTN security can basically be broken down into three main areas: DTN infrastructure protection, as provided by perimeter access control and the hop-by-hop Bundle Authentication Header (BAH); DTN application protection, as provided by the end-to-end Payload Security Header (PSH) and support for application-layer confidentiality; and security policy routers, which use the PSH to enforce their own access control decisions at junctures within a DTN. Let's examine how each of these areas of DTN security would be affected by defining a multidestination delivery option within DTN.

3.1 DTN Infrastructure Protection for Secure Multidestination Delivery

This section examines how each of the three areas of DTN infrastructure protection (perimeter access control, hop-by-hop data integrity, and hop-by-hop endpoint authentication) would be affected by incorporation of multidestination delivery, and also discusses how multidestination delivery would affect the existing security risks posed by lack of replay detection within DTN.

3.1.1 Perimeter Access Control

Recall that when the bundle agent at a source host receives a Send.request primitive from a bundle application, the first step it may take is to check the permissions of the requesting application and limit or enforce access control based on source application permissions and local policy. Multidestination delivery would not affect how this access control security service operates. If multidestination delivery were to be incorporated into the Bundle Protocol, perimeter access control could continue to operate as it currently does. The fact that the bundle is destined for multiple destinations, however, may be used as an additional criteria according to which the local policy of the source bundle agent determines whether the bundle should be admitted to the DTN and, if admitted, whether the rate at which it is injected or the class of service (COS) options it is allowed to use should be limited in any way. If the local policy of the source host takes destination address into consideration in determining which bundles to admit into the network or how to treat those bundles, then the question arises as to how that host bundle agent should treat a bundle that contains two destination addresses that are expected to be treated differently from each other. The source host's local access control policy would need to account for this potentially ambiguous situation.

3.1.2 Data Integrity and Endpoint Authentication (hop-by-hop)

Recall that the bundle protocol supports mandatory data integrity and endpoint authentication services along every hop in the DTN network, and that these services may be provided on any given DTN link either via the use of the BAH on that link or by the convergence layer of the receiving host asserting the bundle's authenticity along that link. If multidestination delivery were to be incorporated into the Bundle Protocol and the Bundle Protocol were run over an underlying network that includes security features such as IPsec or link encryption that make it sufficiently secure that the convergence layer of each receiving bundle agent could assert the bundle's authenticity at every hop, then the Bundle Protocol could continue to operate as it currently does, without any impact on hop-by-hop security. When using the BAH to provide infrastructure protection, however, some small modifications would be required to support multidestination delivery.

When a bundle is sent from a single source to a single destination, at each intervening hop at which a BAH is to be used to provide security on that hop, the sending bundle agent computes a hash of the bundle, signs the hash with its private key, and forwards the bundle out a single interface to the next receiving bundle agent. The receiving bundle agent receives the bundle, verifies the validity of its signed hash value by comparing it with what it determines the hash value should be, and then computes a new signed hash for the bundle before forwarding it out a single interface to the next hop along the way. The fact that the receiving bundle agent can decrypt the signed hash into the correct value serves to authenticate the sender and recipient of the bundle as well as to verify the integrity of the bundle.

If multidestination delivery were to be incorporated into the Bundle Protocol, a sending bundle agent would compute a hash of the bundle, sign the hash with its private key, and forward the bundle as now occurs in the unicast case. However, as distinguished from the unicast case, it may forward the bundle out multiple interfaces to the next set of receiving bundle agents. Given that the forwarding node must either mark or strip some of the destination addresses on each instantiation of the bundle that is forwarded out multiple interfaces, and that the addresses that are marked or stripped are interface-specific, the forwarding node is required to calculate a unique BAH authentication information value for each interface on which a given bundle is forwarded. With regard to the process of verifying the validity of this authentication information at the receiving bundle agent, however, nothing changes from the unicast case.

3.1.3 Lack of Replay Detection at Arbitrary Routers

We have already mentioned that DTNs do not include a mechanism to detect and reject replayed packets at arbitrary DTN nodes. This means that an attacker could eavesdrop on DTN traffic, record a legitimate bundle during transmission, and then later inject that bundle into the DTN network. This replayed bundle would not necessarily be detected at all. If a

destination application opts to check received bundles for replays, the replay will be detected at that destination. However, the replay will not be detected until the bundle reaches the destination host. Repeated injection of a replayed packet into a DTN, therefore, can cause congestion on the path of that replayed packet through the DTN until the replayed packet expires. The capacity of a replayed packet to cause congestion in the DTN is increased in the case of a packet that is destined for multidestination delivery. Instead of causing congestion on a single linear path from the source to the destination in the network, it causes congestion on all links of a tree-shaped delivery path from the source to multiple destinations. Therefore, the risk posed by not detecting and discarding replayed packets in the DTN is increased with the use of the multidestination delivery option.

3.2 DTN Application Protection and Secure Multidestination Delivery

This section examines how each of the areas of DTN application protection (application data confidentiality, bundle integrity, endpoint authentication, and replay protection) would be affected by incorporation of multidestination delivery.

3.2.1 Application Data Confidentiality

DTN users are free to use the DTN Bundle Protocol to support end-to-end confidentiality for application data, but the actual encryption of the user information to be transmitted as the application data unit must be accomplished by the application before being passed to the bundle layer; similarly, decryption of the application data unit received at the destination host must be accomplished by the destination application rather than the destination bundle layer. There is no mechanism within the Bundle Protocol itself for the bundle layer to perform encryption or decryption of the bundle payload or any portion of the bundle header. Although confidentiality is not provided by the Bundle Protocol itself, however, encryption at the application layer is supported by the Bundle Protocol's ability to signal from the source to the destination which encryption algorithm and encryption key was used to encrypt the application data. This enables flexibility in the use of various algorithms and keys and in performing key rollover.

If multidestination delivery were to be incorporated into the Bundle Protocol, the Bundle Protocol would require modification in order to enable it to support encryption of application data that is intended for receipt (and decryption) at multiple destinations. Designing a DTN solution based on the way that secure mail, for example, would typically provide confidentiality for a message sent from a single source to multiple destinations, a DTN application wanting to provide confidentiality for data that it is sending to multiple destinations would use a single symmetric key, known as a content encryption key (CEK) to encrypt the application data that is eventually transmitted as the bundle payload. It could not use public key cryptography because in order to use public key cryptography, it would have to encrypt the data with the public key of the destination. Using public key cryptography on data intended for multiple destinations, therefore, would require the application to encrypt

the data separately for each destination, using the private key of each destination, and include all of the resulting cyphertext in the bundle, which would so diminish the benefits of using multidestination delivery that it wouldn't be worthwhile. Instead, the DTN application would use a symmetric key (a CEK) to encrypt the application data. It would then encrypt this CEK once for each of the respective destinations, using the public key for each of those destinations, and include each of these encrypted CEKs in the bundle along with the encrypted payload. At a given destination application, upon receipt of the application data, the destination application would use its private key to decrypt the encrypted CEK that pertains to the destination, and then use the resulting plaintext CEK to decrypt the received application data.

Although encryption and decryption of application data for multidestination delivery would occur in the DTN application rather than at the bundle layer, additional signaling between the DTN application and the bundle layer would be required to enable this encryption to be supported, and additional fields would be required in the Bundle Protocol to carry the encrypted CEKs that pertain to each destination that is included in the bundle. It has already been recommended that additional optional parameters of the Send.Request and the Data.Indication primitives of the bundling service should be defined to enable these primitives to adequately support confidentiality for unicast delivery. In particular, additional optional parameters are needed to enable the following information to be conveyed:

- Which encryption algorithm (if any) has been used to provide confidentiality
- The key ID of the key that was used with this encryption algorithm

If multidestination delivery were to be incorporated into the Bundle Protocol, the above parameters would be sufficient, providing that all destinations to which the bundle is to be sent have the same a priori understanding of what symmetric CEK corresponds with the key ID used. However, making this assumption only avoids the question of how that key initially got distributed to each of the destinations. A more robust and complete solution would support the mechanism of encrypting the CEK once per destination, using the public key of each destination, and including these CEKs in the bundle. To support this encrypted CEK mechanism for multidestination delivery, optional primitives would be required to enable the DTN application to convey the following information to its underlying bundle agent:

- Which symmetric encryption algorithm (if any) has been used to provide confidentiality for the application payload
- Which asymmetric encryption algorithm has been used to provide confidentiality for the CEK (if the payload was encrypted)
- For each destination endpoint ID supplied, the corresponding result of encrypting the CEK with the public key associated with that endpoint ID.

Note that this solution is a bit restrictive because it assumes that a given endpoint ID will unambiguously indicate a corresponding public key, meaning that keys will strictly be associated with endpoint IDs rather than be associated with specific applications or roles, several of which could reside at a given endpoint. A more flexible and complete solution that enables a finer granularity of key identification at the cost of using additional bits of the header would be to associate not only an encrypted CEK with each destination, but also a key ID that indicates the private key to be used with the asymmetric algorithm that was used to encrypt the CEK, with each destination. This solution provides flexibility for supporting efficient key rollover and the ability to associate keys with whatever entities are most appropriate for the application. To recap, the additional, optional parameters that would be needed to support the complete and flexible confidential multidestination delivery solution would have to convey the following information:

- Which symmetric encryption algorithm (if any) has been used to provide confidentiality for the application payload
- Which asymmetric encryption algorithm has been used to provide confidentiality for the symmetric CEKs (if the payload was encrypted)
- For each destination endpoint ID supplied,
  - the keyID corresponding to the public/private key pair used to encrypt the CEK
  - the result of encrypting the CEK with the public key associated with that keyID.

Optional bundle header fields will also need to be defined for carrying the above new values:

- An optional “symmetric encryption algorithm ID” field
- An optional “asymmetric encryption algorithm ID” field
- A keyID field associated with each destination endpoint ID
- An encrypted CEK field associated with each destination endpoint ID

3.2.2 Data Integrity and Endpoint Authentication (Source-to-Destination)

The Payload Security Header (PSH) of the Bundle Protocol can optionally be used to provide end-to-end data integrity and endpoint authentication for the entire bundle, meaning that it provides a mechanism whereby the destination bundle agent can verify that the bundle received has not been modified in transit since being sent by the originating source; it can also enable the destination bundle agent to verify that the bundle was intended for the named destination (because the signed hash value was calculated over the contents of the destination field in the Primary Bundle Header). Lastly, it can enable the destination bundle agent to verify that the bundle originated from the endpoint ID listed in the source field (because the signed hash value was calculated over the contents of the source field in the Primary Bundle

Header). Also, either the source field or some other bundle header field was used to look up the appropriate public key to use to decrypt the signed hash. Because the signed hash decrypted correctly, this means that the corresponding private key must have been used to sign the hash, and given that the private key is known only to the source, this authenticates either the endpoint ID listed in the source field or the entity associated with the key used to decrypt the hash as having in fact signed the hash and having been the location from which the bundle originated.

These services are provided by having the source bundle agent calculate the hash over the entire bundle, use the private key of the source to sign the hash, and place the signed hash value in the Security Information Field of the Payload Security Header. Upon receipt of the bundle, the destination bundle agent applies the source's public key to the signed hash, thereby decrypting it into the original unsigned hash value. The destination then calculates its own hash of the bundle and compares the hash value that it has calculated with the (now unsigned) hash value received. If the two values are equivalent, then the destination bundle layer can be assured that the contents of the bundle have not been modified since being sent from the source. Furthermore, after the destination bundle agent compares the Payload Security Header hash value with the computed value and finds them to be equivalent, it may optionally make sure that the bundle is not a replay by comparing its (source, timestamp) pair value with the (source, timestamp) pair values of bundles that it has already received. This optional replay detection capability is discussed further in a separate section below. The signed hash value in the Security Payload Header provides data integrity protection for the entire bundle (except for the BAH and mutable fields), including source endpoint ID and timestamp, which determine the bundle's uniqueness; destination endpoint ID; class of service; and payload. Because all of these fields are included in the PSH hash calculation and the hash is signed with the sending application's private key, the PSH truly provides an end-to-end data integrity and endpoint identification service from source application to destination application. The PSH enables a destination application to reliably determine whether or not a received bundle has been modified while in transit, even in the case in which one or more of the DTN routers along the path from source to destination have been compromised.

If multidestination delivery were to be incorporated into the Bundle Protocol, the signed hash in the PSH that is received at a given destination *d* has to be calculated over the entire bundle, minus mutable fields such as the custodian field and the BAH, and including the source endpoint ID, that particular destination *d*'s endpoint ID, and the keyID field and encrypted CEK associated with destination *d* (if present), but not necessarily including other destination endpoint IDs or the keyID or encrypted CEK fields associated with those other destinations. Including the other destinations and their associated encryption information does not do any harm with respect to the security protection provided, but it is not essential in order to provide assurance of end-to-end bundle integrity and endpoint authentication from the source to that particular destination. At a particular destination *d,* however, the integrity

of *d*'s endpoint ID, keyID field, and encrypted CEK (if present) absolutely needs to be protected by the PSH hash received at *d.* This leaves designers with two choices regarding how to calculate the PSH hash at the source:

- Compute a single hash over the entire bundle, less the usual mutable fields such as the BAH and the custodian field. This means the hash would be computed over all destination addresses. Sign the hash and put it in the PSH. In this case, the destination must receive the entire bundle including all destination addresses in order to be able to verify the correctness of the hash. No destination addresses would be allowed to be stripped off by intermediate routers at points at which a bundle's path forks and thereby eliminates some addresses from being potential destinations for that particular instantiation of the bundle. This solution requires destination addresses to be marked rather than stripped.

- If a bundle is destined for *n* endpoint addresses, compute n different hashes for the bundle. The hash for destination *d* would omit all destination addresses, key IDs, and encrypted CEKs (if they are present) except for *d*'s destination address, keyID, and encrypted CEK. This hash would then be signed by the source and put into the d^{th} PSH field in the bundle. In this case, the bundle would have to have not only *n* destination addresses, *n* encrypted CEK's (if present) and *n* keyIDs (if present), but it would also have *n* PSHs. Destination *d*, however, need not receive the entire bundle. It need only receive the bundle with *d*'s endpoint address, *d*'s encrypted CEK (if present), *d*'s keyID field (if present) and *d*'s PSH. This information is sufficient for *d* to be able to compute the hash of the bundle received and compare it with the decrypted value of the signed hash received in PSH *d*. Using this method of providing end-to-end bundle integrity and endpoint authentication, destination addresses could be stripped off by intermediate routers at points at which a bundle's path forks and thereby eliminates some addresses from being potential destination for that particular instantiation of the bundle. In fact, not only could the destination addresses be stripped off, but their associated encrypted CEKs (if present), keyIDs (if present), and PSHs could also be stripped off.

It must be noted, however, that although such stripping does not interfere with the ability of the destination bundle agent to authenticate the received bundle, it does interfere with the ability of a security policy router to authenticate a received bundle and authenticate all of the destination addresses to which the bundle would eventually be forwarded from the security policy router. This complication that destination address stripping causes to security policy routers is discussed further in section 3.3, and a solution that addresses it is proposed in section 3.4.

3.3 Security Policy Routers

As discussed in section 3.2.2, the PSH is calculated at the source host, using the source's private key, and it is checked at the destination host to provide end-to-end security. It may, however, also be checked at one or more DTN security policy routers while in transit. Checking the signed hash in the Payload Security Field is accomplished by having the security policy router's bundle agent calculate its own hash of the received bundle and compare this received hash with the decrypted value of the hash that was received with the bundle. If the hash values are equivalent, the security policy router can be assured that the bundle has not been modified in any way since being sent from the source, which is one (of possibly many) necessary criterion for determining whether the bundle should be forwarded.

In order to be able to verify the correctness of the hash value, the bundle agent must have access to the bundle as it was when the original hash was calculated on it. This has ramifications for which of the two choices regarding how to calculate the PSH hash at the source should be selected, as discussed in section 3.2.2 Data Integrity and Endpoint Authentication (Source-to-Destination) above. It also has ramifications regarding whether destination addresses should be marked or stripped from the multidestination bundle, as discussed in section 2. If the method of PSH calculation that involves computing *n* different hashes for the bundle, one per destination, is used, then a security policy router could choose a destination and compute that destination's hash on the bundle and thereby assure itself of the identity of the source of the bundle and of the integrity of the single destination address and associated keyID and CEK information for that destination that was used to calculate the hash. The security policy router, however, would not be able to compute a single hash on the bundle received and use this hash value to be assured of the integrity of all of the bundle destination endpoints addresses and their associated keyIDs and CEKs. If the method of PSH calculation that involves computing a single hash on the entire bundle, including all destination endpoint addresses and associated keyID and CEK information is used, then a security policy router would be able to compute a single hash on the bundle received and use this hash value to be assured of the integrity of the source of the bundle as well as of all of the destination endpoint addresses and their associated keyID and CEK information. From the standpoint of enabling a security policy router to provide assurance of bundle integrity for the entire bundle rather than for only a particular branch of the bundle's multidestination path, computing the PSH over all destination addresses is preferable. Computing a single PSH for the bundle that is calculated over all destination endpoint addresses and leaving all destination addresses in the bundle rather than deleting them as the bundle progresses on its tree-like path enables a security policy router to provide more security than does the method of computing one PSH per destination address.

3.4 Optional Replay Detection at Destination Hosts and Security Policy Routers

Although there is no requirement for duplicate bundles to be detected at arbitrary nodes within the DTN, there is a requirement for destination bundle agents to be able to optionally detect and discard duplicate bundles received. The same optional ability is also required at DTN security policy routers that may want to enforce their own access control policy before forwarding a bundle on a certain link. The optional ability to detect and discard duplicates could be crucial to the ability of the security policy node to protect the link from having its resources wasted by transmitting replayed bundles. Because there is no way within the Bundle Protocol for an arbitrary DTN node to distinguish an illegitimately replayed bundle from a legitimate retransmission, it is expected that a security policy router will not be configured to detect and discard replayed bundles unless the security policy router is itself serving as a bundle custodian. As a bundle custodian, a security policy router would be in a position to consider all duplicate bundles received as illegitimate replays. If a security policy router were to be located between a custodian and some destination addresses, on the other hand, then the security policy router would not have any way to distinguish legitimate retransmissions originating from that custodian from illegitimate replays. ***Is this correct???*** If not, how can a security policy router distinguish a replay from a legitimate retransmission??

If multidestination delivery were to be incorporated into the Bundle Protocol, then, given the same assumption that security policy routers configured to detect and discard replays are also custodians, the actions performed by the security policy router and destination bundle agents to detect and discard duplicates would not be affected in any way. The security policy router and destination bundle agents would still use the (source, timestamp) pair value to uniquely identify bundles received and thereby detect and discard replays.

3.5 A Modified Method of PSH Hash Calculation and Bundle Stripping

While calculating the PSH hash over all bundle destination addresses is required to enable security policy routers to provide optimal security, keeping these destination addresses in the bundle (marking), as opposed to stripping them out, is undesirable in terms of bandwidth usage. Ideally, we would like to have the PSH be calculated only once, and such that it can be used to authenticate all destination addresses, but we would also like to save bandwidth by stripping destination addresses (which may be very large) from the bundle at those junctures in the delivery path at which they are no longer needed. The following innovation has been suggested as a means of calculating the PSH to achieve both of these goals:

- Create a hash value of each of the destination addresses in the bundle.

- Add these hash values as new fields of the bundle in such a way that associates each destination address with its corresponding hash value.
- Calculate the PSH hash over the entire bundle except for
  - The BAH
  - All mutable fields, such as the custodian and sender fields
  - All destination endpoint addresses

When a bundle arrives at a router and the router has to forward the bundle on multiple interfaces, the router should strip off all destination addresses (but not any destination hash values) that will not be needed on that interface or beyond. This way, when a bundle arrives at a security policy router or at a destination endpoint, the bundle will still contain all information over which the PSH hash was calculated, so the PSH can be adequately verified. Furthermore, this information includes hashed values for each destination, thus enabling the authenticity of all destination addresses that remain in the bundle to be verified, and ensuring that the source address, time stamp, destination addresses, payload, and other relevant portions of the bundle have not been modified since the bundle was sent.

When a security policy router receives a multidestination bundle, it should validate the content of that bundle by performing the following steps:

- Decrypt the hash value received in the bundle's PSH.
- For each destination address in the bundle, calculate the hash value of that destination address and verify that this calculated value is equivalent to the destination address's corresponding hashed value as found in the bundle.
- Calculate the hash over the entire received bundle, except for
  - The BAH
  - All mutable fields, such as the custodian and sender fields
  - All destination endpoint addresses
- Verify that this calculated hash is equivalent to the PSH hash value received in the bundle, as decrypted above.

This option combines the best of both the "marking" and the "stripping" options for preventing delivery loops, and appears to be optimal. It enables a security policy router to authenticate the integrity of all destination addresses in the received bundle with a single hash calculation, rather than authenticating the integrity of only a single destination address or having to calculate a separate hash for each destination. Yet it also enables destination addresses that are not relevant to a particular instantiation of a multidestination bundle on a

given path of the delivery tree to be stripped from that instantiation of the bundle, thus preventing bandwidth from being wasted by bundles carrying destination addresses that are no longer relevant to them.

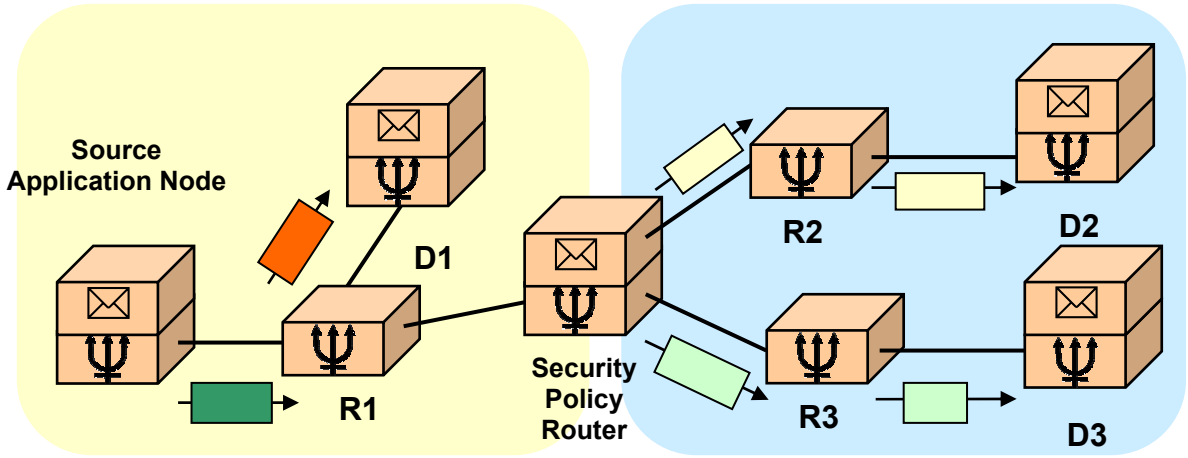
To implement this mechanism, a new, optional bundle field for indicating what hash algorithm to use to compute the hash of each destination endpoint address would need to be defined, as would the new fields for holding the hash values associated with each destination address.

Section 4

Secure Multidestination Delivery: walking through an example

In this section we walk through an example of the sending of a multi-destination bundle from a single source to multiple destinations and focus on the security-related processing that would need to occur. To fully explore the security ramifications of enabling DTN multidestination delivery, in our example the application data unit that is sent in the multidestination bundle will be encrypted, and the bundle itself will be protected with end-to-end security as provided by the PSH and with hop-by-hop security as provided by the BAH. We will also follow the steps that the bundle takes as it is processed at a security policy router within the DTN, as it is received at its destination host bundle agent and checked for replays, and as the application data unit is received by the destination application for decryption.

The topology of the network over which the bundle will travel is shown in the following figure. The source sends a bundle to destination hosts D1, D2, and D3 and the path on which the bundle must travel to reach destinations D2 and D3 includes a security policy router.



Here are the following steps that need to occur at various locations within the DTN network:

4.1 Source Application Processing

In order to send encrypted application data to multiple destinations using a single bundle and multiple destination fields, a source application would perform the following steps:

1. The source application encrypts the data to be sent using a symmetric encryption algorithm and symmetric key of its choosing. This encryption key is known as the content encryption key (CEK).
2. The source application (or key signing library or tool) encrypts the CEK once for each of the destinations to which the data is to be sent, using the public key associated with each of those destinations.
3. The source application invokes the Send.Request Primitive, which includes the following arguments that are specific to secure multidestination delivery:

   – multiple destination communication endpoint IDs;

   For confidentiality:

   – encrypted application data unit;

   – an indication of which symmetric encryption algorithm has been used to provide confidentiality for the application data unit

   – an indication of which asymmetric encryption algorithm has been used to provide confidentiality for the symmetric CEKs (if the payload was encrypted), and

   – associated with each destination endpoint ID supplied,

   - the keyID corresponding to the public/private key pair used to encrypt the CEK
   - the result of encrypting the CEK with the public key associated with that keyID.

   For end-to-end integrity and authentication:

   – an indication of which hash algorithm should be used to compute the hash in the PSH security information field

   – an indication of which encryption algorithm should be used to encrypt the PSH hash

   – an indication of which key should be used to encrypt the PSH hash

   – an indication of which hash algorithm should be used to compute the hash of each destination endpoint address

4.2 Source/Sending Bundle Agent Processing

Upon receipt of a Send.Request primitive from a DTN application requesting initiation of the authenticated transfer of an encrypted application data unit to multiple destinations, the source host bundle agent would first assemble the bundle, then add the Payload Security Header, and then add a Bundle Authentication Header. Specifically, it would perform the following steps to assemble the bundle and add the PSH:

1. Perform permission checking and optionally enforce a local access control policy that may restrict the injection of this data into the network, as discussed in Step 1 of section 4.1 of the Bundle Protocol Specification.
2. Assemble the bundle except for the BAH. (If the bundle is to have a BAH, do not put it in the bundle yet.) The bundle should include the following fields that are specific to providing secure multicast delivery:
   - An optional “symmetric encryption algorithm ID” field
   - An optional “asymmetric encryption algorithm ID” field
   - An optional “PSH hash encryption algorithm ID” field
   - An optional “PSH hash algorithm ID” field
   - An optional “PSH hash encryption key ID” field
   - An optional “destination address hash algorithm ID” field
   - An optional “key ID” field associated with each destination endpoint ID
   - An “encrypted CEK” field associated with each destination endpoint ID
   - Multiple destination addresses
   - Hash values of each destination address
3. Populate all fields with appropriate values, except assign the Custodial field a zeroed-out value, the security information field of the Payload Security Header a zeroed-out value, and the sender endpoint ID field a zeroed-out value. (In short, all mutable fields must be zeroed-out, and the BAH must not be present.)
4. If there is more than one destination endpoint address, and if the bundle includes an optional “destination address hash algorithm ID” field, use the value of this field to look up the appropriate hash algorithm to use. Compute the hash value of each destination endpoint address and place these hash values into the bundle in such a way that each hash value is associated with its corresponding destination address.
5. Use the PSH hash algorithm ID field parameter in the Send.Request primitive to determine what hash algorithm to use.

6. Calculate the hash over the entire bundle as assembled above but omitting all destination endpoint addresses (but including the hash values of each of these destination endpoint addresses) and pass this value to an entity (the application, a key-signing service, etc.) that is privy to the private key that will be used to sign this hash for this bundle. Also, if the bundle includes an optional PSH key ID field, the value of this field must also be passed to the key-signing entity.
7. Receive the signed hash back from the key-signing entity and insert it into the security information field of the PSH.
8. Populate the Custodial field with the appropriate endpoint ID value and the sender endpoint ID field with the correct value (if desired).

At this point the bundle is ready for forwarding. Because the bundle has multiple destination addresses and therefore may need to be forwarded on multiple interfaces, the bundle agent performs the following steps to ready each instantiation of the bundle that will be forwarded:

9. For each destination address included in the bundle, the bundle agent consults its routing service to determine on what interface the bundle should be forwarded in order to reach that destination address.
10. For each interface on which a bundle needs to be forwarded, the bundle agent performs the following on the instantiation of the bundle that will be forwarded on that interface:
    - Deletes all destination addresses in the bundle that are not intended to be reached from this interface.
    - Adds a BAH to the bundle.
    - Zeros-out the value of the authentication information field of the BAH.
    - Populates all other fields of the BAH with appropriate values. For example, if the optional hash algorithm ID field is to be used in the BAH, insert the appropriate algorithm ID value into this field. Similarly, if the optional key ID field is to be used in the BAH, insert the appropriate key ID value into this field. Insert the correct value into the BAH length field.
    - Calculate the hash over the entire bundle except for the payload field of the Bundle Payload Header.
    - Sign the hash.
    - Insert this signed hash into the authentication field of the Bundle Authentication Header.

At this point the sending bundle agent has prepared a unique instantiation of the bundle for each interface on which the bundle needs to be forwarded to reach its multiple destination addresses. The PSHs on all instantiations of the bundle are identical, but the BAHs on each instantiation of the bundle are unique to the interface, because their signed hash values reflect which destination addresses are and are not relevant on each interface.

4.3 Receiving Bundle Agent Processing

Upon receipt of an authenticated, encrypted, multidestination bundle from a sending bundle agent, a receiving bundle agent must authenticate the bundle, determine on which interfaces the bundle should be forwarded, and (if necessary) prepare interface-specific BAHs to replace the BAH on the received bundle before forwarding the bundle on each of those interfaces. Specifically, the receiving bundle agent performs the following steps to authenticate the received bundle:

1. If the bundle authenticity parameter of the Bundle.Indication Primitive that was used to deliver this bundle to the bundle agent asserted the authenticity of the bundle, the bundle is authenticated and no further steps are required.
2. If the bundle authenticity parameter of the Bundle.Indication Primitive that was used to deliver this bundle to the bundle agent did not assert the authenticity of the bundle, but the bundle does not include a BAH, the bundle must be discarded and processed no further; in this case, a bundle status report indicating the authentication failure may be generated, destined for the receiving bundle agent's own administration endpoint.
3. If the received bundle includes a key ID field in the BAH, use it to look up the appropriate key to use; else if the received bundle includes a non-zero-valued sender endpoint ID field, use it to look up the sending bundle agent's public key; else use the sending bundle agent's endpoint ID, which was passed up as a parameter of the Bundle.Indication Primitive, to look up the sending bundle agent's public key.
4. Use the key derived in step 3 to decrypt the signed hash value received in the authentication information field of the Bundle Authentication Header.
5. Determine which hash algorithm to use.
6. Zero out the authentication field of the BAH.
7. Calculate the hash over the entire bundle except for the payload field of the Bundle Payload Header.
8. Verify that the decrypted signed hash value received is equivalent to the hash value computed. If the values are not equal, the bundle has failed to authenticate and the bundle must be discarded and processed no further; in this case, a bundle status report indicating the authentication failure may be generated, destined for the receiving

bundle agent's own administration endpoint. If the values are equal, the bundle (excluding the payload) has been authenticated. The sender of the bundle has been verified to have been the endpoint address claimed, and the bundle (excluding the payload) has been verified not to have been modified since being sent from the previous hop sender.

Next, the receiving bundle agent must consider each destination address remaining in the bundle. These are the destination addresses to which the bundle must be forwarded by the bundle agent.

For each of the destination addresses included in the bundle, the bundle agent consults its routing service to determine on what interface the bundle should be forwarded in order to reach that destination address. The bundle agent will then know which interface(s) on which it must forward the bundle, as well as which destination addresses are intended to be reached from each of these interfaces. The bundle agent must then prepare interface-specific BAHs for the bundle to be forwarded on each interface. The steps that must be taken are very similar to those that were already described in the Source/Sending Bundle Agent Processing section, with variations to account for the fact that this bundle was received from a previous-hop bundle agent rather than having just been assembled. Specifically, for each interface on which a bundle needs to be forwarded, the bundle agent performs the following on the instantiation of the bundle that will be forwarded on that interface:

1. Deletes all destination addresses in the bundle that are not intended to be reached from this interface.
2. If the bundle will not be protected with a BAH on the next hop from this interface, then no BAH hash has to be calculated for this instantiation of the bundle because it will not have a BAH. Therefore, if the bundle that was received includes a BAH, remove it. If the received bundle did not include a BAH, BAH-related processing is finished for this interface.
3. If the bundle will be protected with a BAH on the next hop from this interface, then a BAH hash will have to be calculated (per the steps below) and the instantiation of the bundle to be sent over this interface will need to have a BAH. If the bundle that was received included a BAH, proceed to the next step. If the received bundle did not include a BAH, then add one to the bundle.
4. If the received bundle is being transformed into a bundle fragment, this fragmentation must be accomplished (by insertion of a fragment header with offset zero as well as by changing the payload length field) before the rest of the steps for creating the new BAH listed below.
5. Zero-out the value of the authentication information field of the BAH.

6. Populate all other fields of the BAH with appropriate values. For example, if the optional hash algorithm ID field is to be used in the BAH, insert the appropriate algorithm ID value into this field. Similarly, if the optional key ID field is to be used in the BAH, insert the appropriate key ID value into this field. Insert the correct value into the BAH length field.
7. Insert the bundle agent's endpoint ID into the sender endpoint ID field, if this field is to have a non-zero value on this link.
8. Calculate the hash over the entire bundle except for the payload field of the Bundle Payload Header.
9. Insert this signed hash into the authentication field of the Bundle Authentication Header.

At this point, the receiving bundle agent has authenticated the bundle it received and prepared separate instantiations of it for forwarding on all appropriate interfaces, thereby preparing itself to become a sending bundle agent. The PSHs on all instantiations of the bundle are identical, but each instantiation of the bundle may or may not have a BAH and, if present, the BAH on a given instantiation of a bundle is unique to the interface on which that instantiation of the bundle is to be forwarded, because the signed hash value in the BAH of each bundle instantiation reflects which destination addresses the bundle is intended to reach from that particular interface.

4.4 Security Policy Router Processing

Upon receipt of a multidestination bundle from a sending bundle agent, a receiving security policy router must have its bundle agent verify the bundle's BAH hash value to authenticate the bundle, as described in the previous section; reassemble the bundle, if necessary; verify the value of the security information field of the PSH and optionally detect and discard replays; and enforce the router's local access control policy for this bundle that may restrict the rate at which this bundle is allowed to be forwarded, or even whether it is allowed to be forwarded on particular interfaces. If the bundle is to be forwarded on one or more interfaces, the security policy router must, as described in the previous section, determine on which interfaces the bundle should be forwarded, and (if necessary) prepare interface-specific BAHs before forwarding the bundle on each of those interfaces.

In order to perform source endpoint authentication and ensure bundle integrity using the PSH, the security policy node must be in possession of all parts of the original bundle (header and all payload) over which the PSH hash was calculated. If the bundle was fragmented, the fragments must now be reassembled into the original bundle and all fragment headers discarded. If not all fragments are received at this node, making reassembly impossible, then source authentication cannot be performed. After the entire bundle has been

received (and reassembled, if necessary), the bundle agent at the security policy node would perform the following steps to authenticate the signed hash information in the PSH:

1. If the PSH of the received bundle includes an optional "key ID" field, use it to look up the appropriate key to use.
2. If the PSH of the received bundle does not include an optional "key ID" field, use the value in the source field to look up the appropriate public key to use.
3. Apply the key to the signed hash in the security information field of the Payload Security Header, thereby decrypting it into the original unsigned hash value.
4. If there is a destination address hash algorithm ID field in the bundle, use it to determine what hash algorithm to use for the next step.
5. For each destination address remaining in the bundle, calculate the hash value of that destination address and verify that this calculated value is equivalent to the destination address's corresponding hashed value as found in the bundle. If any of these hash values is not correct, the bundle must be discarded and a bundle status report indicating the failure may be generated, destined for the receiving bundle agent's own administration endpoint.
6. If there is a hash algorithm ID field in the PSH, use it to determine what hash algorithm to use for the next step.
7. Assign the Custodial field, the security information field of the Payload Security Header, and the sender endpoint ID field zeroed-out values. (All mutable fields should be zeroed-out.)
8. If there is a BAH in the bundle, remove it.
9. Calculate the hash value of the entire bundle except for the destination addresses (and the BAH, which has already been removed per the previous step).
10. Compare the calculated hash value with the (now unsigned) hash value received in the PSH.
11. If the values are not equal, the bundle must be discarded and a bundle status report indicating the failure may be generated, destined for the receiving bundle agent's own administration endpoint.
12. If the values are equal, the bundle agent may optionally compare the (source, timestamp) pair values of the bundle with the local list of such values of already-received bundles. If the pair value is a duplicate, the bundle must be discarded and a bundle status report indicating the failure may be generated, destined for the receiving bundle agent's own administration endpoint. If the pair value is unique, the pair must be added to the local list. At this point, the bundle has been authenticated:

the identity of its sender has been verified, and it has been determined not to have been modified since being sent from the source.

13. The bundle agent would determine on which of possibly multiple interfaces the bundle would need to be forwarded to reach its destinations.

14. The security policy router now imposes its own access control policy on the bundle using the following steps: based on the source of the bundle in combination with local access control policy unique to each interface, the bundle agent may enforce policy that affects the rate at which this bundle is forwarded, the COS options that the bundle is allowed to use, or even whether or not the bundle is forwarded on each interface. If the bundle is not forwarded as a result of the enforcement of the access control policy, a bundle status report indicating the failure may be generated, destined for the receiving bundle agent's own administration endpoint ID.

4.5 Destination Bundle Agent Processing

Upon receipt of a bundle at a destination bundle agent, the bundle agent would authenticate the bundle using the bundle's BAH, if one is present, as described previously in the "Receiving Bundle Agent Processing" section; authenticate the signed hash information in the PSH, if one is present, and check for replays if requested by one or more of the registered destinations, as described above in the "Security Policy Router Processing" section. However, the destination bundle agent would not, as described in that section, enforce any sort of local access control policy. The destination bundle agent would invoke the Data.Indication primitive at all DTN applications that had registered to receive this bundle, thereby delivering the bundle's application data unit to the destination application(s). The following security-related parameters would have to be available for inclusion in the Data.Indication primitive to support decryption of the application data unit by the receiving application:

- encrypted application data unit;
- an indication of which symmetric encryption algorithm has been used to provide confidentiality for the application data unit
- an indication of which asymmetric encryption algorithm has been used to provide confidentiality for the symmetric CEKs (if the payload was encrypted), and
- the keyID corresponding to the public/private key pair used to encrypt the CEK for this particular destination
- the encrypted CEK associated with this particular destination.

4.6 Destination Application Processing

Upon receipt of a Data.Indication primitive from its local bundle agent indicating the receipt of an encrypted application data unit, the destination application would have to perform the following steps:

1. Use the encryption keyID (if present) and the asymmetric encryption algorithm ID (if present) to decrypt the encrypted CEK received (if present).
2. Use the decrypted CEK and the symmetric encryption algorithm ID (if present) to decrypt the application data unit received.

Section 5

Summary of Security Ramifications of Multidestination Delivery in DTN

Security Service	How a multidestination delivery option would affect the security service
Perimeter Access Control	If the access control policy at any source bundle agent takes destination addresses into consideration, the access control policy would need to be amended to account for the possibility that a single bundle may contain two destination addresses that are intended to be treated differently from each other.
Hop-by-hop Data Integrity and Endpoint Authentication	Routers at which a bundle's path forks will need to calculate separate BAH hash values for each interface on which the bundle will be forwarded.
Lack of Replay Detection at Arbitrary Routers	The risk posed by replays is multiplied with the use of multidestination delivery, because a single replayed multidestination packet can affect many more DTN links than can a single replayed unicast packet.
Application Data Confidentiality	If distribution of the content encryption key (CEK) to all destinations is performed out of band, no modifications would be required to the Bundle Protocol to support application data confidentiality. Modifications would be required to distribute the CEK and the following optional related information as part of the bundle: • ID of the symmetric encryption algorithm used for confidentiality (the algorithm with which the CEK is used) • ID of the asymmetric encryption algorithm used to encrypt the CEK • For each destination endpoint: – ID of the public/private key pair used to encrypt the CEK – The result of encrypting the CEK with the public key
End-to-End Data Integrity and Endpoint	The following new fields and new mechanisms must be defined to enable a single PSH hash to provide end-to-end security for the bundle, yet enable destination addresses to be stripped from the bundle as they are no

Authentication	longer needed on a certain path of the delivery tree: • Each destination address in the bundle must have an associated field that contains a hash of that destination address. • The PSH hash should be calculated over the entire bundle except for – The BAH – All mutable fields, such as the custodian field and sender field – All destination endpoint addresses • To validate the hash, a receiving node would first ensure that each destination address present in the bundle has a corresponding hash value which is correct, and then compare the received (and decrypted) PSH hash value with the PSH hash value as calculated above. If the receiving node is a destination, it must also ensure that its own address is present in the bundle.
Optional Replay Detection at Destination Hosts and Security Policy Routers	No change is required. The (source, time stamp) pair is still used to uniquely identify bundles. If two bundles with the same (source, time stamp) pair are received at a destination host or security policy router at which optional replay detection has been enabled, then the second such bundle received should be discarded as a duplicate.
Security Policy Routers	Security policy routers would validate the PSH hash as described in the end-to-end integrity and endpoint authentication item above. Assuming that the security policy router is not a destination, however, it would not need to ensure that its own address is listed as a destination in the bundle.
Avoiding delivery loops	Unnecessary addresses will be stripped off at each branch in the bundle's delivery tree.

Section 6

Comparing the security ramifications of DTN multidestination delivery with those of "real" DTN multicast

This paper has examined the security ramifications of enabling a single bundle to be delivered to multiple destination endpoint addresses. In actuality, the DTN Bundle Protocol does not currently have any such provisions for enabling bundles to be delivered to multiple destinations. Ideally, it would be desirable to define a "real" DTN multicast capability that would be analogous to the Internet's multicast capability. Such a capability would involve the use of special multicast addresses such that a single such multicast address could be used to denote a group of destination endpoint addresses instead of requiring the source of each bundle to individually specify each destination address in each bundle. Accompanying group management protocols would need to be defined whereby destination endpoints could join and leave groups, and associated security services would need to be defined to distribute cryptographic keying information to group members, remove members from a multicast group, and re-key the remaining group members, when necessary. In addition, a multicast routing protocol would be required to enable DTN routers to determine on what interfaces a received multicast DTN bundle should be forwarded.

Given the current absence of such a "real" multicast capability, this paper has explored the possibility of using a modified version of multicast in which a source is presumed to know each destination address to which it needs to transmit a bundle. While this capability is not as robust as a "real" multicast capability would be, it could still provide important bandwidth-saving benefits. In this section, we will examine how the security ramifications of the DTN multidestination delivery mechanism that we have already explored compare with such a "real" multicast capability.

6.1 Avoiding Delivery Loops

If "real" multicast is used, then to avoid multicast routing and delivery loops, either the multicast routing protocols used must be inherently loop free or each multicast router must have the capability to detect and discard duplicates. Without some such provision, a multicast routing loop could have detrimental affects on a large portion of the DTN network. The mechanisms of marking or stripping addresses that we have defined for use with multidestination delivery are not applicable to "real" DTN multicast as a means of avoiding the receipt of duplicate bundles because there is only a single multicast address in the "real" multicast bundle.

6.2 Perimeter Access Control

DTN access control at the source host would be no different for multicast bundles as compared to unicast bundles. If the treatment that a host should accord to a bundle is dependent on the destination address of the bundle, then the fact that some addresses are unicast addresses while others are multicast addresses may affect the access control policy itself, but not the way that it is enforced.

6.3 Hop-by-Hop Bundle Integrity and Endpoint Authentication

Under "real" multicast, the provision of hop-by-hop bundle integrity and endpoint authentication is simplified somewhat over its provision under multidestination delivery. At each router at which a bundle's path forks, the use of a single, uniform, multicast address in each of the instantiations of that bundle that are forwarded on different interfaces enables the forwarding router to calculate a single bundle BAH hash that is applicable to all of the bundles on all of the outgoing interfaces, as opposed to the unique BAH hash that each router must calculate on each outgoing interface in the case in which multidestination delivery is used. If digital signatures are used to protect the BAH hash, then each BAH hash can be encrypted with the same private key of the forwarding router, meaning that no matter how many paths into which a given bundle's path may fork at a particular router, that router will only be required to calculate one signed BAH hash, which will be applicable to all bundles going out all interfaces. If the BAH is instead protected using a message authentication code, then the message authentication code, which involves use of a symmetric cryptographic algorithm between the forwarding router and its next hop router, will require the forwarding router to calculate a unique message authentication code for each outgoing interface, assuming that a router shares a unique secret key with each of its neighbors rather than shares the same secret key with all of its neighbors.

6.4 Confidentiality and Key Management and Distribution

Under "real" multicast, in which a multicast group with its own destination address exists, key distribution and management can be performed in conjunction with the procedures that are used to set up the group and add and remove members. Assuming that all members of the group are made privy to a secret, symmetric key upon being joined to the group, this group key can be used to encrypt the CEK that is used to protect the confidentiality of the payload of any bundle. Using this mechanism, a source wanting to send confidential application data to a multicast group would encrypt the data using whatever algorithm and key desired, and then encrypt this CEK used with the multicast group's key. The bundle, therefore, would have as part of its content the encrypted application payload, the ID of the encryption algorithm used to encrypt the payload, the ID of the encryption algorithm used to encrypt the CEK, and the encrypted CEK itself. The multicast bundle, however, would only have to include a single encrypted CEK that each of the recipients in the multicast group would be able to decrypt using a secret key that is shared by all group

members, as opposed to having to include one encrypted CEK per destination, as is required in the multidestination case.

Key revocation becomes a little trickier. In the case of multidestination delivery, a key can be revoked from a particular destination simply by replacing that key with a new key and dropping the particular destination from the list of recipients. In the real multicast case, some entity must be controlling membership to the multicast group so that the destination can be removed from the group and a new, replacement, key distributed to the remaining group members.

The area of key management and distribution, however, is one in which the comparison between "real" multicast and multidestination delivery breaks down. Real multicast implies the existence of a mechanism for creating groups, joining and removing members from those groups, and enabling a single destination address to denote all group members. Multidestination delivery does not include any such mechanisms. Instead, it assumes that a source will have knowledge of all of the destination addresses to which it needs to send a particular bundle without specifying by what means the source is expected to obtain this information. The process by which a source is to be expected to come into possession of the addresses of each of the appropriate destinations to which to direct a bundle would need to be defined, and the resulting mechanism would probably be very similar to the capabilities that are needed to manage the membership of a "real" multicast group. In any case, these mechanisms would not require changes to be made to the Bundle Protocol.

Using both "real" multicast and multidestination delivery, if the content encryption key (CEK) is distributed to all destinations outside of the Bundle Protocol, no modifications would be required to the Bundle Protocol to support application layer confidentiality using either method of delivery.

6.5 End-to-End Bundle Integrity and Endpoint Authentication

In contrast with multidestination delivery, if "real" multicast is performed, no modifications are required to the Bundle Protocol to enable the signed hash in the PSH to provide end-to-end data integrity and endpoint authentication. The PSH hash would be calculated over the destination address as it is in the unicast case, and the fact that the address happens to be a multicast one rather than a unicast one is not of any import. When received by each destination in the multicast group, the value in the PSH hash would be authenticated as it is for unicast, thereby authenticating the multicast group address as the intended destination, assuming each member of a multicast group is aware of its membership in that group and of that group's address. Furthermore, because a single group address is used instead of multiple individual destination addresses, there is no requirement that addresses be stripped from the bundle to save bandwidth or to avoid multicast delivery loops as is required when multidestination delivery is performed. There is a requirement, however, that whatever algorithm is being used to perform multicast routing be inherently loop-free.

6.6 Optional Replay Detection at Destination Hosts and Security Policy Routers

There would be no change required to the bundle protocol to enable a security policy router or a destination host to optionally detect and discard duplicate bundles received. The (source, time stamp) pair would still be used to uniquely identify bundles and if two bundles with the same (source, time stamp) pair values were to be received and optional replay detection has been enabled, the second such bundle would be discarded, whether the address was that of a unicast destination, multiple destinations, or a single multicast group destination.

6.7 Security Policy Router Processing

In contrast with multidestination delivery, no changes would be required to the way that the security policy router validates the PSH hash in a multicast bundle to authenticate the bundle before considering whether or not it should be forwarded.

6.8 Bandwidth Savings

We have already seen that the bandwidth savings benefit provided by multidestination delivery depends on the payload being distributed being large, and that the bandwidth savings benefit increases with the size of the payload. The same is true for "real" multicast.

We have also seen that multidestination delivery uses about the same amount of bandwidth as unicast delivery to carry the endpoint addresses of all destinations in the bundle header or headers. In the real multicast case, much less bandwidth would be used to carry the destination address in the multicast bundle, because a single address would be used to denote all destinations. So, whereas a multidestination bundle addressed to 100 destinations would require that the region and endpoint IDs that uniquely identify each of those destinations be present in the bundle upon its initiation, a real multicast bundle addressed to a group of the same 100 destinations would require only a single destination address be present in the bundle. This bandwidth savings is obviously significant. However, it is also misleading, because additional bandwidth would be required by the group management protocols that would be required to set up the multicast address and associate it with each of the group's members. In fact, the group management protocols required to join all destinations to the group would probably require more bandwidth than the cumulative amount that would be used by the presence of all of the destination addresses in the multidestination bundle. If there are many transmissions to the group once it is set up, however, the bandwidth saved by using a multicast address in the multicast bundle versus individual unicast addresses in the multidestination bundle could be significant, and would obviously increase with each additional bundle transmitted.

6.9 Custodianship and Bandwidth-Efficient Multidestination Retransmission

When the multidestination delivery option is used in combination with custodial delivery, for purposes of custodianship, a bundle is no longer uniquely identified by its (source, timestamp) pair value. When a bundle is forwarded from a DTN router onto two different interfaces, even though the (source, timestamp) values of the bundles that are forwarded are identical, conceptually, two different bundles are instantiated. From the point at which the bundle's path diverges, the two bundle instantiations need to be accounted for separately for custodial purposes. The relevant destination addresses for a given instantiation of a bundle, along with the bundle's (source, timestamp) pair, are what uniquely define each instantiation of a multidestination bundle. (Assuming destination address stripping is used, the relevant destination addresses are simply those that remain in the bundle instantiation.) Therefore, to identify instantiations of a multidestination bundle that need to be accounted for individually for custodial purposes, the (source, timestamp, destination addresses) triple values are necessary. In order to be able to alert a custodian regarding which destination addresses failed to received a bundle so that the custodian can take advantage of the ability to identify individual instantiations of the bundle and re-forward it only to the required destinations, the format of the Custodian Signal administrative payload would have to be augmented to include fields for holding the destination endpoint addresses of the bundle about which the custodial signal is reporting. Upon receipt of a "failed" custodial signal, the custodian bundle agent would be able to use the destination address information in the custodial signal to know which destination addresses to include in the bundle that it re-forwards. Because the bundle's path may have branched one or more times after having been forwarded from the custodian, with destination addresses having been stripped off at each branch, the inclusion of only the particular destination addresses to which delivery was not successful enables the custodian to re-forward the bundle in a very bandwidth-efficient manner. Only the paths of the original multidestination delivery tree that are required for the bundle to reach the destination addresses listed in the custodial signal will be affected by the re-forwarding of the bundle.

As a concrete example, consider the DTN network topology depicted in figure 6-1. A multidestination bundle is sent from a source application to five destinations: D1, D2, D3, D4, and D5 with a class of service requesting custody reporting and the current custodian identified as the source bundle agent. When the bundle is transmitted from the source bundle agent to the next-hop router, the bundle contains all five of these destination addresses. This receiving router, which is labeled "Custodian A" elects to take custody of the bundle, so it generates a "Succeeded" custodial signal for the bundle destined for the agent administration endpoint of the source bundle agent and it modifies the current custodian ID of the bundle contain the value of its own (Custodian A's) administration endpoint ID.

When custodian A forwards the bundle, the bundle continues to contain all five of the destination addresses. Router R1 receives the bundle and elects not to take custody of it. R1 determines that the bundle needs to be forwarded on two different interfaces in order to reach all of its destinations. It forwards one instantiation of the bundle, which contains destination addresses D1 and D2, to router R2, and a second instantiation of the bundle, which contains destination addresses D3, D4, and D5 to router R5.

Upon receipt of the bundle, R2 elects not to take custody of it and forwards one instantiation of it, which contains only destination address D1, to R3 and another instantiation of it, which contains only destination address D2, to R4. Similarly, upon receipt of the bundle, R5 elects not to take custody of it and forwards it on to R6, which also elects not to take custody of it. R6 forwards one instantiation of the bundle, which contains destination addresses D3 and D4 to R7, and another instantiation of the bundle, which contains only address D5, to R8.

R3, which elects not to take custody of the bundle it receives, forwards it to D1 and the bundle agent for D1 generates a "Suceeded" custodial signal for the bundle with destination address D1 in the new custodial signal administrative payload destination address field. R4 , which elects not to take custody of the bundle it receives, forwards it to D2 and the bundle agent for D2 generates a "Succeeded" custodial signal for the bundle with destination address D2 in the new custodial signal administrative payload destination address field; R7, which elects not to take custody of the bundle it receives, forwards one instantiation of it, which contains only address D3, to D3, and another instantiation of it, which contains only address D4, to D4; the bundle agent for D3 generates a "Suceeded" custodial signal for the bundle with destination address D3 in the new custodial signal administrative payload destination address field and the bundle agent for D4 generates a "Suceeded" custodial signal for the bundle with destination address D4 in the new custodial signal administrative payload destination address field. R8, which elects not to take custody of the bundle it receives, forwards it to D5, and the bundle agent for D5 generates a "Suceeded" custodial signal for the bundle with destination address D5 in the new custodial signal administrative payload destination address field. Each of the "Succeded" custodial signals that were generated by each of the five destination host bundle agents is received at the administrative endpoint for Custodian A, alerting Custodian A to the fact that custody transfer for all instantiations of the bundle has been completed. This describes what happens when the multidestination delivery of a bundle from the source to the five destinations completes without mishap.

Now let's consider what happens when the delivery does not occur as smoothly. Suppose that the above delivery scenario occurs, except that router R6 elects to discard the bundle altogether due to some sort of failure. In this case, R6 would generate a "Failed" custodial signal for the bundle with destination addresses D3, D4, and D5 in the new custodial signal administrative payload destination address field. Upon receipt of this "Failed" custodial

signal, Custodian A could re-forward the original bundle, except instead of including all of

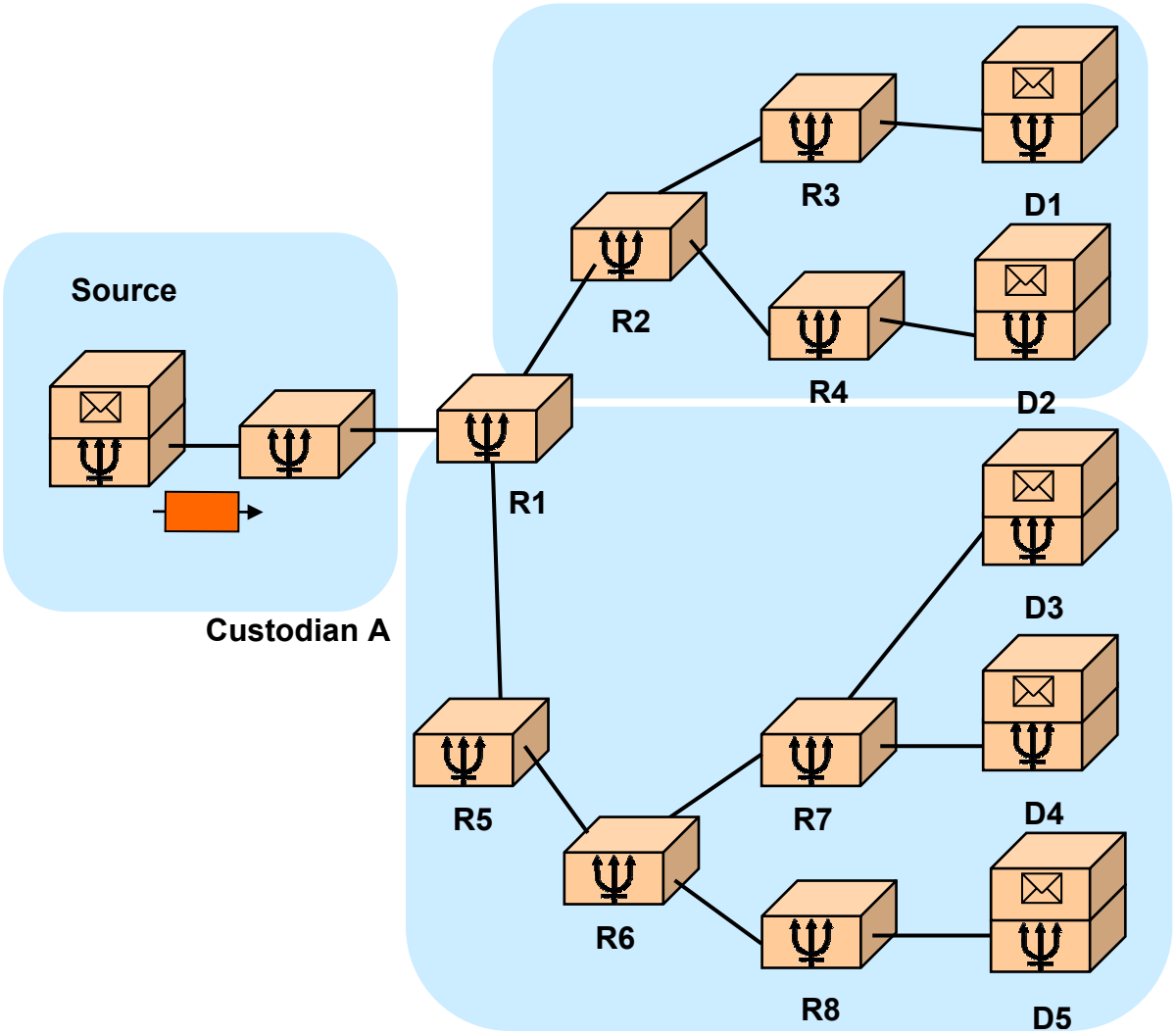


Figure 6-1. Custodianship and Bandwidth-Efficient Retransmission

its original destination addresses, it would only include the destination addresses, D3, D4, and D5, that had appeared in the "Failed" custodial signal that it received. None of the other branches on the original multidestination delivery tree would be affected by the re-forwarded bundle.

When multicast is used, on the other hand, there is no easy available mechanism for identifying individual instantiations of a multicast bundle that may need to be accounted for

individually for custodial purposes. A single multicast address is used to denote all destinations, and the multicast bundle on one branch of a delivery tree is not distinguishable from a different instantiation of that multicast bundle on a different branch. If multicast is used, there is no point in augmenting the Custodial Signal administrative payload format to include the destination (multicast) address, because this address does not help at all to distinguish among the various instantiations of a given bundle on the multicast delivery tree. The inability to uniquely identify individual instantiations of the multicast bundle, however, can result in wasting a significant amount of bandwidth during custodial retransmission.

Upon receipt of a "failed" custodial signal, the custodian bundle agent for a multicast bundle would simply include the same multicast destination address in the bundle that it re-forwards. A simple retransmission of a multicast bundle on a given interface would result in all downstream paths of the delivery tree receiving the retransmitted bundle. The bundle would be re-forwarded on all the branches of the multicast delivery tree indiscriminately, whether it had already been received successfully on some paths or not, because there is no mechanism for distinguishing the branches along which the bundle may have already been received successfully from those along which delivery has failed. There would be no way of limiting the path of the retransmitted bundle to only that single branch of the delivery tree on which the retransmission is required. In terms of the example discussed above, upon receipt of the "Failed" custodial signal generated by R6's bundle agent, if Custodian A wishes to re-forward the bundle so that it can be received by R6 and then forwarded on to its intended destinations, it must re-forward the bundle to the multicast destination address, which would cause the bundle to travel along all paths of the multicast tree, to all destinations, rather than simply targeting those destinations beyond R6.

This inability to forward with discrimination could potentially result in a significant amount of wasted bandwidth on bundle retransmissions. In this respect of enabling bundle instantiations to be identified uniquely, the multiple destination addresses found in a multidestination bundle are advantageous over the single multicast destination address found in a multicast bundle. If multicast delivery is ever fully developed and defined for use in DTN, the ability to retransmit multicast bundles bandwidth-efficiently would require the definition of some sort of mechanism for identifying unique instantiations of multicast bundles so that these instantiations can be accounted for individually for custodial purposes.

6.10 Multidestination versus Multicast Table Summary

The following table summarizes the differences between the security ramifications of using multidestination delivery versus using "real" multicast.

Table Summarizing the Ramifications of Securing a "real" multicast bundle versus a bundle addressed to multiple destinations

Security Service	How a multidestination delivery option would affect the security service	How a "real" multicast capability would affect the security service
Access Control	If the access control policy at any source bundle agent takes destination addresses into consideration, the access control policy would need to be amended to account for the possibility that a single bundle may contain two destination addresses that are intended to be treated differently from each other.	No impact.
Hop-by-hop Data Integrity and Endpoint Authentication	Routers at which a bundle's path branches will need to calculate separate BAH hash values for each interface on which the bundle will be forwarded.	If digital signatures are used to protect the BAH hash, a router need only calculate a single hash per received bundle, regardless of the number of interfaces on which the bundle will be forwarded. If message authentication codes are used to protect the hash, a router must calculate one hash per interface on which the bundle will be forwarded, because it shares a unique symmetric key with each neighboring router.
Lack of Replay Detection at Arbitrary Routers	The risk posed by replays increases dramatically with the use of multidestination delivery.	As with multidestination delivery, the risk posed by replays increases dramatically with the use of multidestination delivery.
Application Data Confidentiality	If distribution of the content encryption key (CEK) to all destinations is performed out of band, no modifications would be required to the Bundle Protocol to support application data confidentiality. However, such distribution is not really feasible. Modifications would be required to	If distribution of the content encryption key (CEK) to all destinations is performed out of band or as part of a separate group key management and distribution protocol, no modifications would be required to the Bundle Protocol to support application data confidentiality.

	distribute the CEK and the following optional related information as part of the multidestination delivery bundle: • ID of the symmetric encryption algorithm used for confidentiality (the algorithm with which the CEK is used) • ID of the asymmetric encryption algorithm used to encrypt the CEK • For each destination endpoint: – ID of the key used to encrypt the CEK – The result of encrypting the CEK with that key	The same modifications as are required to support multidestination delivery would be required to support "real" multicast delivery, except that the ID of the key used to encrypt the CEK and the result of encrypting the CEK with that key need only be included once in the bundle and it would be applicable to all multicast group member destinations. There is no need to include a separate key ID and encrypted CEK for each multicast destination. So, real multicast requires two additional fields whereas multidestination delivery requires 2 additional fields per destination.
End-to-End Data Integrity and Endpoint Authentication	New fields and new mechanisms must be defined to enable a single PSH hash to provide end-to-end security for the bundle, yet enable destination addresses to be stripped from the bundle as they are no longer needed on a certain path of the delivery tree. • Each destination address in the bundle must have an associated field that contains a hash of that destination address. • The PSH hash should be calculated over the entire bundle except for – The BAH – All mutable fields, such as the custodian field and the	No impact. The PSH hash value would be calculated and checked the same way that it is now.

	sender field – All destination endpoint addresses To validate the hash, a receiving node would first ensure that every destination address present in the bundle has a corresponding hash value which is correct, and then compare the received PSH hash value with the PSH hash value as above.	
Optional Replay Detection at Destination Hosts and Security Policy Routers	No change is required. The (source, time stamp) pair value is still used to uniquely identify bundles.	As with multidestination delivery, no change is required.
Security Policy Routers	To validate the PSH hash, a security policy router would first ensure that each destination address present in the bundle has a corresponding hash value which is correct, and then compare the received PSH hash value with the PSH hash value as calculated to include hashed destination addresses, but not real destination addresses (as described in the end-to-end integrity and endpoint authentication item above).	No impact. The security policy router would calculate and check the PSH hash value in the same way that it is now.
Avoiding delivery loops	Addresses must be stripped off at each fork in the bundle's delivery tree.	No impact. However, the multicast routing protocol would need to be inherently loop-free.
Bandwidth Savings	Because a bundle contains every destination address to which it will be delivered, with respect to the amount of bandwidth required to carry destination information, multidestination delivery is about the	Because a bundle contains only a single multicast address, with respect to the amount of bandwidth required to carry destination information, multicast enjoys a huge savings over

	same as unicast delivery, with some savings being incurred in multidestination delivery by using the data dictionary for addresses that share the same region IDs. Slightly more bandwidth is required to support optional application data confidentiality in the multidestination case as compared with the real multicast case. This bandwidth amounts to the bits in one encrypted CEK field per destination address in the bundle.	multidestination delivery. However, multicast also requires bandwidth to be used by a group management protocol needed to set up the group and join and resign members from the group, so the savings is not as large as it initially might seem. This is a one-time initial setup cost, plus some cost to maintain the group membership information. If the group is used infrequently, this cost is probably comparable to the bandwidth cost of including the destination addresses in the bundle, as occurs in the multidestination delivery case. If the group is used frequently, however, the savings can be significant.
Custodianship	The (source, timestamp, relevant destination addresses) triple values can be used to uniquely identify instantiations of a multidestination bundle that need to be accounted for individually for custodial purposes. Amending the custodian signal administrative payload format to include the destination addresses found in the bundle would enable the custodian to re-forward the bundle bandwidth-efficiently, so that only the particular branch of the delivery tree requiring the retransmission would receive the retransmitted bundle.	Individual instantiations of a multicast bundle are not uniquely identifiable. Without a modification to the bundle protocol to enable individual multicast bundle instantiations to be identified, retransmissions can be potentially very wasteful of bandwidth, because all downstream branches of the delivery tree will receive retransmitted bundles, even those that do not require them.