

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 21, 2008

J. Wyllie
Ohio University
W. Eddy
Verizon
J. Ishac
W. Ivancic
NASA
S. Ostermann
Ohio University
November 18, 2007

Automated Bundle Agent Discovery for Delay/Disruption-Tolerant
Networking
draft-wyllie-dtnrg-badisc-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on May 21, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Internet-Draft

DTN Bundle Agent Discovery

November 2007

Abstract

In Delay/Disruption-Tolerant Networking (DTN), Bundle Agents form an overlay network that forwards DTN bundles between their source and destination applications. This document describes a mechanism that Bundle Agents can use to discover when they are in contact with one another and optionally provide information on the additional properties of current or future contacts, such as duration and capabilities. This information can be used to trigger bundle forwarding or make future bundle scheduling decisions.

Table of Contents

1.	Introduction	3
1.1.	Terminology	4
1.2.	Protocol Overview	5
2.	Message Format	6
2.1.	Bundle Processing Control Flags	6
2.2.	Autodiscovery Message Format	7
2.3.	Autodiscovery Flags	8
2.4.	Capabilities	8
2.4.1.	Convergence Layers Supported	9
2.4.2.	Grokability of Schemes	9
2.4.3.	Structure of the Capabilities SDNVs	9
2.5.	Autodiscovery Protocol Type	10
3.	Convergence Layer Behavior	11
4.	Processing Behavior	12
4.1.	Exported Bundling Agent Interface	12
4.2.	Bundling Daemon Autodiscovery Request Handling	12
4.3.	Processing of Received Autodiscovery Bundles	13
4.4.	Adding and Overwriting Contact Times	13
4.5.	Bundle Status Reports	14
5.	Examples	15
5.1.	Sensor Node to Sensor Node	15
5.2.	Deep-Space Probe to Earth Discovery	15
5.3.	LEO Satellites to Terrestrial Center	15
5.4.	Updating and Overwriting Contacts	16
5.5.	A Note on Flexibility	17
6.	Security Considerations	18
6.1.	Security in a Trusted Network	18
6.2.	Security in an Untrusted Network	18
7.	IANA Considerations	20

8.	Acknowledgements	21
9.	Informative References	22
	Authors' Addresses	24
	Intellectual Property and Copyright Statements	26

[1.](#) Introduction

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

The Delay/Disruption-Tolerant Networking (DTN) architecture [[RFC4838](#)] describes an overlay network of Bundle Agents (BAs). Each BA manages the forwarding of bundles during contacts with other BAs and queues bundles between contacts. As the bundle format and basic forwarding operations were designed, it became clear that an automated means for BAs to discover neighboring nodes, their neighbors' capabilities, and contact times was needed. This document describes such a mechanism as well as the mechanism's relationship to potential DTN routing protocols.

While the mechanism described within this document is specifically designed to convey peer-to-peer contact information between two BAs that are adjacent in the overlay topology, it also can be used as one of the building blocks to enable multi-hop routing decisions between BAs. To clarify, consider a typical Internet routing protocol (e.g. OSPF [[RFC2328](#)]) which has two main types of messages: (1) "Hello" messages used to automatically discover connectivity with other directly-reachable peers, and (2) messages used to update databases of connectivity information for further hops. The mechanism described in this document satisfies the first function ("Hello" messaging) within a DTN overlay by allowing Bundle Agents to automatically discover other nodes to which they can directly forward bundles.

Previously, implementations of DTN Bundle Agents provided these functions using either static, manual configuration or ad-hoc, custom advertisement mechanisms that were neither formally defined nor independent of convergence layers. This document provides a means that will interoperate between differing BA implementations and allow them to exchange contact information over any shared convergence

layers.

This exchange of contact information can be useful even when contact properties are known in advance. It can be used to verify the pre-configured information or reveal discrepancies in contact properties (expected duration, expected future contact times, etc.).

The remainder of this document contains a high-level overview of the discovery mechanism's design in [Section 1.2](#), a specification of the message formats used in [Section 3](#) and [Section 2](#), and a description of the rules for generating and processing these messages in [Section 4](#). In addition, [Section 5](#) contains example message exchanges to clarify

the protocol's operation.

[1.1](#). Terminology

Defining neighboring nodes on a typical network is fairly straightforward. However, the overlay nature of a DTN complicates the definition of neighboring DTN BAs. Furthermore, communication may be unidirectional in DTNs: for example, DTN node A may be able to send to DTN node B, but not vice versa. We therefore define the following terms:

Neighbor: Two nodes are said to be 'neighbors' over a convergence layer if and only if the nodes can communicate symmetrically directly over that convergence layer. For example, assume A, B, and C are all DTN BAs, and A is connected to B over one TCP convergence layer adapter while B is connected to C over another TCP convergence layer adapter, but TCP segments between A and C are blocked by a firewall, so no direct contact between them is possible. A and C are not neighbors while B is a neighbor to both A and C, regardless of how many IP-layer hops exist between them. The convergence layers here imply connection symmetry: if B were connected to C over a convergence layer which only permitted unidirectional transmission (e.g. FLUTE), B may not be a neighbor to C, and instead might be one of the following:

Pitcher: When a contact exists between two nodes, but they are not neighbors because they lack bidirectional connectivity, one is called a "pitcher", and the other a "catcher". The node that can only send over the convergence layer takes the role of the

pitcher. A pitcher meets the definition of "neighbor" in every other way. As an example, A is a pitcher to B over a particular convergence layer adapter if A can directly send and B can directly receive using that convergence layer.

Catcher: A node that can only receive over a convergence layer adapter fills the role of catcher. A catcher meets the definition of "neighbor" in every other way. As an example, A is a catcher to B over a particular convergence layer adapter if A can directly receive and B can directly send using that convergence layer.

In-Contact: A node X is said to be 'in-contact' to another node Y if X is either a neighbor, pitcher, or catcher to node Y.

The terms 'pitcher' and 'catcher' were borrowed from JPL's implementation of the Asynchronous Message Service.

[1.2.](#) Protocol Overview

BA autodiscovery is complicated by the diverse set of deployment environments in which DTN may be used. As stated in [[RFC4838](#)], deployment environments may include networks with any combination of:

- o Delays which are orders of magnitude larger than terrestrial networks
- o 'High' bit-error rates, either overall or bursty in nature
- o Relatively high asymmetry in network properties compared to the terrestrial Internet

To adequately accommodate all of these environments, the protocol in this document provides two levels of information: (1) domain-specific and (2) domain-independent, where a domain refers to a particular class of deployment environment. Domain-specific information typically is needed to determine the expected duration and other properties that are clearly determined by different factors. For instance, deep-space networks have known and stable orbital properties while terrestrial vehicular ad-hoc networks have unknown

and fairly random motion and link fading. Domain-independent information directly specifies contact properties that can be determined and communicated independently of any particular deployment environment. As a result, it contains only basic information that is consistently useful across environments such as the remaining amount of storage space and EIDs in use.

This dual-level design provides flexibility in determining information about other BAs. For example, DTN-enabled nodes on a local sensor network may send domain-specific bundles including both power levels and received signal strengths to a central point that determines contact times and durations. In contrast, DTN-enabled deep-space probes may send no domain-specific information at all, instead relying on preconfigured schedules or internal ephemeris data to determine contacts, and using autodiscovery only as a status and sanity check of the DTN-level networking configuration.

The protocol also allows for third-party configuration of contact information (a third DTN node may inform two other nodes about their expected contact times). This can facilitate remote configuration. For instance, 'dumb' sensor network nodes may not have complete information to determine optimal contact times on their own. However, a single, more sophisticated node can communicate this information to the entire sensor network. The potential for misuse of this capability raises some security concerns that are discussed in [Section 6](#).

[2.](#) Message Format

[2.1.](#) Bundle Processing Control Flags

Bundle Processing Control Flags MUST adhere to the following guidelines:

- o Bundle Fragment -- Nominally, autodiscovery bundles will be small enough that fragmentation is never needed. However, unless fragmentation is forbidden by a domain-specific application, autodiscovery bundles may be fragmented. This bit MUST be set according to the fragmentation guidelines in [\[SB07\]](#).
- o Administrative Record -- Autodiscovery bundles are not administrative records; this bit MUST be clear.

- o Fragmentation Allowed -- As stated above, autodiscovery bundles may be fragmented at the discretion of the domain-specific rules.
- o Custody Transfer Required -- Since the autodiscovery information is advisory by nature, it does not require reliable transmission and custody transfer should never be required; this bit SHOULD always be clear. Furthermore, since bundles used for autodiscovery move over only a single overlay-hop, a Bundle Status Report is sufficient for acknowledgement without invoking custody transfer semantics.
- o Destination Endpoint is a Singleton -- In certain scenarios, it may make sense to define contact timings for large sets of nodes. Therefore, autodiscovery may take place between non-singleton endpoints, and autodiscovery messages may be multicast at the bundle layer and/or below.
- o Acknowledgement by application is requested -- As there is no 'application' associated with autodiscovery, this should not be necessary and the bit SHOULD be cleared.

Therefore, the low-order Bundle Processing Control Flags SHOULD be set to:

000Z0Y0X

where the value of X is determined by whether the current bundle is a fragment, the value of Y is determined by whether the domain-specific case allows for fragmentation, and the value of Z is determined by the number of intended destinations.

[2.2.](#) Autodiscovery Message Format

In order to work regardless of the convergence layer adapters that are available, BA autodiscovery uses DTN bundles to communicate its data. More specifically, all autodiscovery data is encoded in the payload block of the bundle. Some of the autodiscovery fields use Self-Delimiting Numeric Values (SDNVs) [[I-D.irtf-dtnrg-sdnv](#)] within the messaging format, as in several other parts of the Bundle

Protocol and its extensions.

Any bundles related to autodiscovery MUST contain the domain-independent portion of the autodiscovery bundle, followed optionally by the domain-specific portion. The fields should be ordered as follows:

- o Version -- 16-bit value specifying the autodiscovery version to which this bundle adheres. This document defines version 0. A daemon MUST drop any bundle with any other version number.
- o Autodiscovery Flags -- SDNV-encoded field describing the contents of the autodiscovery header. The meanings of the individual bits within this field are defined in [Section 2.3](#).
- o Contact Start Time -- An optionally included field indicating the beginning point of a contact. The time is encoded as a DTN timestamp as specified in [\[SB07\]](#).
- o Contact End Time -- An optionally included field indicating the ending point of a contact, encoded as a DTN timestamp as specified in [\[SB07\]](#).
- o Pitcher -- An endpoint ID reference as specified in [\[SB07\]](#) referring to the node designated as the 'pitcher' in the connection.
- o Catcher -- An endpoint ID reference as specified in [\[SB07\]](#) referring to the node designated as the 'catcher' in the connection.
- o Capabilities -- SDNV-encoded field describing the capabilities of the node referred to by this block. This is described in [Section 2.4](#).
- o Autodiscovery Protocol Type -- A 16-bit field that indicates the format of the domain-specific data and domain-specific procedures. This is described in [Section 2.5](#).

The bits of the Autodiscovery Flags indicate which optional fields of the Autodiscovery Block are present, and are explained in detail below:

Bit 0 -- Start Time Present
Bit 1 -- End Time Present
Bit 2 -- Connection is bidirectional
Bit 3-7 -- Reserved

- o Start Time Present -- If this bit is set, then the autodiscovery message includes the Contact Start Time field. A sender can either indicate a specific time when a predicted future contact may start, or this bit can be clear (and the corresponding field omitted) to indicate an immediate contact (or, put differently, that the nodes can commence communication upon processing of the autodiscovery bundle). If this bit is set, then a Contact Start Time field **MUST** be included in the autodiscovery message; if the bit is cleared, then that field **MUST NOT** be present.
- o End Time Present -- If this bit is set, then the autodiscovery message includes the Contact End Time field. If the field is omitted, this means that the nodes will remain in contact for the indefinite future. If this bit is set, then a Contact End Time field **MUST** be included in the autodiscovery message; if the bit is cleared, then that field **MUST NOT** be present.
- o Connection is bidirectional -- If this bit is set, it signifies that not only can the pitcher send to the catcher, but vice versa as well. This establishes the two nodes as neighbors.

[2.4.](#) Capabilities

Though many capabilities could be considered domain-specific, some capabilities are nearly ubiquitous in utility and relevance. Those capabilities are defined here. To reduce bit overhead when not all capabilities are relevant or desired, the capabilities are set as a two-level SDNV. The top-level SDNV specifies which second-level SDNVs are present in the bundle. Each second-level is a categorized SDNV with flags signifying the presence or absence of some capability.

Currently, two second-level SDNVs are defined. Therefore, the top-level capabilities SDNV has the following values:

- o Convergence Layers Supported -- Setting this bit signifies the presence of the convergence layer capabilities SDNV.
- o Grokability of Schemes -- Setting this bit signifies the presence of the schemes capabilities SDNV. Clearing this bit signifies that the SDNV is absent from this bundle.

For any capabilities that are not defined, default values may be chosen by the BA in an implementation-specific manner.

[2.4.1.](#) Convergence Layers Supported

This capabilities SDNV defines over which convergence layers the BA can communicate. Setting the convergence layer bit indicates that the convergence layer is understood; clearing indicates that it is not. Currently, the SDNV is defined as follows:

- o Bit 0 -- Single-packet UDP (one bundle per UDP packet)
- o Bit 1 -- TCPCL, defined in [[I-D.irtf-dtnrg-tcp-clayer](#)]
- o Bit 2 -- LTP, defined in [[I-D.irtf-dtnrg-ltp](#)]
- o Bit 3 -- Saratoga, defined in [[I-D.wood-tsvwg-saratoga](#)]
- o Bit 4 -- FLUTE, defined in [[RFC3926](#)]
- o Bit 5-7 -- Reserved

[2.4.2.](#) Grokability of Schemes

This capabilities SDNV defines which EID schemes the BA can understand. Currently, the SDNV defines the following:

- o Bit 0 -- The BA understands the "dtn" scheme
- o Bit 1 -- The BA understands the "ipn" scheme

[2.4.3.](#) Structure of the Capabilities SDNVs

All capabilities information is included as part of the "capabilities" section of the discovery header. The top-level SDNV MUST be present. If any of its bits are set, the accompanying SDNV MUST be concatenated immediately following the top-level SDNV. If multiple second-level SDNVs are indicated, they MUST be concatenated together in the order in which they are indicated in the top-level

[2.5.](#) Autodiscovery Protocol Type

If an autodiscovery bundle contains domain-specific information, this field indicates the structure of the payload data. Effectively, this field defines the "domain" of the domain-specific portion.

Currently, the following fields are defined:

- o 0x00 -- Basic - no domain-specific information; payload MUST be empty
- o 0x01 -- DMC-like Satellites - payload format defined in a separate document
- o 0x02 through 0xEF -- Currently undefined; can be defined in later documents
- o 0xF0 through 0xFF -- Experimental

Domain-specific information, in a bundle which has any, immediately follows the payload of the autodiscovery message.

3. Convergence Layer Behavior

Due to the nature of an overlay network, autodiscovery may take place over many different convergence layers. In addition, depending on the convergence layer used, convergence layer discovery might be necessary prior to bundling discovery. For instance, when using Bluetooth as a convergence layer, paired devices must undergo device discovery and service discovery before encapsulated data (such as the autodiscovery bundle) can be sent. Therefore, initiating autodiscovery over Bluetooth for previously unknown devices will require cooperation between device/service discovery and the bundling layer [[BTSPEC](#)]. If any data obtained in service discovery is relevant, it could be communicated as domain-specific information in the discovery bundle. For the common terrestrial Internet protocols TCP and UDP, discovery over TCP implies that the devices have bidirectional communication where discovery over UDP can use unidirectional communication and multicast. TCP requires handshaking prior to bundle communication; UDP does not.

Certain capabilities of a convergence layer node may be relevant in the bundling autodiscovery process. For instance, when using TCP, it may be relevant to communicate the round-trip time (RTT) to the bundle layer as it may impact routing decisions. Extra features like this can be included as part of a domain-specific protocol.

The capabilities as described in are explicitly for the bundle layer. Convergence layer-specific information (aside from their existence) is purposefully not included there. Communicating relevant convergence layer-specific information can be done in a domain-specific protocol.

Due to the myriad convergence layers over which bundles are expected

to travel, exact mechanisms for executing bundle-layer autodiscovery over convergence layers are outside the scope of this document and are left as future work.

[4.](#) Processing Behavior

Due to the domain-specific portions of autodiscovery, exact processing of autodiscovery bundles can vary widely between domains. Therefore, only a general framework for autodiscovery bundle processing is given in this document while exact autodiscovery procedures for particular domains is defined separately.

Autodiscovery functions can either be implemented within a BA, or as one or more external 'helper' applications. An interface supporting both types of implementation are outlined below. Although a helper application may assist in interpreting autodiscovery payloads, autodiscovery is not itself a DTN application. Autodiscovery bundles are addressed to the administrative endpoints of BAs themselves, whose EIDs are not suffixed with an application de-muxing token, or otherwise altered. In this way, autodiscovery works for all EID schemes regardless of how EIDs are constructed within them.

[4.1.](#) Exported Bundling Agent Interface

In addition to providing the basic bundling API as defined in [\[SB07\]](#), the daemon SHOULD allow for explicit values specified for the following fields in an outgoing bundle:

- o Local Convergence Layer Adapter for transmission

- o Destination EID -- SHOULD be set to dtn:discovery for discovery of BAs with unknown EIDs; may be set to an actual EID if it is known
- o Bundle Payload -- to include the autodiscovery message
- o Lifetime -- A lifetime of 0 squelches forwarding of discovery bundles.

Exactly how this interface is supplied (e.g. via IPC, RPC, etc) is an implementation-specific consideration.

A BA SHOULD use the "expedited" priority for all autodiscovery bundles, given that they may affect immediate decisions and are typically small bundles, although this interface MAY be extended in some implementations to allow the priority to be specified.

[4.2.](#) Bundling Daemon Autodiscovery Request Handling

Requests for autodiscovery will be honored as long as the supplied parameters are valid and the requested convergence layer adapter exists and is capable of sending bundles.

Upon receiving a request to initiate autodiscovery, the BA MUST initiate this procedure regardless of its current knowledge of contacts. It will, however, maintain its current contact information base entries at least until autodiscovery completes.

[4.3.](#) Processing of Received Autodiscovery Bundles

Processing of incoming autodiscovery responses proceeds differently depending on the Autodiscovery Type field:

Zero (fully domain-independent): A domain-independent bundle can be processed entirely within any bundle agent capable of autodiscovery. The contact information given in the autodiscovery message can be used to update the contact information base at the bundle agent's discretion.

Non-Zero (domain-specific): Upon reception of an Autodiscovery Bundle with a domain-specific payload (indicated by a non-zero

Autodiscovery Type code), the bundle agent can either process the domain-specific payload itself (if it understands the autodiscovery protocol used) or may hand the entire payload block to external autodiscovery code that can suggest changes to the bundle agent's contact information base after processing.

It is possible that after receiving an Autodiscovery Bundle, a bundle agent will wish to advertise its own view of contact information in response. This is fully at the discretion of the bundle agent or external autodiscovery code.

[4.4.](#) Adding and Overwriting Contact Times

When a valid new contact is received, it will be updated according to the following rules:

More recently-received and valid updates will always be added to the BA. Only the contact window, pitcher EID, and catcher EID are considered in decisions to keep or eliminate older discovery entries.

If no contact information about the pitcher and catcher currently exists, or if contact information exists but does not overlap with the time period specified in the received bundle, then the contact information will be added to the BA without modification to existing entries.

If contact information about the pitcher and catcher currently exists and overlaps the time period specified in the received bundle at any point in time, the now "outdated" entry should be deleted in its entirety, including the part which does not overlap with the newly-

received bundle. Other entries pertaining to the pitcher and catcher which have no time overlap with the newly received bundles SHOULD be retained.

More examples to illustrate the updating procedure can be found in [Section 5.4](#).

[4.5.](#) Bundle Status Reports

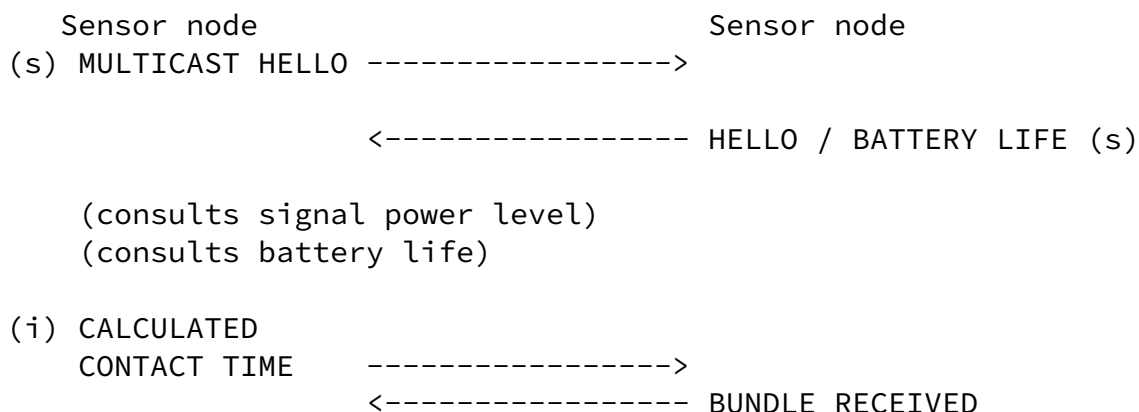
As any bundle sender can request a Bundle Status Report (a Bundle Reception Status Report in this case), Autodiscovery Bundles may also

be acknowledged using this mechanism, if desired. These could be utilized by a bundle agent or external autodiscovery code in order to suppress retransmission of autodiscovery information, for instance, when sending over unreliable convergence layer adapters that lack feedback of their own (e.g. single-packet UDP). This is not necessary in all domains, and since information gained through autodiscovery is generally advisory, it is assumed that Bundle Status Reports will not be frequently requested for Autodiscovery Bundles.

[5.](#) Examples

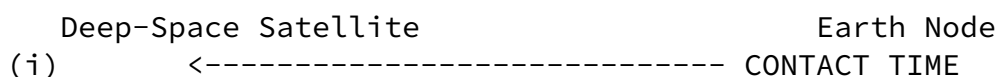
[5.1.](#) Sensor Node to Sensor Node

In this example, dozens of sensor nodes are deployed in an area without any infrastructure and must discover one another in order to communicate at the bundle layer. All nodes are connected on the same network. Exact discovery would depend on the domain-specific protocol used, but it could look similar to the the diagram below:



5.2. Deep-Space Probe to Earth Discovery

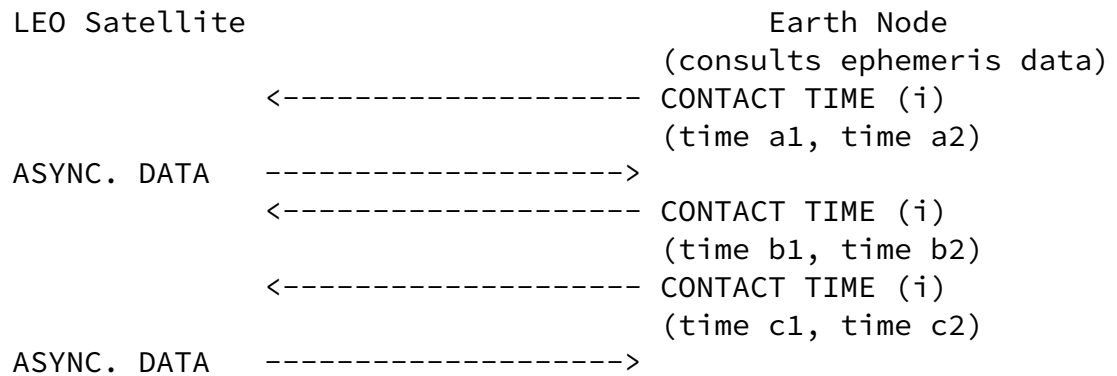
In this example, a deep-space node has a set contact schedule with Earth nodes established prior to each window of communication. Due to some event that causes re-allocation of ground-station resources, changing the space probe's schedule may be necessary.



5.3. LEO Satellites to Terrestrial Center

Many LEO satellites are particularly concerned with optimizing link utilization, and primarily send data downwards. Contacts with ground stations can be brief, on the order of several minutes, so synchronizing clocks and getting the BA onboard a satellite to be ready to forward at the correct time can be accomplished through autodiscovery. The authoritative timing figure would be the terrestrial station, which would asynchronously send contact time information to the satellite on some schedule that let it optimize its forwarding decisions to avoid reactive fragmentation due to interrupted transfers and possibly to ensure that transmission

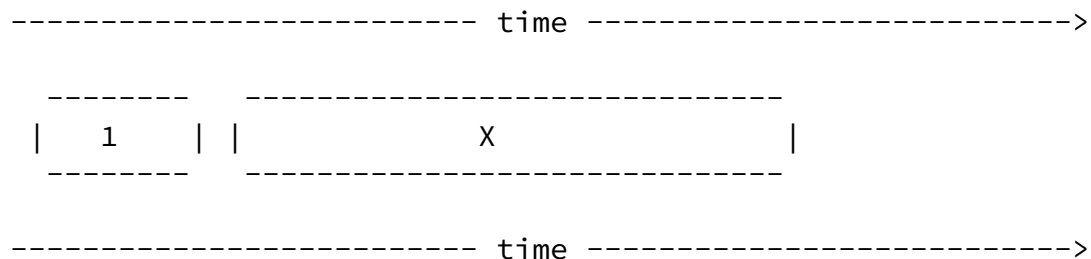
preconditions are met before the contact (transmitter powered up, on correct frequency, using correct modulation and coding, etc.).



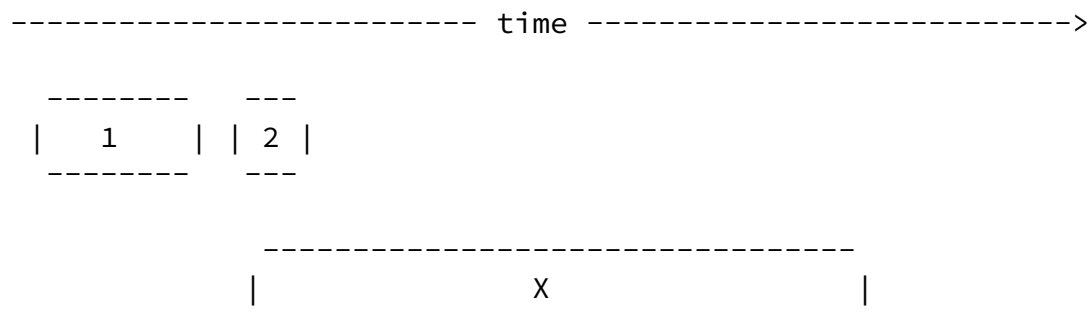
[5.4.](#) Updating and Overwriting Contacts

This section provides examples for updating and overwriting contact information depending on discovery bundles received.

Existing contact windows are numbered 1...n: the received bundle is labeled X.



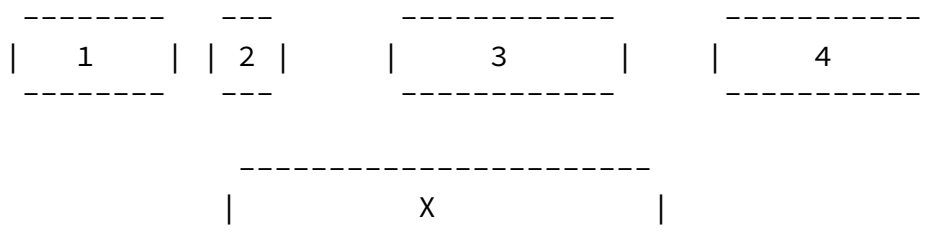
Procedure: 1 is kept; X is added



----- time ----->

Procedure: 1 is kept, 2 is removed, and X is added

----- time ----->



----- time ----->

Procedure: 1 is kept, 2 and 3 are removed, 4 is kept, and X is added.

[5.5.](#) A Note on Flexibility

The lack of structure in the protocol is due to the necessary flexibility in deployment. The domain-independent timing packets are meant to give a universal way to apply discovery and timing principles to any DTN environment that does not currently exist. The domain-dependent part is meant to add the needed flexibility to adhere to all DTN environments. Consequently, there is no 'typical' example that generalizes all DTN neighbor discovery.

More specific examples of neighbor discovery can be found in documentation describing domain-specific exchanges.

[6.](#) Security Considerations

Nearly all threats that apply to other neighbor discovery protocols apply to DTN neighbor discovery. As suggested in [[RFC3756](#)], many of these are solved by ensuring a 'trusted' network. Networks that are not trusted are subject to different limitations (and expectations), so we deal with probable attacks and mitigations for each network separately.

[6.1.](#) Security in a Trusted Network

Unlike general networking in the Internet, in DTN scenarios, there are many cases where a 'stranger' on the network is an unexpected condition. For example, sensor networks in poorly-connected environments are often controlled by a single organization which grants connection to only approved nodes. This may be implemented through link-layer encryption, keying of spreading sequences, or other means. Currently, as these two deployment environments are the two major proposed uses for DTN, securing the 'trusted' network against attack deserves further consideration.

The most likely attack on a trusted network is an attack where a node infiltrates a closed network, masquerading as a trusted node using a trusted address. To mitigate this attack, one could pre-share symmetric keys and use encryption with nodes that are allowed on the closed network. Blocks to accomplish this are defined in [[DTNSEC](#)]. All aspects of neighbor discovery, therefore, could be encrypted using these keys. In this manner, no nodes lacking the pre-shared key (and, by extension, permission to use the network) can communicate on the network.

In nodes that can afford the overhead of public-key cryptography, [[DTNSEC](#)] also supports X.509 certificate processing. This would be

more desirable for nodes that may communicate with many entities (such as the international space station) that each require separate keys (such as member nations with varying levels of cooperation). If necessary, this would also help protect the system from a single point of failure as in the pre-shared key system, only one pre-shared key needs to be compromised to compromise an entire system.

6.2. Security in an Untrusted Network

Though many currently proposed deployments of DTN involve closed networks, future DTN networks may include untrusted public networks [[haggle](#)]. For example, a DTN node on a public transportation system may send a bundle to a curbside receiver which would then forward it through the network. As part of public transportation, it is impossible to guarantee trust to nodes on the network, so the above

Wyllie, et al.

Expires May 21, 2008

[Page 18]

Internet-Draft

DTN Bundle Agent Discovery

November 2007

system will not work.

In this environment, likely attacks generally are some variation of man-in-the-middle attacks. For instance, a PDA on the public transportation system may masquerade as a router and sniff traffic or deny it altogether. As stated in [[RFC4251](#)], there is no known way to prevent these attacks outright.

To help mitigate these attacks, we propose a 'fingerprint' system combined with public-key cryptography. If a node considers the above attack serious enough, it can require public-key cryptography as provided in [[DTNSEC](#)], storing the fingerprint of the server's public key. In a man-in-the-middle attack, the fingerprint would change, and the user could be notified in an appropriate way. This system is very similar to the fingerprinting system currently in widespread use in SSHv2 [[RFC4251](#)].

Due to the potentially severe limitations on processing power and bandwidth of DTN nodes, all of the above security is optional with respect to neighbor discovery. In many deployments, the costs may simply outweigh the benefits.

[7.](#) IANA Considerations

This document does not update or create any IANA registries.

[8.](#) Acknowledgements

Some of the work on this document was performed at NASA's Glenn Research Center with funding provided by NASA's Earth Science Technology Office.

[9.](#) Informative References

[BTSPEC] "Specification of the Bluetooth System", July 2007.

[DTNSEC] Symington, S. and S. Farrell, "Bundle Security Protocol

Specification", [draft-irtf-dtnrg-bundle-security](#) (work in progress), April 24 2007.

[I-D.irtf-dtnrg-ltp]

Ramadas, M., Burleigh, S., and S. Farrell, "Licklider Transmission Protocol - Specification", [draft-irtf-dtnrg-ltp-07](#) (work in progress), October 2007.

[I-D.irtf-dtnrg-sdnv]

Eddy, W., "Using Self-Delimiting Numeric Values in Protocols", [draft-irtf-dtnrg-sdnv-00](#) (work in progress), September 2007.

[I-D.irtf-dtnrg-tcp-clayer]

Demmer, M. and J. Ott, "Delay Tolerant Networking TCP Convergence Layer Protocol", [draft-irtf-dtnrg-tcp-clayer-00](#) (work in progress), July 2007.

[I-D.wood-tsvwg-saratoga]

Wood, L., McKim, J., Eddy, W., Ivancic, W., and C. Jackson, "Saratoga: A Scalable File Transfer Protocol", [draft-wood-tsvwg-saratoga-00](#) (work in progress), October 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, [RFC 2328](#), April 1998.

[RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", [RFC 3756](#), May 2004.

[RFC3926] Paila, T., Luby, M., Lehtonen, R., Roca, V., and R. Walsh, "FLUTE - File Delivery over Unidirectional Transport", [RFC 3926](#), October 2004.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", [RFC 4251](#), January 2006.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant

Networking Architecture", [RFC 4838](#), April 2007.

- [SB07] Scott, K. and S. Burleigh, "Bundle Protocol Specification", [draft-irtf-dtnrg-bundle-spec-10](#) (work in progress), April 2007.
- [haggle] Scott, J., Hui, P., Crowcroft, J., and C. Diot, "Haggle: A Networking Architecture Designed Around Mobile Users", IFIP WONS, Les Menuires, France, January 2006.

Internet-Draft

DTN Bundle Agent Discovery

November 2007

Authors' Addresses

Jim Wyllie
Ohio University
EECS Department #18
Stocker Center
Athens, OH 45701

Phone: +1-740-593-1562
Email: jw280601@ohiou.edu

Wesley M. Eddy
Verizon Federal Network Systems
NASA Glenn Research Center
21000 Brookpark Rd, MS 54-5
Cleveland, OH 44135

Phone: 216-433-6682
Email: weddy@grc.nasa.gov

Joseph Ishac
NASA Glenn Research Center
21000 Brookpark Road
Cleveland, Ohio 44135
USA

Phone: +1-216-433-6587
Fax: +1-216-433-8705
Email: jishac@nasa.gov

Will Ivancic
NASA Glenn Research Center
21000 Brookpark Road
Cleveland, Ohio 44135
USA

Phone: +1-216-433-3494
Fax: +1-216-433-8705
Email: William.D.Ivancic@nasa.gov

Internet-Draft

DTN Bundle Agent Discovery

November 2007

Shawn Ostermann
Ohio University
Department of Computer Science
322b Stocker Center
Athens, Ohio 45701

Phone: +1-740-593-1234
Email: ostermann@cs.ohiou.edu

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).