Workgroup: Delay-Tolerant Networking
Internet-Draft: draft-johnson-dtn-ipnd-00
Published: 7 October 2023
Intended Status: Standards Track
Expires: 9 April 2024
Authors: D. Ellard R. Altman A. Gladd
Raytheon BBN Visionist, Inc.
D. Brown R. in 't Velt S. Johnson
Crowdstrike TNO Spacely Packets, LLC

DTN IP Neighbor Discovery (IPND)

Abstract

Delay and Disruption Tolerant Networking (DTN) IP Neighbor Discovery (IPND), is a method for otherwise oblivious nodes to learn of the existence, availability, and addresses of other DTN participants. IPND both sends and listens for small IP UDP announcement “beacons.” Beacon messages are addressed to an IP unicast, multicast, or broadcast destination to discover specified or unspecified remote neighbors, or unspecified local neighbors in the topology, e.g. within wireless range. IPND beacons advertise neighbor availability by including the DTN node’s canonical endpoint identifier. IPND beacons optionally include service availability and parameters. In this way, neighbor discovery and service discovery may be coupled or decoupled as required. Once discovered, new neighbor pairs use advertised availabilities to connect, exchange routing information, etc. This document describes DTN IPND.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 9 April 2024.

Copyright Notice

Table of Contents

1. Introduction

Delay and Disruption Tolerant Networks (DTNs)[RFC4838] make no presumptions about network topology, routing, or availability. DTNs therefore attempt to provide communication in challenged environments where, for instance, contemporaneous end-to-end paths do not exist. Examples of such DTNs arise in a variety of contexts including mobile social networks, space communications, rural message delivery, military networks, etc.

In many DTN scenarios, the identity and meeting schedule of participating nodes is not known in advance. Therefore, an important primitive is Neighbor Discovery (ND), or the ability to dynamically discover other DTN nodes. This document specifies Internet Protocol Neighbor Discovery (IPND). In contrast to link or physical layer discovery, IPND enables a general form of neighbor discovery across a heterogeneous range of links, as are often found in DTN networks. IPND is particularly useful in mobile, ad hoc DTN environments where meeting opportunities are not known a priori and connections may appear or disappear without warning. For example, two mobile nodes might come into radio distance of each other, discover the new connection, and move data along that connection before physically disconnecting.

In addition to discovering neighbors, it is often valuable to simultaneously discover services available from that neighbor. Examples of DTN services include a neighbor's available Convergence Layer Adapters (CLAs) and their parameters (e.g. TCP CLA [RFC9174]), available routers (e.g. PRoPHET [RFC6693]), tunnels, etc. Newly discovered nodes will then typically participate in bundle [RFC9171] routing and delivery.

In other situations it is useful to decouple service discovery from neighbor discovery for efficiency and generality. For example, upon discovering a neighbor, a DTN node might initiate a separate negotiation process to establish 1-hop connectivity via a particular convergence layer, perform routing setup, exchange availability information, etc.

IPND beacons thus optionally advertise a node's available services while maintaining the ability to decouple node and service discovery as necessary. This flexibility is important to various DTN use scenarios where connection opportunities may be limited (thus necessitating an atomic message for all availability information), bandwidth might be scarce (thus implying that service discovery should be an independent negotiation to lower beacon overhead), or connections have very large round-trip-times (service negotiation is therefore too costly with respect to time).

DTN IPND is designed to be simple, efficient, and general.

Although this document describes a neighbor discovery protocol in terms of IP, the principles and basic mechanisms used in this protocol may also be expressed in terms of other datagram protocols.

The remainder of this document describes DTN IPND.

1.1. Terminology

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

The following terminology is used for describing DTN IPND.

Bundle A PDU as defined in [RFC9171].

Node A DTN entity in the network that receives and processes bundles.

Beacon message An IPND-specific message, defined in this document, used to announce the presence of a DTN node and parameters with which to connect to that node.

Convergence layer adapter A convergence layer adapter (CLA) sends and receives bundles on behalf of a node by providing the conversion between bundles and a transport protocol such as TCP or UDP.

2. Protocol Description

Nodes use DTN IPND beacons, small UDP messages in the IP underlay, to advertise presence. Similarly, IPND beacons received from other nodes serve to detect the availability of DTN neighbors. Nodes SHOULD both send and receive beacons. When the IP underlay is based on the the IPv4 protocol, these beacon messages, detailed in Section 2.6, may be sent as UDP datagrams in either unicast, multicast, or broadcast packets. When the IP underlay uses IPv6, the beacon messages may be sent as either unicast or multicast packets. The beacon message content is agnostic to the underlying transport mode.

Broadcast beacons are designed to reach unknown neighbors in neighborhoods within the local network broadcast domain. IPv4 multicast [RFC1112] or IPv6 multicast [RFC4291] beacons extend the scope of beacon dissemination to potentially include multiple networks across routed boundaries. On broadcast media such as Ethernet or wireless, multicast and broadcast beacons are sent as link-layer broadcast messages.

Broadcast and multicast discovery are described in Section 2.2. In contrast, unicast beacons are sent only to explicitly known and enumerated neighbors as described in Section 2.3.

Upon discovering a neighbor and its services, IPND can establish a connection to the new neighbor via an IP-based Convergence Layer

Adapter (CLA), for example the TCP [RFC9174] or Datagram [RFC7122] CLA. The CLA then negotiates the connection per its individual specification and installs the appropriate next-hop routing information in the local node.

2.1. Beacon Period

An IPND node SHOULD send beacons periodically. The time interval between beacons SHOULD be appropriate for the conditions of the network and MAY be configurable.

An IPND node MAY make use of the OPTIONAL Beacon Period field in the beacon message to explicitly inform neighbors of the interval on which to expect future beacons. The Beacon Period is not fixed for a given sender and MAY change with each beacon message. If the Beacon Period is included and set to zero, then it SHALL be interpreted as negating any expectation for future beacons.

A receiving node SHOULD either know the expected beacon interval of neighbors or extract the interval from the Beacon Period field of arriving messages. The beacon interval along with the existence and receive time of beacons SHOULD be used to determine the state of the sender’s ability to transmit to the receiver (i.e. the up or down state of the sender-to-receiver link). The exact algorithm for determining the link status based on received beacons is implementation-defined.

2.2. Unknown Neighbors

In the general case, the IP addresses of potential neighbors are not known in advance. To discover unknown neighbors, IPND beacon messages are sent as IP packets with either multicast or broadcast destination addresses. An IPND node MUST support multicast IP destination addresses [RFC1112] [RFC4291] and multicast IGMP / MLD group membership [RFC3376] [RFC2710] [RFC3810]. A node MAY support IP broadcast destinations. IPv4 multicast addresses for IPND SHOULD be from the IANA assigned local network control block 224.0.0/24 [RFC5771]. This block of multicast addresses is intentionally scoped to the local network to prevent dissemination to the wider Internet. Likewise, IPv6 multicast addresses for IPND should have link-local scope [RFC4291] [RFC7346].

An IPND node MAY also use other multicast addresses as required, such as IPv4 multicast addresses from the IANA assigned Internetwork Control Block [RFC5771] or IPv6 multicast addresses with wider scope than link-local. One use case for this would be a mobile ad hoc network (MANET) environment which includes nodes that are not DTN-capable, but do support IP multicast forwarding, e.g. by means of

SMF [RFC6621]. Those nodes that are DTN-capable would then be able to discover each other over multiple IP hops.

In all multicast addressing cases, a node MUST support a configurable IPv4 time-to-live value or IPv6 Hop Limit value for all beacon messages.

2.3. Enumerated Neighbors

An IPND node SHOULD support unicast beacons. Since multicast or broadcast discovery may not always be feasible over internetworks, the IP addresses of potential neighbors reachable only across multiple underlay hops must be explicitly enumerated for discovery. While the neighbor's address is therefore known, the availability of that neighbor is not known. IPND thus permits DTN nodes to discover available remote neighbors across multiple IP underlay hops when provided with the addresses of those neighbors. In this way, IPND can be used to bridge IP-based DTNs while detecting disconnections among and between the DTNs.

2.4. Allowing Data to Substitute for Beacons

Sending data to an IP address matching a configured beacon destination SHOULD suppress the generation of beacon messages to that destination for a period of time up to but no longer than the beacon sending interval. This suppression SHOULD NOT occur if the parameters of a new beacon message would differ from the preceding beacon including the advertised services (Section 2.6.3) or the Neighborhood Bloom Filter (NBF) (Section 2.6.4).

Upon receiving a data packet from a neighbor where the packets do not represent a beacon, a node SHOULD behave as if a beacon had been received from that neighbor, as follows. If the data packet is addressed to this node via a unicast address, then the behavior SHOULD be as if the implied received beacon contains a Neighborhood Bloom Filter advertisement which indicates the membership of the receiving node in the sender's 1-hop neighborhood. Otherwise, if the destination address is multicast or broadcast, then the receiving node should presume that the link is bidirectional if and only if its state was bidirectional prior to receiving the data packet (Section 2.5). The sender's advertised services and beacon period are presumed to be unchanged since the sender's last received beacon. If no beacons have previously been received from such a neighbor, then it is presumed that there are no services associated with the sender.

2.5. Discovering Bidirectional Links

Many routing protocols work correctly only when links are bi-directional. In wired IP networks, link bi-directionality can often

be presumed. For other types of networks, such as Mobile Ad Hoc Networks (MANETs) this assumption often does not hold. If a link to a neighbor is said to be “up” only because one or more beacon messages have been received from that neighbor over a wireless medium, it is not generally safe to assume that the link is bidirectional. In practice, MANETs often have links that are only unidirectional due to differences in antennae, transmit power, hardware variability, multi-path effects, etc.

To discover the bi-directionality of links, an IPND Neighborhood Bloom Filter (NBF) (Section 2.6.4) facility MAY be employed in which each node advertises a Bloom filter representation of the set of neighbors from whom it has received enough recent beacons to be considered “up”. Upon receiving a beacon from an “up” neighbor that advertises an NBF which represents a set containing the receiving node’s ID, the link SHOULD then be considered bi-directional.

2.6. Beacon Message Format

Figure 1 depicts the format of beacon messages. Note that IPND follows the DTN convention of using Self-Delimiting Numeric Values (SDNVs) [RFC6256] to represent variable length integers. An IPND node MUST use UDP checksums to ensure correctness.

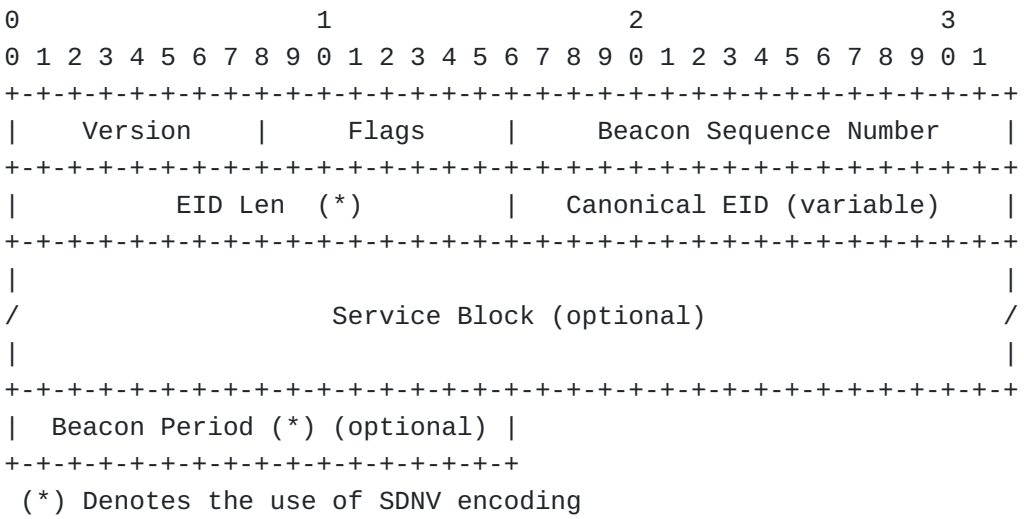


Figure 1: Beacon Message Format

The beacon message is comprised of the following fields:

*Version: An 8-bit field indicating the version of IPND that constructed this beacon. The present document describes version 0x04 of IPND. This version field is incremented for IPND if either the IPND protocol is modified or the Bundle Protocol version is incremented. In this way the field can also be used to determine the BP version supported by a potential DTN neighbor.

*Flags: An 8-bit field indicating IPND processing flags. Four flags are currently defined. 0x00 indicates that no special processing should be performed on the beacon. If more than one of the Flags bits is set, then the associated structures will appear in the beacon message according to their bit order (Bit 0 is first). Semantics of bits are described here from least significant (LSb) to most significant (MSb).

Bit 0 Source EID present: iff set, indicates that the source node's EID is present in the beacon. If the EID is present, it is preceded by an SDNV indicating its length. An IPND node SHOULD include its EID in all beacons, therefore this flag SHOULD always be set.

Bit 1 Service Block present: iff set, indicates that a service block is present.

Bit 2 Neighborhood Bloom Filter present: iff set, indicates that a Neighbor Bloom Filter is present within the Service Block.

Bit 3 Beacon Period present: iff set, indicates that a Beacon Period field is present.

Bits 4-7 Reserved.

*Beacon sequence number: A two-octet unsigned integer value incremented once for each beacon transmitted to a particular IP address.

*EID Length: The byte length of the canonical EID contained in the beacon. The EID length field is an SDNV and is therefore variable length. A two-octet length is shown for convenience of representation.

*Canonical EID: The canonical end node identifier of the neighbor advertised by the beacon message. The canonical EID is variable length and represented as a Uniform Resource Identifier [RFC3986].

*Service Block: Optional announced services (Section 2.6.1) in the beacon. Services MAY include CLAs (Section 2.6.3), routing parameters, a Neighborhood Bloom Filter (Section 2.6.4), and other implementation-dependent services.

*Beacon Period: Optional field indicating the sender’s current beacon interval in seconds. A value of zero indicates that the beacon period is undefined. The Beacon Period is an SDNV and is

therefore variable length. A two-octet length is shown for convenience of representation.

2.6.1. Service Block

As described previously, beacon messages may optionally include a block of service availability information. The service block is intended to contain representations of available CLAs, routers, a Neighborhood Bloom Filter, etc., but is sufficiently general to accommodate implementation-specific services provided by the advertising node.

For example, the source IP address of a received beacon suffices to identify the remote node at the IP level. However, the IP address alone does not inform other processes via which transport mechanism (e.g. TCP or UDP) or via which transport port the remote node is offering a connection. Similarly, nodes do not know which routers (e.g. PRoPHET [RFC6693]) are running on a remote node in order to inform bundle exchange. Therefore, a beacon MAY contain a service block which serves to notify nodes about the availability of these services.

The format of a service block is given in Figure 2.

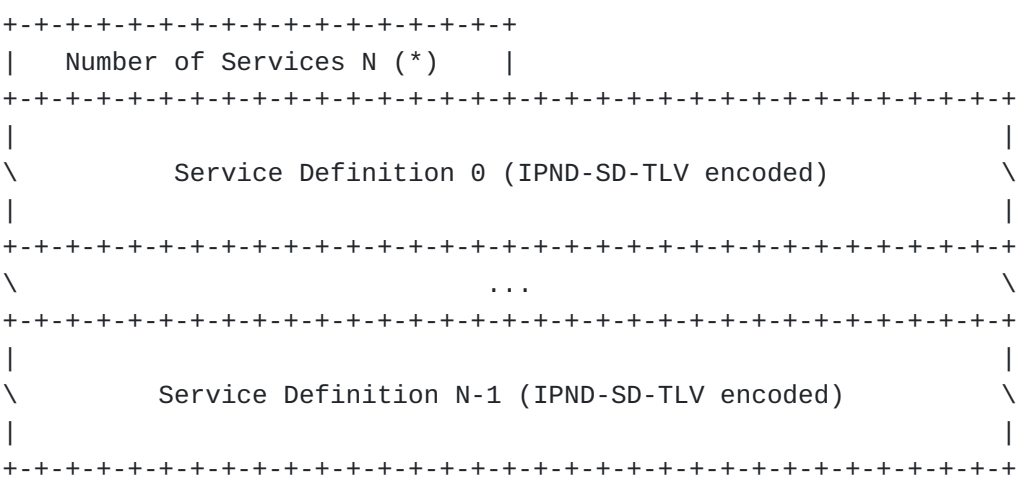


Figure 2: Service Block Format

A service block is comprised of the following fields:

* Number of services: The number of services described in the block. The number of services is an SDNV and is therefore variable length.

*Service Definition(s): A list of service definitions encoded according to the IPND Service Definition TLV encoding (Section 2.6.2) specification.

2.6.2. IPND Service Definition TLV Encoding

The IPND Service Definition TLV encoding scheme (IPND-SD-TLV) provides for the standardization of service definitions using a format that focuses on simplicity, flexibility, and efficiency. IPND-SD-TLV borrows many ideas from from the ASN.1 Basic Encoding Rules (BER) specification [ASN1-BER]. Like ASN.1 BER, IPND-SD-TLV structures are generally composed of three distinct parts:

1. Tag: A numeric token which identifies the structure (REQUIRED).

2. Length: A numeric value which specifies the size of the content block (sometimes REQUIRED).

3. Value: The content block, which contains the value(s) described by the tag (REQUIRED).

IPND-SD-TLV tags SHALL be 8-bit values, providing IPND a range of 256 possible tag numbers. Tag assignments are designed to provide a basic, standard set of building blocks while remaining flexible enough to allow the implementation of unforeseen specifications. The first 128 tag numbers (i.e. 0-127) SHALL be reserved for standard definitions; the remaining tags (i.e. 128-255) MAY be used for implementation-specific (private) definitions. This design allows a node to inspect the most significant bit (bit-7, zero-indexed) of the tag to determine whether it is a reserved or private value.

IPND-SD-TLV defines two classes of data types: Primitive and Constructed (the difference between these data types is discussed below). Reserved tag numbers are designed such that the class of a data type can be determined by examining the second-most significant bit (bit-6, zero-indexed) of the tag. If this bit is not set, the data type is primitive, otherwise it is constructed. As a result of this design, reserved primitive types SHALL be assigned tag numbers 0-63, while reserved constructed types SHALL be assigned tag numbers 64-127.

Private tag numbers are always expected to represent constructed data types, therefore private (implementation-specific) constructed types (if in use by IPND) SHALL be assigned tag numbers 128-255.

The construction of IPND-SD-TLV tags is depicted in Figure 3

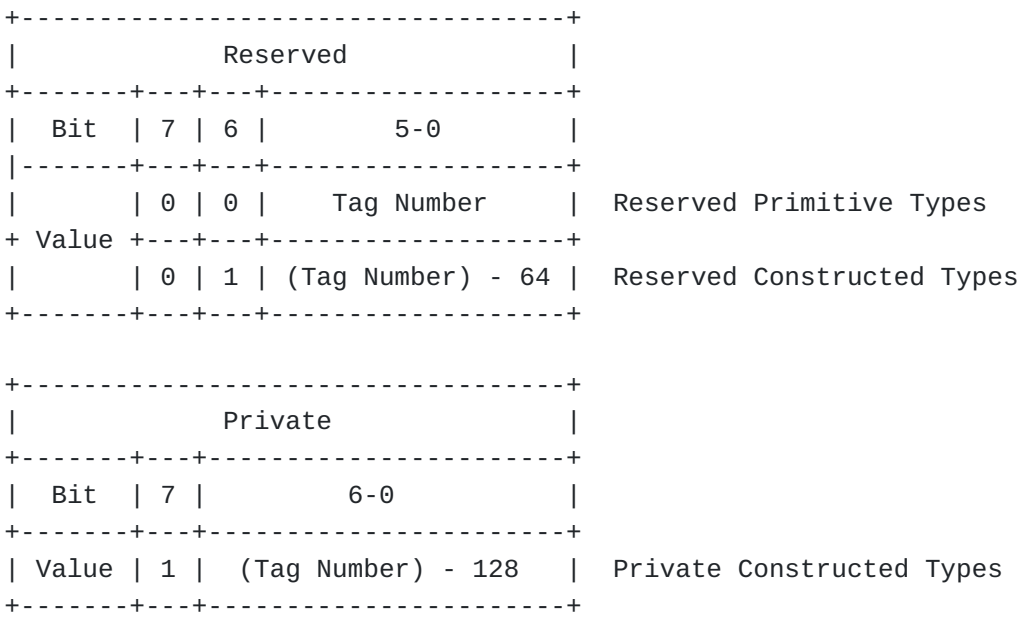


Figure 3: IPND-SD-TLV Tags

In order to keep encoded services simple and compact, IPND-SD-TLV SHALL omit the length field in cases where the content's length is always fixed (e.g. an IP address) or described in-place (e.g. an SDNV value). In the cases where an explicit length field is required (e.g. string content), an IPND node SHALL SDNV encode the length values. Additionally, a length field MUST be included in constructed types immediately following the tag value which describes the length, in bytes, of the structure's content block. This constraint allows a node to skip constructed types that are unrecognized while reading a received Service Block. An IPND node SHALL SDNV encode these length values.

Again, IPND-SD-TLV defines two classes of data types: Primitive and Constructed. Primitive types represent fundamental data types such as integers or strings. An IPND node MUST support the primitive data types specified in Figure 4. Note that primitive types use one of three distinct length specifiers:

*Fixed: The content always has a fixed length and SHALL NOT include a length field. Fixed length numeric values (including floating point numbers) SHALL be written in network byte order.

*Variable: The content is variable length but is encoded as an SDNV, therefore it SHALL NOT include a length field.

*Explicit: The content is variable and does not describe its own length, therefore it MUST include a length field immediately following the tag value.

TAG #	Definition	Length Type	Content Length (unencoded bytes)
0	boolean	Fixed	1
1	uint64	Variable	1-8*
2	sint64	Variable	1-8*
3	fixed16	Fixed	2
4	fixed32	Fixed	4
5	fixed64	Fixed	8
6	float	Fixed	4
7	double	Fixed	8
8	string	Explicit	1-N
9	bytes	Explicit	1-N
10-63	UNASSIGNED		

*Denotes content that is SDNV encoded

Figure 4: IPND-SD-TLV Primitive Types

Note that a special case exists for representing the empty string and the empty byte array for the “string” and “bytes” data types, respectively. In both cases, “empty” is represented by an explicit length value of 1 and content of a single null byte.

Constructed data types represent structures that are composed of other data types. As described earlier, reserved constructed types SHALL be assigned tag numbers 64-127. Additionally, nodes MAY assign tag numbers 128-255 to private constructed types in order to allow for implementation-specific constructed types. An IPND node SHALL use constructed types to specify service definitions as described in Section 2.6.3.

It is important to note that the order in which other types are composed within a constructed type need not be explicitly stated. Ordering only becomes an issue in the case where a constructed type (not representing an array structure) contains multiple instances of the same data type. In order to defeat this issue, implementations MUST create data type wrappers in order to differentiate identical types. This design allows IPND to be order-agnostic when it comes to reading data types that compose a constructed type. Appendix B describes an example where data type wrappers are used to differentiate identical fundamental types.

2.6.3. Services

A service is an IPND-SD-TLV structure that represents an advertisement for a DTN-related resource available on the beacon source node. Each service type SHALL have a unique tag number in

order to identify it within the service block. Nodes SHALL use the initial set of tag assignments described in Figure 5 (the rationale for tag numbering is described in Section 2.6.2).

TAG #	Definition	Construction
64	CLA-TCP-v4	{IP (fixed32), Port (fixed16)}
65	CLA-UDP-v4	{IP (fixed32), Port (fixed16)}
66	CLA-TCP-v6	{IP (bytes), Port (fixed16)}
67	CLA-UDP-v6	{IP (bytes), Port (fixed16)}
68	CLA-TCP-HN	{Hostname (string), Port (fixed16)}
69	CLA-UDP-HN	{Hostname (string), Port (fixed16)}
70	CLA-DCCP-v4	{IP (fixed32), Port (fixed16), Servicecode (fixed32)}
71	CLA-DCCP-v6	{IP (bytes), Port (fixed16), Servicecode (fixed32)}
72	CLA-DCCP-HN	{Hostname (string), Port (fixed16), Servicecode (fixed32)}
73-125	UNASSIGNED	
126	NBF-Hashes	Hash IDs (bytes)
127	NBF-Bits	Bit Array (bytes)
128-255	PRIVATE USE	

Figure 5: IPND-SD-TLV Constructed Services

An IPND node MUST support the service definitions for CLA-TCP-v6 and CLA-UDP-v6; that is, a node MUST support the standard definitions for TCP CLA advertisements and UDP CLA advertisements, respectively (both supporting IPv6 128-bit addresses). An example bitwise representation of the CLA-TCP-v6 service is depicted in {{figure6}. Note that the format of the CLA-UDP-v6 service is identical except for the initial tag number, which would instead be 67 (hex 0x43).

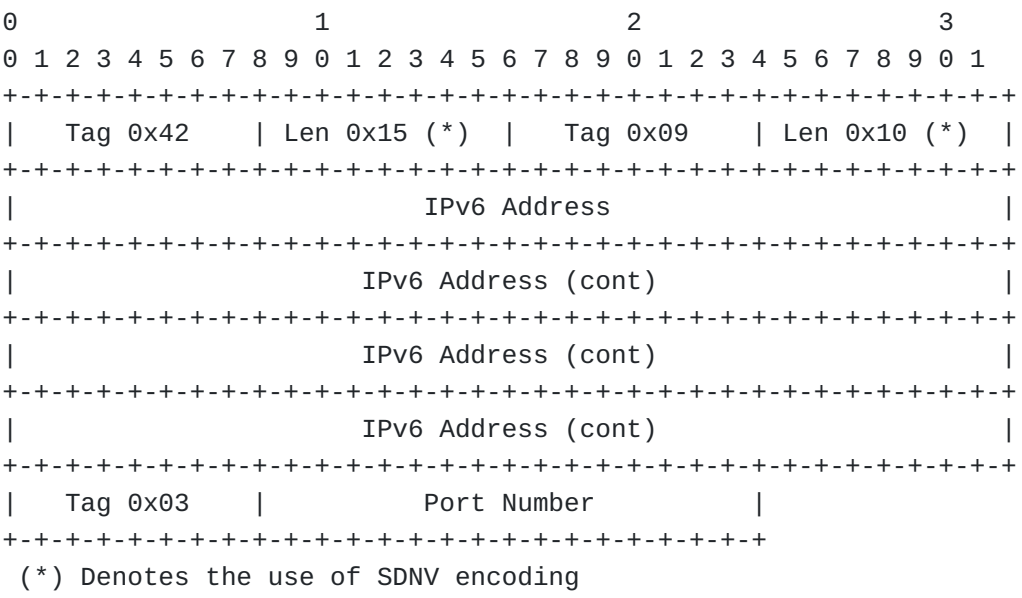


Figure 6: CLA-TCP-v6 Service Format

An IPND node MAY support the CLA-TCP-v4, CLA-UDP-v4, CLA-TCP-HN, CLA-UDP-HN, CLA-DCCP-v4, CLA-DCCP-v6 and CLA-DCCP-HN service definitions. Bitwise representations of the CLA-UDP-v4, CLA-TCP-HN and CLA-DCCP-v6 services are depicted in Appendix A. Additionally, a node MAY support the Neighborhood Bloom Filter services (NBF-Hashes and NBF-Bits). These services are described below (Section 2.6.4). Lastly, a node MAY support any implementation-specific services with tag numbers 128-255. Appendix B describes an example of an implementation-specific service that makes use of private tag number assignments.

2.6.4. Neighborhood Bloom Filter

In order to efficiently determine link bi-directionality, a node represents the set of its 1-hop neighbors using a Bloom filter referred to as the Neighborhood Bloom Filter (NBF). Upon receiving a beacon from a neighbor that contains NBF service information, a node can quickly determine whether it is in the neighbor’s NBF set, and thereby determine whether the link is bidirectional.

Every node that might operate in an environment where discovered links may not be bidirectional SHOULD include NBF service advertisements in its multicast or broadcast beacons which describe the membership of its 1-hop neighbor set. This is especially true if a node’s routing protocol presumes that links are bidirectional.

An NBF need not be included within every beacon, but one SHOULD be present within at least one broadcast or multicast beacon following a change in the 1-hop neighborhood of the node. An NBF advertisement MAY be present in every broadcast or multicast beacon.

In order to advertise an NBF, an IPND node MUST include two distinct services in the Service Block of some (or all) of its beacons: NBF-Hashes, which describes the hash algorithms used to compute the NBF bit array; and NBF-Bits, which contains the actual bit array of the NBF. The bits set in the NBF-Bits structure MUST be defined by computing hashes on the canonical EID of each 1-hop neighbor considered to be “up”. Each hash algorithm used to compute the NBF bit array MUST be identified in the NBF-Hashes structure (using numerical identifiers; one byte per identifier). Exemplary bitwise formats of fictional NBF-Hashes and NBF-Bits structures are depicted in Figure 7 and Figure 8, respectively. Note that the NBF bit array in the NBF-Bits structure must be byte-aligned, and SHALL be padded with zero bits at the end of the bit array to achieve byte-alignment.

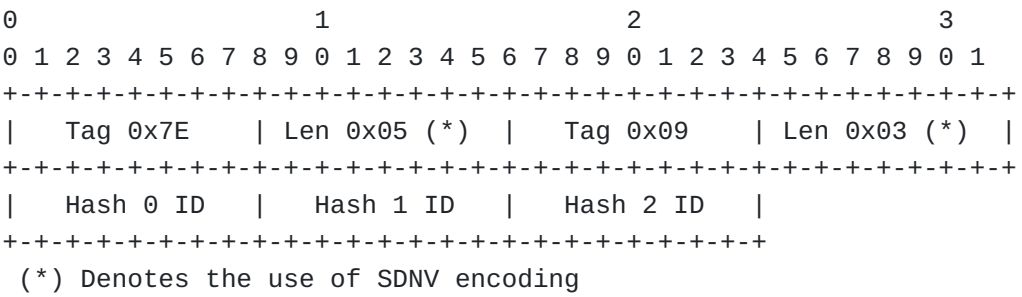


Figure 7: Fictional NBF-Hashes Service Format

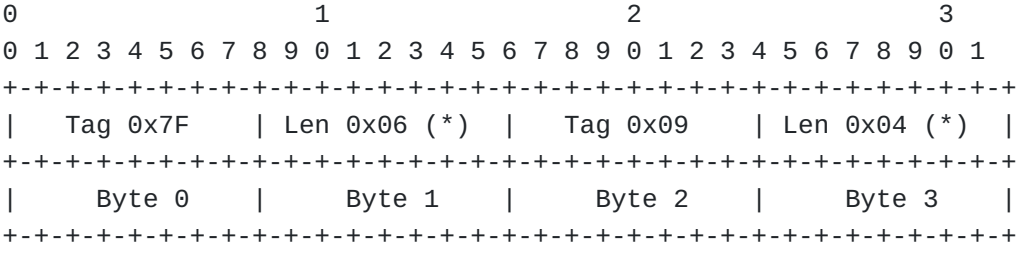


Figure 8: Fictional NBF-Bits Service Format

Different networks naturally have distinct requirements, tolerance for overhead, and node computing resources, so the parameters of the Bloom Filter such as the bit array length, and the number and types of hash algorithms, are not mandated by IPND. However, all nodes participating in such a DTN SHOULD be aware of the same set of hash algorithms and their respective identifiers used in NBF-Hashes structures.

NBF services, if present, MAY be ignored by a receiving IPND node if its implementation does not provide for it, or if the parameters of

the Bloom filter cannot be determined with certainty (e.g. if the hash function identifiers are not recognized).

2.7. IPND and CLAs

IP-based CLAs are generally expected to depend on an IPND implementation module for their discovery service. A CLA MAY opt not to use IPND, either because that CLA does not require discovery or provides its own.

Once IPND discovers a new neighbor it MUST inform all CLAs which depend on IPND of the neighbor's existence and the discovered parameters. The exact means by which IPND communicates with the CLAs is implementation dependent.

Similarly, once IPND determines that a link has gone down, it MUST inform all dependent CLAs of the link down event.

2.8. Disconnection

Note that an IPND node SHOULD maintain state over all existing neighbors in order to prevent CLAs from needlessly attempting to establish connections between nodes that are already connected. To maintain the current neighbor set, IPND removes stale neighbors after the defined neighbor receive timeout period elapses without receiving any beacon messages from a particular neighbor.

Upon detecting a neighbor that is no longer available, IPND MAY provide hints to the CLAs that the neighbor is gone. Note that some CLAs themselves provide keepalive-type functionality and therefore IPND is not necessarily required to detect down neighbors. However, relying on IPND to provide both discovery and availability information provides a single, coherent point in the system design to maintain neighbor information.

3. Relation to Other Discovery Protocols

A variety of discovery protocols exist in other contexts and domains. These discovery protocols include the ability to discover available neighbors and services. For example, the IETF zero configuration working group [RFC3927], the Bonjour protocol [BONJOUR], and the OLSRv2 neighborhood discovery protocol(NHDP) [RFC6130] all provide similar functionality.

Other rendezvous mechanisms are possible that allow a node to find a neighbor of a particular type or with particular properties. For example, the Domain Name System (DNS) or Distributed Hash Tables (DHTs) could be used to find a neighbor that provides an inter-planetary gateway. Such advanced rendezvous schemes are beyond the scope of this document.

In contrast, DTN-IPND is designed to be DTN-specific, efficient, and extremely lightweight. For instance, DTN-IPND is capable of supporting arbitrary length DTN EIDs, and may include CLA information in order to maximize the utility of each beacon message without requiring multiple round-trip transmissions in order to perform complex protocol negotiation.

While DTN-IPND MAY be used in non-DTN environments, its use is RECOMMENDED only in DTNs.

4. Implementation Experience

Raytheon BBN Technologies (BBN) developed an implementation of DTN IPND which has been added to the bundle protocol reference implementation, DTN2, as an experimental build option.

BBN has also implemented and deployed an earlier version of DTN IPND as part of the [SPINDLE] project.

An earlier version of this specification has also been implemented as part of the [IBR-DTN] project at Technical University of Braunschweig, Germany.

5. Security Considerations

Neighbor discovery may be perceived as an impediment to security because it advertises a potential target for attacks. Discovering the existence of a particular node is orthogonal to securing the services of that node. Nodes that desire or require higher-levels of security SHOULD disable the broadcast IPND beacons and rely instead on static neighbor configuration.

Further, neighbor discovery represents a potential source of network congestion and contention. Therefore, careful consideration should be made to the frequency and TTL / Hop Limit scope of beacons when setting implementation-specific parameters, particularly when a setting affects larger regions of the network.

6. IANA Considerations

6.1. Port Number

Port number is requested to be assigned by IANA as the default UDP port for IPND, prior to publication if this draft is approved for publication as an RFC.

Service Name: dtn-ipnd

Transport Protocol(s): UDP

Assignee: Scott Johnson (scott@spacelypackets.com)

Contact: Scott Johnson (scott@spacelypackets.com)

Description: DTN IP Neighbor Discovery Protocol

Reference: (This document)

Port Number: TBD

6.2. Tag numbers

A new IANA registry should be created to document the standard tag number assignments for IPND Service Definition TLV structures. The registry shall define a single numberspace with values representing the IPND-SD-TLV tag numbers as described in Section 2.6.2.

The registration policy for this new registry shall be:

0-63: Expert Review. Specifications in this subset must only be for primitive datatypes, and the specification must describe which “length type” will be used for the new datatype as well as the unencoded content length (see Figure 4). New registrations shall only be approved for datatypes reasonably expected to have a use case applicable throughout the community.

64-127: Expert Review. Specifications in this subset must only be for constructed datatypes, and the specification must describe the composition of the new datatype using references to existing datatypes (as in Figure 5). New registrations shall only be approved for datatypes reasonably expected to have a use case applicable throughout the community.

128-255: Private or Experimental use. No assignment by IANA.

The value range is: unsigned 8-bit integer.

Value	Description	Reference
0	Primitive boolean tag	This Document
1	Primitive uint64 tag	This Document
2	Primitive sint64 tag	This Document
3	Primitive fixed16 tag	This Document
4	Primitive fixed32 tag	This Document
5	Primitive fixed64 tag	This Document
6	Primitive float tag	This Document
7	Primitive double tag	This Document
8	Primitive string tag	This Document
9	Primitive byte array tag	This Document
10-63	Unassigned (primitive only)	
64	CLA-TCP-v4 service tag	This Document
65	CLA-UDP-v4 service tag	This Document
66	CLA-TCP-v6 service tag	This Document
67	CLA-UDP-v6 service tag	This Document
68	CLA-TCP-HN service tag	This Document
69	CLA-UDP-HN service tag	This Document
70	CLA-DCCP-v4 service tag	This Document
71	CLA-DCCP-v6 service tag	This Document
72	CLA-DCCP-HN service tag	This Document
73-125	Unassigned (constructed only)	
126	NBF-Hashes service tag	This Document
127	NBF-Bits service tag	This Document
128-255	Private/Experimental Use	

Figure 9: IANA IPND-SD-TLV Tag Number Assignments

7. References

7.1. Normative References

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <http://www.rfc-editor.org/info/rfc1112>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <http://www.rfc-editor.org/info/rfc2710>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <http://www.rfc-editor.org/info/rfc3376>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <http://www.rfc-editor.org/info/rfc3810>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC7346] Droms, R., "IPv6 Multicast Address Scopes", RFC 7346, DOI 10.17487/RFC7346, August 2014, <http://www.rfc-editor.org/info/rfc7346>.

[RFC9171] Scott, K. and S. Burleigh, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <http://www.rfc-editor.org/info/rfc9171>.

[RFC9174] Sipos, B., Demmer, M., Ott, J., and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol Version 4", RFC 9174, DOI 10.17487/RFC9174, January 2022, <http://www.rfc-editor.org/info/rfc9174>.

7.2. Informative References

[ASN1-BER] "ITU-T Rec. X.690, Abstract Syntax Notation One (ASN.1), Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ISO/IEC 8825-1:2002", 2002.

[BONJOUR] Cheshire, S., "Bonjour", April 2005.

[IBR-DTN] Schildt, S., Morgenroth, J., Poettner, W-B., and L. Wolf, "IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation", in Electronic Communicatios of the EASST, Vol. 37, pages 1-11, January 2011.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927,

DOI 10.17487/RFC3927, May 2005, <http://www.rfc-editor.org/info/rfc3927>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <http://www.rfc-editor.org/info/rfc4838>.

[RFC5771] Cotton, M., Vegoda, L., and D. Meyer, "IANA Guidelines for IPv4 Multicast Address Assignments", BCP 51, RFC 5771, DOI 10.17487/RFC5771, March 2010, <http://www.rfc-editor.org/info/rfc5771>.

[RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, DOI 10.17487/RFC6130, April 2011, <http://www.rfc-editor.org/info/rfc6130>.

[RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", RFC 6256, DOI 10.17487/RFC6256, May 2011, <http://www.rfc-editor.org/info/rfc6256>.

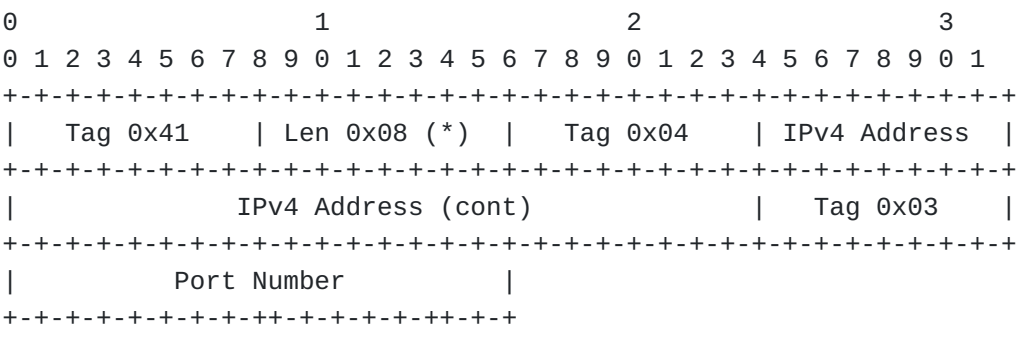
[RFC6621] Macker, J., Ed., "Simplified Multicast Forwarding", RFC 6621, DOI 10.17487/RFC6621, May 2012, <http://www.rfc-editor.org/info/rfc6621>.

[RFC6693] Lindgren, A., Doria, A., Davies, E., and S. Grasic, "Probabilistic Routing Protocol for Intermittently Connected Networks", RFC 6693, DOI 10.17487/RFC6693, August 2012, <http://www.rfc-editor.org/info/rfc6693>.

[RFC7122] Kruse, H., Jero, S., and S. Ostermann, "Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP)", RFC 7122, DOI 10.17487/RFC7122, March 2014, <http://www.rfc-editor.org/info/rfc7122>.

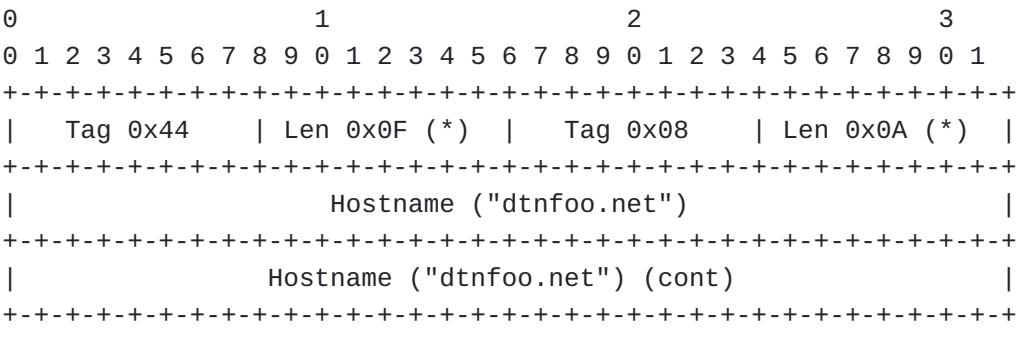
[SPINDLE] Krishnan, R., "Survivable Policy-Influenced Networking: Disruption-tolerance through Learning and Evolution", October 2006.

Appendix A. Additional Figures



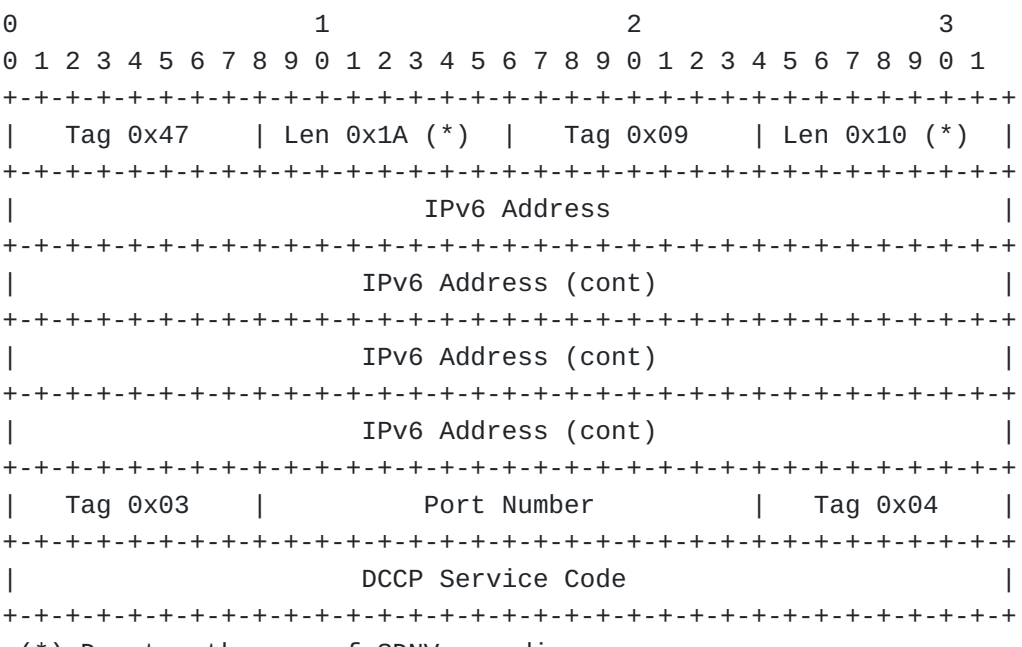
(*) Denotes the use of SDNV encoding

Figure 10: CLA-UDP-v4 Service Format



(*) Denotes the use of SDNV encoding

Figure 11: CLA-TCP-HN Service Format



(*) Denotes the use of SDNV encoding

Figure 12: CLA-DCCP-v6 Service Format

Appendix B. Fictional Private Service Example

The following describes a fictional implementation-specific routing service in order to demonstrate the use of IPND-SD-TLV encoding rules. Figure 13 defines the construction of the service structure using tag numbers out of the private tag assignment space. Note the use of “wrapper” data types in order to differentiate between what would otherwise be identical data types within the composition of the router service’s definition.

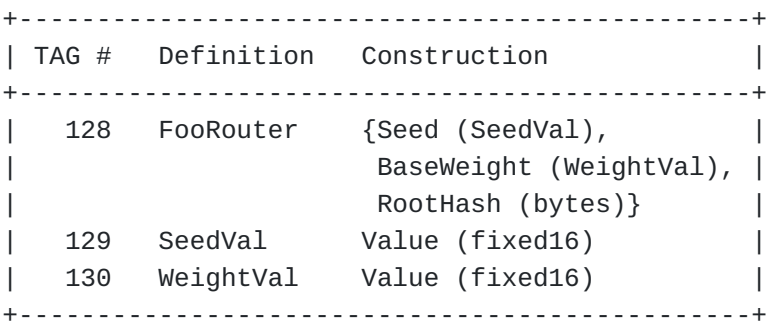


Figure 13: Fictional Router Definition

Figure 14 depicts the bitwise representation of an IPND-SD-TLV encoded FooRouter service using fictional content values. Note that the ordering of the service’s composition does not exactly match the definition; this should not be an issue for a receiving node with knowledge of the FooRouter service.

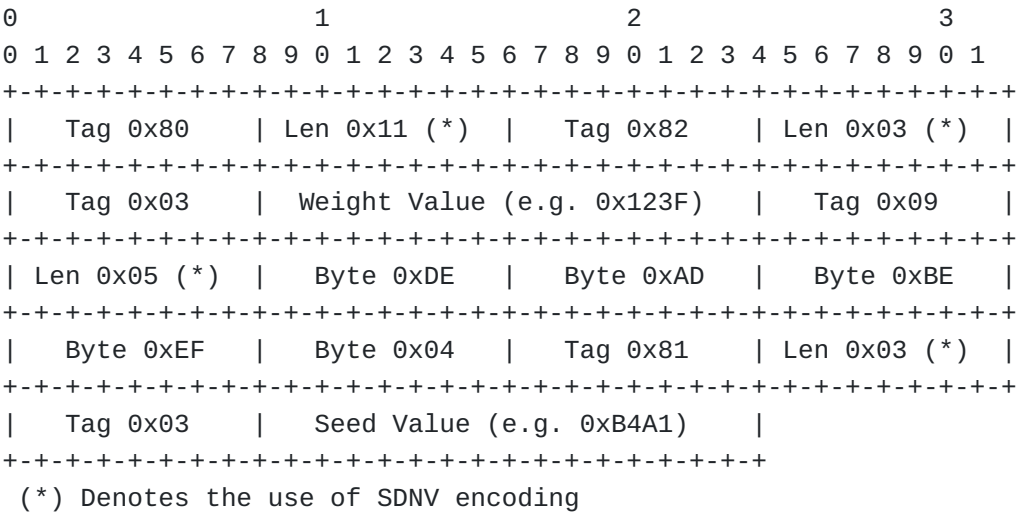


Figure 14: Fictional FooRouter Format

Authors' Addresses

Daniel Ellard
Raytheon BBN
10 Moulton St.
Cambridge, MA 02138
United States of America

Email: dan.ellard@rtx.com

Richard Altmann
Visionist, Inc.
9861 Broken Land Parkway, Suite 400
Columbia, MD 21046
United States of America

Email: raltmann@gmail.com

Alex Gladd

Email: argladd86@gmail.com

Daniel Brown
Crowdstrike

Email: danbrown9725@gmail.com

Ronald in 't Velt
TNO
Anna van Buerenplein 1
Den Haag

Email: Ronald.intVelt@tno.nl

Scott M. Johnson
Spacely Packets, LLC
46 High Ridge Road
Ormond Beach, FL 32117
United States of America

Email: scott@spacelypackets.com