Yoid Distribution Protocol (YDP) Specification

Paul Francis

ACIRI
francis@aciri.org

April 2, 2000

Contents

1 Changes

April 2, 2000 Updated for name change from Yallcast to Yoid.

2 Introduction

This document specifies the Yoid Distribution Protocol (YDP). An overview of YDP and its role in the Yoid architecture can be found in “Yoid: Extending the Internet Multicast Architecture”. This specification assumes the reader is familier with that document.

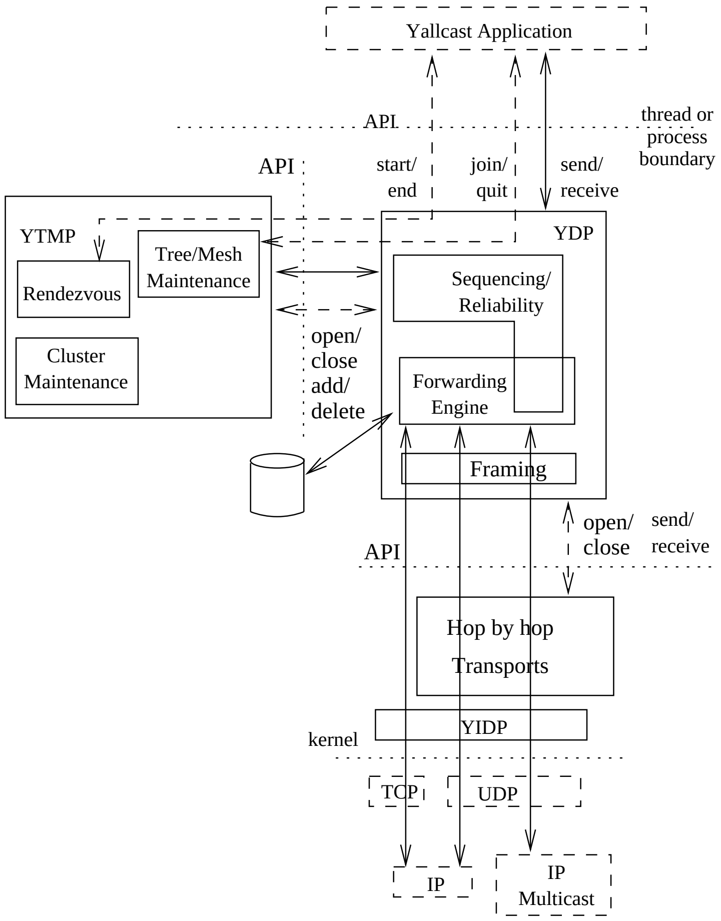

Figure 1: Position of YDP

2.1 Document Status

This spec is very much a snapshot of a work in progress. In particular, we can expect YDP to evolve considerably as we try it under different applications and find what works and what doesn't.

Virtually all of what is in this spec has been implemented, though not yet very thoroughly tested.

3 Position of YDP

The position of YDP, relative to other protocols as it might appear in a typical implementation, is shown in Figure 1. Note that YDP plays a central role. It forwards frames between upper and lower layers. It coordinates extensively with YTMP—it must tell YTMP when connections go down or have been accepted, and YTMP tells it when new tree or mesh neighbors have been established. While this is implementation specific, it is reasonable for the YDP module to be the single point of contact with the application, passing commands between the application and YTMP (create rendevous, join group, etc.) where necessary.

4 YDP Frame Header Format

This section describes the header format for YDP frames. The 20-byte (without options) header of each frame, when transmitted over either UDP or TCP, is formatted as shown below:

```
                   1                   2                   3
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-------+-------+-------+-------+---------------+---------------+
|Version| Deliv | Flags | Trans |   Protocol    | Header Length |
+-------+-------+-------+-------+---------------+---------------+
|          Source Code          |           Dest Code           |
+---------------+---------------+---------------+---------------+
|           Hop Count           |          Data Length          |
+---------------+---------------+---------------+---------------+
|                                                               |
+----                    Sequence Number                    ----+
|                                                               |
+---------------+---------------+---------------+---------------+
|                                                               |
\                         Header Options                        \
|                                                               |
+---------------+---------------+---------------+---------------+
```

In addition to these fields, it is assumed that the ID Code of YIDP, as well as the information associated with it (the Group ID and HxH source and destination names), is made available from the lower layer.

The fields are defined as shown below:

Version: Header version number. Set at 1 for this header.

Deliv: This field determines the delivery mode of the frame, as follows:

Reserved (0): Reserved for later definition.

Multicast (1): If the frame has not already been received and transmitted (as determined by the Frame Source and Sequence Number), transmit frame to all tree neighbors other than the one from which it was received. The IP multicast group of a cluster to which the member belongs is considered, for this purpose, to be a single tree neighbor. That is, a frame received from the IP multicast group is not transmitted back to the IP multicast group. Frames are delivered to transit members, foot transit members, and stub members, depending on the value of the Dest Code field.

If the frame was received from a member that is not a tree neighbor, then the frame is discarded, and an Invalid Neighbor Error Message is sent to the member from which the frame was received.

Note that YDP has no scoping field analogous to that of IP multicast. The intent for now is that scoping can be achieved simply by creating a new group consisting of the desired scope. If this proves inadequate (for instance, because of the cost or delay of creating groups on the fly), the use of a more traditional scope field can be considered.

Tree Anycast (2): This frame is randomly routed either to a tree neighbor other than the one from which it was received, or delivered to the member's upper layer, with the following probabilities: Assume there are N tree neighbors. If the frame was received from the parent, then it is randomly routed to a different tree neighbor or the upper layer all with probability $1/N$. Otherwise, the frame is routed to the parent with probability 1/2, and sent to a child, including the cluster child (taken as a whole), or the upper layer all with probability $1/2(N-1)$.

Only transit members, foot transit members, and stub members are considered as neighbors in the algorithm, depending on the value of the Dest Code field.

Mesh Anycast (3): When this frame is received, the upper 8 bits and lower 8 bits of the Dest Code are treated as distinct fields. The value in the upper 8 bits of the Dest Code field is decremented. If it is decremented to 0, the frame is forwarded to the member's upper layer. Otherwise, it is randomly forwarded to any appropriate neighbor (mesh or tree), other than the one from which it was received. Only transit members, foot transit members, and stub members are considered as neighbors in the algorithm, depending on the value of the lower 8 bits of the Dest Code field.

It can be used in datagram mode only.

- **Unicast (4):** The frame is transmitted to the upper layer of a single group member, as identified by the Dest Code field. In the case of datagram, it is transmitted directly (via IP) to the member, even if the sender is not a neighbor. In the case of stream, it is transmitted neighbor-to-neighbor along the tree to the recipient.
- **Broadcast (5):** If the frame has not already been received and transmitted (as determined by the Frame Source and Sequence Number), transmit it, via stream, to all mesh and tree neighbors other than the one from which it was received. Frames are delivered to transit members, foot transit members, and stub members, depending on the value of the Dest Code field.
- **Reserved (6-15):** Reserved for later definition.

Flags: The following flags are defined:

- **Pushback (0):** Set to 1 when sender of frame can receive frames with application data. Set to 0 otherwise. Used for pushback flow control to prevent loss of frames at members. If the pushback state must be changed when there is no frame to send, then a frame with a Protocol field value of 2 (YDP) is sent with no attached data (the Data Length field is set to 0).
- **Reserved:** The remaining flages are reserved for later definition. They must be transmitted as 0 and ignored upon receipt.

Trans: This field determines the transmission mode of the frame, as follows:

- **Reserved (0):** Reserved for later definition.
- **Datagram (1):** Transmit the frame as a datagram over UDP.
- **Stream (2):** Transmit the frame as part of a stream over TCP or yTCP for an individual neighbor, and over a reliable IP multicast transport protocol for a cluster. When the term stream is used, it generally means either IP unicast or IP multicast. When the two need to be distinguished, the terms *unicast stream* and *cluster stream* are used.
- **Reserved (3-15):** Reserved for later definition.

Protocol: This field identifies the protocol encapsulated by the YDP header. Its purpose, in addition to identifying YTMP messages, is to allow various other yoid protocols with different sorts of semantics (such as end-to-end reliability) to be slipped between the YDP header and the application protocol. The Protocol field has the following values:

- **YTMP (0):** The next protocol is the YTMP protocol. There are never any protocols above YTMP.
- **Upper Layer (1):** The upper layer protocol (as identified by the Group ID) is the next protocol. In other words, there is no other yoid protocol above YDP.
- **YDP (2):** The next protocol is for YDP control. There are never any protocols above YDP.
- **Reserved (3 - 255):** These values are reserved for future yoid protocols.

Header Length: The length, in 32-bit words, of the header, including options.

Source Code: The source member of this frame. This value is a mnemonic, locally defined by the transmitting yoid member, for the full DNS name and UDP port number of the original sender (E2E, not HxH) of the frame. It is modified HxH to match the value expected by the receiving member or members. The combination of Source Code, transmitting neighbor member, and Group ID, identify the sender to the receiver of the frame.

The values 0 and all-ones are reserved for future use. The value 1 means that the source member in unknown. The value 2 means that the source member's name is given in the source member name option field, and is otherwise not to be interpreted as a mnemonic. The value 3 means that the (E2E) source member is the same as the HxH source derived from the ID Code, and is otherwise not to be interpreted as a mnemonic. The remaining values are used as Source Code mnemonics.

Dest Code: This field has different uses depending the frame delivery mode:

Unicast: In the case of unicast frame transmission, this field identifies the destination member for the frame. It matches the Source Code for the same member from a previously received frame. It is modified HxH to match the value expected by the receiving member. The values 0 and all-ones are reserved for future use. The value 1 always means the immediate recipient of the frame, and is used, for instance, to send control frames directly to neighbors. The values 2 and 3 are undefined. The remaining values are mnemonics for destination members matching those learned through the Source Code.

Tree Anycast, Multicast, and Broadcast: In these modes, the field is treated as a bit mask. Bit 1 indicates whether the frame should be forwarded to transit members, not including transit members currently configured as feet. Bit 2 indicates whether the frame should be forwarded to feet. Bit 3 indicates whether the frame should be forwarded to stub members. For each bit, if the bit is a 1, then the frame should be forwarded to the corresponding member type. Otherwise, it should not. All other values should be transmitted as 0, and ignored upon receipt.

Note that bit 1 must be set if either bit 2 or bit 3 are set. (Generally speaking, frames cannot be delivered to leaf members unless they are first delivered to transit members.)

Mesh Anycast: In the case of Mesh Anycast, the Dest Code field is partitioned into two 8-bit fields. The lower 8-bits (bits 1 - 8) are treated as for Tree Anycast, Multicast, and Broadcast defined above, with one exception. If the lower 8-bits indicates transit members, transit feet known to have mesh links (pairwise knowledge) are included in selecting the next hop.

The upper 8 bits (bits 9 - 16) are treated as a hop count field. It is set to some relatively small value by the originating member (say 5 or 6). It is decremented by each member that handles the frame. When it is decremented to 0, it is delivered to the member's upper layer. Note that value of this (upper 8-bit) field must be smaller than that of the Hop Count field.

Hop Count: Like the analogous field in IP, this field is decremented by each member that handles the frame. If it is decremented to 0, the frame is discarded, and an YTMP Error message of Hop Count Expired is transmitted directly (not over the tree) to the frame source member, if known. If a member experiences multiple hop count expirations, from different sources, in a short period of time, it may wish to check for a loop in the topology using the YTMP Root Path Trace message (see "Yoid Topology Management Protocol (YTMP) Specification").

For generating frames, each member has only a single setting for this field, which it uses in all transmitted frames. This setting is something significantly larger than the longest path in the tree. Note that a member will not received Hop Count Expired Error Messages on frames where the source was not identified. Therefore, members should be conservative in setting the Hop Count field for these frames.

Note, however, that a member can test whether a given setting is big enough by attaching the Frame Source Option to any given frame.

Data Length: The length of the frame data following the header, in bytes.

Sequence Number: This 64-bit unsigned number gives the sequence number of the first byte of data in this frame. Bytes are sequenced in the order transmitted by the application at a given member (as identified by the Source Code or Frame Source Option). The sequence number of the first byte transmitted by the application is defined to be 1024. The sequence number space does not wrap around. In the unlikely event that it reaches its maximum value $(2^{6}4 - 1)$, the member must quit the group. It may rejoin with a new IDP port number, starting again from sequence number 1024.

The value 1 is reserved to mean "no sequence number".

Note that the sequence number is not intended to be visible by the application. It is used "internally" by YDP for the purpose of avoiding duplicate or lost frames in the face of topology changes.

4.1 Header Options

Each header option contains the two-byte fixed part shown below:

Type: This defines the type of option.

Act: This indicates what action should be taken if the option is not recognized, as follows:

```
                   1
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+---------------+---------------+
|    Type   |Act| Option Length |
+---------------+---------------+
```

Ignore Silently(0): Ignore the option, but continue processing other options and process the message. Don't notify the sender.

Ignore With Notification (1): Ignore the option, but notify the sender that the option was not recognized with the YTMP Error Message of type Type Unrecognized.

Drop Silently (2): Drop the entire message, don't notify the sender.

Drop With Notification (3): Drop the entire message, and notify the sender.

Option Length: The length of the option, in units of 32-bit words, including the fixed part.

The following sections describe the available header options.

4.1.1 Frame Source Option

The Frame Source Option is formatted as shown below:

```
                   1                   2                   3
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+---------------+---------------+---------------+---------------+
|      1    | 3 | Option Length |          Source Code          |
+---------------+---------------+---------------+---------------+
|        UDP Port Number        |             Flags             |
+---------------+---------------+---------------+---------------+
|        TCP Port Number        |            reserved           |
+---------------+---------------+---------------+---------------+
|                        IDP Port Number                        |
+---------------+---------------+---------------+---------------+
|       Source Name Length      |        Source Name ....
+---------------+---------------+---------/
                                |        Zero Padding           |
                        /-------+---------------+---------------+
```

These fields are defined as follows:

Fixed Part: The Type code is 1, and the Action is 3 (drop with notification). The Option Length is variable.

Source Code: This is the Source Code mnemonic used by the transmitting (neighbor) member. Future frames containing this Source Code from the transmitting neighbor, with the same ID Code, are assumed to have the same source. The value 1 is reserved to mean that the recipient of this frame should not store the Source Code associated with the source name. This is often used for sending control frames to non-neighbor members.

UDP Port Number: This is the UDP port number that can be used to transmit frames directly to the source member for this group. The value 0 is used if frames cannot be transmitted directly to the source member with UDP.

Flags: The following flags are defined:

Reverse Path (bit 17): Set to 1 if the receiving member should remember the reverse path (used for transmitting unicast frames via stream transmission to the source, as encoded in the Dest Code). Set to 0 otherwise. Source Codes with this bit set are called *reverse-path enabled* Source Codes.

Reserved (bits 18 - 32): Transmit as zero, ignore upon receipt.

TCP Port Number: This is the TCP port number that can be used to transmit frames directly to the source member for this group. The value 0 is used if frames cannot be transmitted directly to the source member with TCP.

IDP Port Number: Port number used in the Dest IDP Port field of IDP for the purpose of initializing communications with the member, via the UDP Port Number above, to the source member or one of its upper layers. If the UDP Port Number is set to value 0, this field must be set to value 1.

Source Name Length: The number of bytes in the source DNS name (not including padding, if any).

Source Name: The DNS name of the source, padded, if necessary, to an integral 32-bit boundary with zeros.

This option is attached to the frames sent from member to member in the case where 1) the source member needs to be known, and 2) the receiving member does not already know the Source Code of the transmitting member for this source. Upon receiving this option, the receiving member independently decides whether or not it wishes to record the Source Code (though generally it would).

If it does, and it is not a foot then it transmits an acknowledgement of the Source Code. In the case of non-cluster neighbors, it is transmitted via the (reliable) stream channel between the two members. In the case of a head, it is transmitted over the reliable multicast stream channel of the cluster.

The acknowledgement is contained in the YDP Header Option Acknowledge Message (Section 5.1). If the receiving member does not acknowledge the frame source option, then the transmitting member simply continues to use the frame source option.

Once the transmitting member receives the acknowledgement, future frames with the same source do not require the frame source option. Instead, the Source Code transmitted in the earlier frame source option is written into the Source Code field of the frame header. Once established, the Source Code is not timed-out by the receiving member. The transmitting member can optionally "refresh" the code, however, by sending another frame source option. The transmitting member can, at its own discretion, re-assign the Source Code value with another frame source option.

If an unrecognized Source Code is received, the frame should be dropped, and an Error Message of type Source Code Unrecognized should be transmitted to the sending member (see Section 5.2).

The use of reverse-path enabled Source Codes for stream frames should be used sparingly. This is because routing information must be installed and maintained at members. This comes at a certain cost, felt especially when members change parents. When this happens, the child member must convey to the parent member all of its child-side Source Codes see "Yoid Topology Management Protocol (YTMP) Specification").

The reason that Frame Source Options are acknowledged is because the same Source Code can be used for different original senders over time. If a member uses a given Source Code for one sender at time T1, and changes to a different sender at time T2, then it must know that the receiver of the Source Code recorded the new sender.

A foot, however, will assign and transmit at most one Source Code, that is, for itself. This is because a foot must not have children of its own. It is for this reason that the foot does not require an acknowledgement of its transmitted Frame Source Option from other feet. If the other feet did not record the Frame Source Option, this will be discovered later when the foot sends a frame with the unknown Source Code, thus triggering a Source Code Unrecognized Error Message.

5 YDP Control Messages

This section describes the control messages necessary for YDP operation. These messages are all sent with a YDP Protocol value of 2. They may be sent stream or datagram mode, depending on the situation.

All YDP control messages have a fixed part, formatted as shown below:

The fields are defined as follows:

Message Type: Indicates the type of message. Current message types (with values in parenthesis) are:

```
                   1                   2                   3
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+---------------+---------------+---------------+---------------+
|        Message Type           |    Message Length     |Rsv|Act|
+---------------+---------------+---------------+-------+---+---+
```

Header Option Acknowledge Message (1): Used to acknowledge receipt of a YDP header option.

Source Code Unrecognized Error Message (2): Used to inform the sender of a source code that it is not recognized.

Invalid Neighbor Error Message (3): Used to inform a non-neighbor sender that the received frame is invalid.

Message Length: The length of the message, in units of 32-bit words, including the fixed part.

Rsv: Reserved, transmit as zero, ignore upon receipt.

Act: This field is defined the same as the Act field of the options header.

The following subsections describe each of the messages.

5.1 Header Option Acknowledge Message (1)

This message is used to acknowledge acceptance of a YDP header option. It is transmitted to the sender of the option immediately upon accepting the option. The fixed part of this YDP control message is followed by the YDP fixed header and option that triggered this message. The sending neighbor uses this to determine what it is that is being acknowledged.

5.2 Source Code Unrecognized Error (2)

This message is sent what a member does not recognize the Source Code in the received YDP header. The body of the message contains the offending YDP header.

The member receiving this message should arrange to send the full Source Code Option the next time it transits a frame from the source identified by the Source Code. If the member has not deleted the frame that caused the Error Message, it may retransmit the frame, with the Source Code Option, right away.

5.3 Invalid Neighbor Error (3)

This message is sent when a node receives from a non-neighbor node a frame that may only be sent between neighbors. This includes multicast, broadcast, and anycast frames, and any unicast frames destined for a member other than itself. The body of the message contains the offending YDP header.

5.4 Hop Count Expired Error (4)

This message is used to inform a member that the Hop Count for a frame that it transmitted expired. It is transmitted by the member that decremented the Hop Count to 0 (see Section 4) to the source of the frame, if known. Upon receiving this message for the first time, the source should increase its Hop Count, probably by doubling it. (If over time it does not receive any Hop Count Expired Error Messages, it may slowly incrementally decrease the Hop Count it uses.) If it has already increased the Hop Count to its maximum value, it may assume that there is a loop in the topology, and stop transmitting for a short time (a few seconds), to allow the topology to sort itself out.

If a member receives multiple frames with expired hop counts, from different sources, in a short period of time, it may have YTMP initiate a check for a loop in the topology.

The Hop Count Expired Error message contains no body. (Note that, unlike the case with IP, each member has only a single setting for the Hop Count field — one that is greater than the longest path in the tree. This is true even for frames sent unicast. Therefore, there is nothing to be gained by, for instance, transmitting the Frame Header of the offending frame to the source.)

6 Future

There needs to be a protocol for two neighbors to determine what sequence numbers each has received for each source, and what sequence numbers each would like to receive from the other.