INTERNET-DRAFT

N. Elkins
B. Jouris
Inside Products
K. Haining
U. S. Bank
M. Ackermann
BCBS Michigan

Intended Status: Proposed Standard
Expires: July 2014
January 30, 2014

IPPM Considerations for the IPv6 PDM Extension Header
draft-elkins-ippm-pdm-metrics-04

Table of Contents

Abstract

To diagnose performance and connectivity problems, metrics on real (non-synthetic) transmission are critical for timely end-to-end problem resolution. Such diagnostics may be real-time or after the fact, but must not impact an operational production network. These metrics are defined in the IPv6 Performance and Diagnostic Metrics Destination Option (PDM). The base metrics are: packet sequence number and packet timestamp. Other metrics may be derived from these for use in diagnostics. This document specifies such metrics, their calculation, and usage.

Status of this Memo

Copyright and License Notice

1 Background

To diagnose performance and connectivity problems, metrics on real (non-synthetic) transmission are critical for timely end-to-end problem resolution. Such diagnostics may be real-time or after the fact, but must not impact an operational production network. The base metrics are: packet sequence number and packet timestamp. Metrics derived from these will be described separately. This document starts with the background and rationale for the requirement for end-to-end response time and packet sequence number(s).

Current methods are inadequate for these purposes because they assume unreasonable access to intermediate devices, are cost prohibitive, require infeasible changes to a running production network, or do not provide timely data. The IPv6 Performance and Diagnostic Metrics destination option PDM) provides a solution to these problems.

1.1 Terminology

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 RFC 2119 [RFC2119].

1.2 Why End-to-end Response Time is Needed

The timestamps or delta values in the PDM traveling along with the packet will be used to calculate end-to-end response time, without requiring agents in devices along the path. In many networks, end-to-end response times are a critical component of Service Levels Agreements (SLAs).

End-to-end response is what the user of a network system actually experiences. When the end user is an individual, he is generally indifferent to what is happening along the network; what he really cares about is how long it takes to get a response back. But this is not just a matter of individuals' personal convenience. In many cases, rapid response is critical to the business being conducted.

When the end user is a device (e.g. with the Internet of Things), what matters is the speed with which requested data can be transferred -- specifically, whether the requested data can be transferred in time to accomplish the desired actions. This can be important when the relevant external conditions are subject to rapid change.

Response time and consistency are not just "nice to have". On many networks, the impact can be financial hardship or endanger human life. In some cities, the emergency police contact system operates

over IP, law enforcement uses TCP/IP networks, transactions on our stock exchanges are settled using IP networks. The critical nature of such activities to our daily lives and financial well-being demand a solution. Section 1.5 will detail the current state of end-to-end response time monitoring today.

1.3 Trending of Response Time Data

In addition to the need for tracking current service, end-to-end response time is valuable for capacity planning. By tracking response times, and identifying trends, it becomes possible to determine when network capacity is being approached. This allows additional capacity to be obtained before service levels fall below requirements. Without that kind of tracking, the only option is to wait until there is a problem, and then scramble to get additional capacity on an emergency (and probably high cost) basis.

1.4 What to measure?

End to end response time can be broken down into 3 parts:

- Network delay - Application (or server) delay- Client delay

Network delay may be one-way delay [RFC2679] or round-trip delay [RFC2681].

Additionally, network delay may include multiple hops. Application and server delay include operating system by stack time. By and large, the three timings are 'good enough' measurements to allow rapid triage into the failing component.

Ways are available (provided by operating systems) to measure Application and Client times. Network time can also be measured in isolation via some of the measurement techniques described in section 1.5. The most difficult portion is to integrate network time with the server or application times. Products exist to do this but are available at an exorbitant cost, require agents, and will likely become more prohibitive as the speed of networks grow and as the world becomes more connected via mobile devices.

Measuring network time needs to occur at the end-points of the transactions being measured. The time needs to be available, regardless of the upper layer protocol being used by the transaction. That is, it cannot be for just TCP packets.

1.5 TCP Timestamp not enough

Some suggest that the TCP Timestamp option might be sufficient to

calculate end-to-end response time.

The TCP Timestamp Option is defined in RFC1323 [RFC1323]. The reason for the TCP Timestamp option is to be able to discard packets when the TCP sequence number wraps. (PAWS)

The problems with the TCP Timestamp option are:

1. Not everyone turns this on.
2. It is only available for TCP applications
3. No indication of date in long-running connections. (That is connections which last longer than one day)
4. The granularity of the timestamp is at best at millisecond level.

In the future, as speeds of devices and networks grow and network types proliferate, TCP timestamp values, both in terms of granularity and date specification, will become more and more inadequate. Even today, on many networks, the timings are at microsecond level not millisecond. New networks called Delay Tolerant Networks may have connection times which are very large indeed - hours or even days.

1.6 Inadequacy of Current Instrumentation Technology

The current technology includes:

1. Synthetic transactions
2. Pings
3. Estimates of network time
4. Server / Client Agents

Let us discuss each of these in detail.

1.6.1 Synthetic transactions

Synthetic transactions, also known as active measurement, can be extremely useful. However, in a dynamic network, the routes taken by the packet or the current load on the application may not be the same for the real transaction as when the active test was performed. For example, if you time how long it takes for me to drive to work at 2:00am in the morning, that may not be the same as how long it takes me to drive to work during rush hour at 8:00am in the morning. So, it is important to have embedded measurement in the actual packet.

1.6.2 PING

An ICMP ping measures network time. First, you can PING the remote device. Then you assume that the time it takes to get a response to a PING is the same as the time that a transaction would take to

traverse the network. However, QoS rules, firewalls, etc. may mean that PING, (and other synthetic transactions) may not be subject to the same conditions. PINGs, though extremely useful, also measure only network delays. Server delays must also be provided.

1.6.3 Estimates of Network Time

If a packet trace is done, it is possible to look at the time between when a response was seen to be sent at the packet capture device and when the ACK for the response comes back.

If you assume that the ACK took the same amount of time as the original query, you have the network time. Unfortunately, the time for the ACK may not be the same as the time for a much larger query transaction to traverse the network.

The biggest problem with this method is that of TCP delayed acknowledgements. If the client is doing delayed ACKs, then the ACK will be held until the next request is ready to go out. In this case, the time to receive the ACK has no correlation with network time.

1.6.4 Server / Client Agents

There are also products which claim that they can determine end-to-end response times, integrating server and network times - and indeed they can do so. But they require agents which must be placed at each point which is to be monitored. That is, it is necessary to add those agents EVERYWHERE around the network, at a very high cost - both in terms of manpower, knowledge and costs. These kind of products can be purchased by only the richest 1% of the corporations. As the speed of networks grow, and as the world becomes more connected via mobile devices, such products will only become more expensive. If, indeed, their technology can keep up.

There are many situations where agents cannot be deployed. Many situations which demand a lightweight, cost effective solution. You may think of an ISP with many customers. If the customer complains of poor response time, it is much more cost-effective for the ISP to simply take a packet trace with embedded diagnostics than to instrument the entire customer network.

TCP/IP networks, including the Internet, are used throughout the world. If there is not a scalable and affordable way to measure performance bottlenecks and failures, the growth of these networks will suffer and indeed may reach a plateau where further growth becomes impossible.

2 Solution Parameters

What is needed is:

1) A method to identify and/or track the behavior of a connection without assuming access to the transport devices.

2) A method to observe a connection in flight without introducing agents.

3) a method to observe arbitrary flows at multiple points within a network and correlate the results of those observations in a consistent manner.

4) A method to signal and correlate transport issues to application end-to-end behavior.

5) A method which does not require changes to a production network in real time.

6) Adequate granularity in the measurement technique to provide the needed metrics.

7) A method that is scalable to very large networks.

8) A method that is affordable to all.

2.1 Rationale for proposed solution

The current IPv6 specification does not provide a timestamp nor similar field in the IPv6 main header or in any extension header. So, we propose the IPv6 Performance and Diagnostic Metrics destination option (PDM) [ELKPDM].

2.2 Merits of timestamp / delta in PDM

Advantages include:

1. Less overhead than other alternatives.
2. Real measure of actual transactions.
3. Less cost to provide solutions
4. More accurate and complete information.
5. Independence from transport layer protocols.
6. Ability to span organizational boundaries with consistent instrumentation

In other words, this is a solution to a long-standing problem. The PDM will provide a metric which will allow those responsible for

network support to determine what is happening in their network without expensive equipment (agents) at each device.

The PDM does not solve every response time issue for every situation. Network connections with multiple hops will still need more granular metrics, as will the differentiation between multiple components at each host. That is, TCP/IP stack time vs. applications time will still need to be broken out by client software. What the PDM does provide is the ability to do rapid triage. That is, to determine quickly if the problem is in the network or in the server or application.

2.3 What kind of timestamp?

Questions arise about exactly the kind of timestamp to use. Both the Network Time Protocol (NTP) [RFC5905] and Precision Time Protocol (PTP) [IEEE1588] are used to provide timing on TCP/IP networks.

NTP has evolved within the IETF structure while PTP has evolved within the Institute of Electrical and Electronics Engineers (IEEE) community. By and large, operating systems such as Windows, Linux, and IBM mainframe computers use NTP. These are the source and destination systems for packets. Intermediate nodes such as routers and switches may prefer PTP.

Since we are describing a new extension header for destination systems, the timestamp to be used will be in accordance with NTP. The document, draft-ackermann-ntp-pdm-ntp-usage [NTPPDM], discusses guidelines for implementing NTP for use with the PDM. The timestamp is only relevant for PDM type 1. PDM type 2 uses delta values and requires no time synchronization.

2 Why Packet Sequence Number

While performing network diagnostics of an end-to-end connection, it often becomes necessary to find the device along the network path creating problems. Diagnostic data may be collected at multiple places along the path (if possible), or at the source and destination. Then, the diagnostic data must be matched. Packet sequence number is critical in this matching process. The timestamp or even the IP addresses may be different at different devices. In IPv4 networks, the IPID field was used as a de facto sequence number.

This method of data collection along the path is of special use on large multi-tier networks to determine where packet loss or packet corruption is happening. Multi-tier networks are those which have multiple routers or switches on the path between the sender and the receiver.

2.1 IPv4 IPID : DeFacto Sequence Number

With IPv4 networks, on many stack implementations, but not all, the IPID field has the property of sequentiality. That is, the IP stack sending the packets sent them in numerical order. This was not a requirement for the field, but an implementation which turned out to be quite useful in diagnostics.

2.1.1 Description of IPID in IPv4

In IPv4, the 16 bit IP Identification (IPID) field is located at an offset of 4 bytes into the IPv4 header and is described in RFC0791 [RFC0791]. In IPv6, the IPID field is a 32-bit field contained in the Fragment Header defined by section 4.5 of RFC2460 [RFC2460]. Unfortunately, unless fragmentation is being done by the source node, the IPv6 packet will not contain this Fragment Header, and therefore will have no Identification field.

The intended purpose of the IPID field, in both IPv4 and IPv6, is to enable fragmentation and reassembly, and as currently specified is required to be unique within the maximum segment lifetime (MSL) on all datagrams. The MSL is often 2 minutes.

2.1.2 DeFacto Use of IPID

In a number of networks, the IPID field is used for more than fragmentation. During network diagnostics, packet traces may be taken at multiple places along the path, or at the source and destination. Then, packets can be matched by looking at the IPID.

The inclusion of the IPID makes it easier to identify flows belonging to a single node, even if that node might have a different IP address. For example, in the case of sessions going through a NAT or proxy server.

For its de-facto diagnostic mode usage, the IPID field needs to be available whether or not fragmentation occurs. It also needs to be unique in the context of the session, and across all the connections controlled by the stack. In IPv4, the IPID is in the main header, so it is available for all packets. As it is a 16-bit field, it wrapped during the course of the session and thus had some limitations.

Even with these limitations, the IPID has been valuable and useful in IPv4 for diagnostics and problem resolution. It is a practical solution that is 'good enough' in many instances. Not having it available in IPv6, may be a major detriment to new IPv6 deployments and contribute to protracted downtimes in existing IPv6 operations.

2.1.3 Merits of DeFacto Usage

As network technology evolves, the uses to which fields are put can change as well. De-facto use is powerful, and should not be lightly ignored. In fact, it is a testament to the power and pervasiveness of the protocol that users create new uses for the original technology.

For example, the use of the IPID goes beyond the vision of the original authors. This sort of thing has happened with numerous other technologies and protocols.

The implementation of the traceroute command sends ICMP echo packets with a varying TTL. This is a very useful for diagnostics yet departs from the original purpose of TTL.

Similarly, cell phones have evolved to be more than just a means of vocal communication, including Internet communications, photo-sharing, stock exchange transactions, etc. Indeed, the Internet itself has evolved, from a small network for researchers and the military to share files into the pervasive global information superhighway that it is today.

2.1.4 Use Cases of IPv4 IPID in Diagnostics

Use Case # 1 --- Large Insurance Company

- (estimated time saved by use of IPID: 7 hours)

Performance Tool produces extraneous packets

- Issue was whether a performance tool was accurately replicating session flow during performance testing.

- Trace IPIDs showed more unique packets within same flow from performance tool compared to IE Browser.

- Having the clear IPID sequence numbers also showed where and why the extra packets were being generated.

- Solution: Problem rectified in subsequent version of performance tool.

- Without IPID, it was not clear if there was an issue at all.

Use Case #2 --- Large Bank

-   (estimated time saved by use of IPID: 4 hours)

Batch transfer duration increases 12x

- A data transfer which formerly took 30 minutes to complete started taking 6-8 hours to complete.

- Was there packet loss? All the vendors said no.

- The other applications on the network did not report any problems.

- 4 trace points were used, and the IPIDs in the packets were compared.

- The comparison showed 7% packet loss.

- Solution: WAN hardware was replaced and problem fixed.

- Without IPID, no one would agree a problem existed

Use Case #3 --- Large Bank

-    (estimated time saved by use of IPID: 6 hours)

Very slow interactive performance

- All network links looked good.

- Traces showed duplicated small packets (which can be OK).

- We saw that the IPID was the same in both packets but the TTL was always + 1.

- A network device was "splitting" only small packets over two interfaces.

- The small packets were control info, telling other side to slow down.

- It erroneously looked like network congestion.

- Solution: Network device replaced and good interactive performance restored.

- Without IPID, flows would have appeared OK.

Use Case #4 --- Large Government Agency

-    (estimated time saved by use of IPID: 9 hours)

VPN drops

- Cell phone connections to law enforcement were being dropped. The connections were going through a VPN.

- All parties (both sides of VPN connection, application, etc.) said it was not their problem. The problem went on for weeks.

- Finally, we took a trace which showed packets with IPID and TTL that did not match others in the flow AT ALL coming from the router nearest the application server end of VPN.

- Solution: Provider for VPN for application server changed. Problem resolved.

- Without IPID, much harder to diagnose problem. Same case also happened with large corporation. Again, all   parties saying not their fault until proven via packet trace.)

2.2 TCP sequence number is not enough

TCP Sequence number is defined in RFC0793 [RFC0793]. Some have proposed that this field will meet the needs of diagnostics for a packet sequence number. Indeed, the TCP Sequence Number along with the TCP Acknowledgment number can be used to calculate dropped packets, duplicate packets, out-of-order packets etc. That is, IF the packet flow itself reflects accurately what happened on the wire!

See Scenario 1 (Section 1.5.2) and Scenario 2 (Section 1.5.3) for what happens with packet trace capture in real networks.

The TCP Sequence Number is, obviously, available only for TCP and not other higher layer protocols.

2.3 Inadequacy of current measurement techniques

The question arises of whether current methods of instrumentation cannot be used without a change to the protocol. Current methods of measuring network data, other than packet traces, are inadequate because they assume unreasonable access to intermediate devices, are cost prohibitive, require infeasible changes to a running production network, or do not provide timely data. This section will discuss each of these in detail.

Current methods include both instrumentation and third party products. These include SNMP, CMIP, router logs, and firewall logs.

2.3.1 SNMP / CMIP Counters

The traditional network performance counters measured by SNMP or CMIP do not provide information at the granularity desired on the behavior of application flows across the network. The problem is that such counters do not contain enough data be able to provide a detailed and realistic view of the end-to-end behavior of a connection.

2.3.2 Router / Firewall Logs

Router and firewall logs may provide some information for diagnostics Routers and firewalls in a production network are generally set to do minimal logging and diagnostics to allow maximum efficiency and throughput. Such devices cannot be asked to collect detailed data for an operational problem, as this requires a change to a production network.

2.3.3 Netflow

Netflow is instrumentation which is available from some middle devices. In production networks, such devices are generally set to do minimal logging and diagnostics to allow maximum efficiency and throughput.

It is often also not possible to start data collection in the middle of the day on a production network.

2.3.4 Access to Intermediate Devices

The above current methods require access to the transport infrastructure - that is, the routers, switches or other intermediate devices. In some cases, this is possible; in others, the connections in question may cross a number of administrative entities (both in the transport and in the endpoints). When it is the enterprise at the endpoint which is interested in the diagnostics, the administrative entities who own the devices in the middle of the path have no stake in operational measurement at the enterprise or application level.   They have no reason to provide the necessary data or to impact the basic transport with the instrumentation necessary to capture flow-oriented data as a continuous stream suitable for general consumption.

In other words, if you don't own the path end-to-end, you will not be able to get the data you need if you are required to get it from the devices in the middle. Not only that, the devices in the middle do

not have the instrumentation necessary to make it easy to do end-to-end diagnostics because they are not responsible for that and so do not want to burden their devices with doing those kind of functions.

Many networks may not own the path end-to-end. They may be working with a business partner's network or crossing the Internet.

2.3.4 Modifications to an Operational Production Network

Even when the enterprise does own all the devices along the entire path, to get enough data to adequately resolve a problem means changing the device configuration to do detailed diagnostics. In a production network, devices are generally set to do minimal logging and diagnostics. This is to allow maximum efficiency and throughput. The more logging and diagnostics such devices do, the fewer resources they have for actually transmitting traffic across the network.

So, if devices are to be asked to collect more data for an operational problem, this requires a change to a production network. This is generally not possible as it destabilizes a critical network during business hours, thus potentially disrupting many customers. Making changes is usually a lengthy process requiring change control, testing on a test network, etc. On networks which are critical to the business function, changing configuration "in flight" is generally not an option.

3 Solution Parameters

What is needed is:

1) A method to identify and/or track the behavior of a connection without assuming access to the transport devices.

2) A method to observe a connection in flight without introducing agents at endpoints.

3) A method to observe arbitrary flows at multiple points within a network and correlate the results of those observations in a consistent manner.

4) A method to signal and correlate transport issues to application end-to-end behavior.

5) A method which does not require changes to a production network in real time.

6) Adequate granularity in the measurement technique to provide the needed metrics.

3.1 Packet Trace Meets Criteria

The only instrumentation which provides enough detail to diagnose end-to-end problems is a packet trace. Packet traces do not require changes to devices in production mode because in many networks, products are available to capture packets in passive mode. Such products continuously monitor network traffic. Often, they are used not for diagnostic reasons but for regulatory reasons. For example, there may be legal requirements to log all stock exchange transactions.

Products for packet tracing are available freely and can be used at a client host without disrupting major portions of the network.

3.1.1 Limitations of Packet Capture

Even though packets are the only reliable way to provide data at the needed granularity, there are limitations with collecting packet traces in some situations. They are as follows:

3.1.2 Problem Scenario 1

1. Packets are captured for analysis at places like large core switches. All packets are kept. Again, not necessarily for diagnostic reasons but for regulatory ones. For example, records of all stock trades may need to be kept for a certain number of years.

2. When there is a problem, an analyst extracts the needed information.

3. If the extract is done incorrectly, as often happens, or the packet capture itself is incorrect, then there may be false duplicate packets which can be quite misleading and can lead to wrong conclusions. Are these real TCP duplicates? Is there congestion on the subnet? Are these retransmissions? Situations have been seen where routers incorrectly send two packets instead of one - is this such a situation?

4. This is the type of problem that can be solved by having an IP packet sequence number.

3.1.2 Problem Scenario 2

1. In this scenario, packets are captured for analysis at places like a middleware box. It may be because problems are suspected with the box itself or it is a central point of the suspected failure.

2. The box may not offer any way to tailor the packet capture. "You

will get what we give you, how we give it to you!" is their philosophy.

3. The packet capture incorrectly duplicates only packets going to certain nodes.

4. Again, there are false duplicate packets which can be misleading and can lead to wrong conclusions. Are these real TCP duplicates? Is there congestion on the subnet? Situations have been seen where routers incorrectly send two packets instead of one - is this such a situation?

4 Rationale for Proposed Solution (PDM)

The current IPv6 specification does not provide a packet sequence number or similar field in the IPv6 main header. One option might be to force all IPv6 packets to contain a Fragment Header. In packets which are entire in and of themselves, the fragment ID would be zero-that is, an atomic fragment. Why was a new destination option header defined rather than recommending that Fragment Header be used?

Our reasoning was that the PDM destination option header would provide multiple benefits : the packet sequence number and the timings to calculate response time.

As defined in RFC2460 [RFC2460], destination options are carried by the IPv6 Destination Options extension header. Destination options include optional information that need be examined only by the IPv6 node given as the destination address in the IPv6 header, not by routers in between.

The PDM DOH will be carried by each packet in the network, if this is configured. That is, the PDM DOH is optional. If the user of the OS configures the PDM DOH to be used, then it will be carried in the packet.

The metrics in the PDM are for 'real' or passive data. That is, they are of the traffic actually traveling on the network.

5 Performance and Diagnostic Metrics Destination Option Layout

5.1 Destination Options Header

The IPv6 Destination Options Header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options Header is identified by a Next Header value of 60 in the immediately preceding header and is defined in RFC2460 [RFC2460].

5.2 PDM Types

The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) is an implementation of the Destination Options Header (Next Header value = 60). Two types of PDM are defined. PDM type 1 requires time synchronization. PDM type 2 does not require time synchronization.

PDM type 1 and PDM type 2 are mutually exclusive. That is, a 5-tuple can either both send PDM type 1 or both send PDM type 2.

5.3 Performance and Diagnostic Metrics Destination Option (Type 1)

PDM type 1 is used to facilitate diagnostics by including a packet sequence number and timestamp.

The PDM type 1 is encoded in type-length-value (TLV) format as follows:

```
    0                   1                   2                   3
    0 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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Option Type  | Option Length | PSN This Packet               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    |                                                               |
    +                          TimeStamp This Packet (64-bit)       +
    |                                                               |
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   PSN Last Packet              |   Reserved                   |
    |-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                                                               +
    |                                                               |
    +                          TimeStamp Last Packet (64-bit)       +
    |                                                               |
    +                                                               +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

TBD = 0xXX (TBD) [To be assigned by IANA] [RFC2780]

Option Length

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. This field MUST be set to 22.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for human use.

Initialized at a random number and monotonically incremented for packet on the 5-tuple. The 5-tuple consists of the source and destination IP addresses, the source and destination ports, and the upper layer protocol (ex. TCP, ICMP, etc).

Operating systems MUST implement a separate packet sequence number counter per 5-tuple. Operating systems MUST NOT implement a single counter for all connections.

Note: This is consistent with the current implementation of the IPID field in IPv4 for many, but not all, stacks.

TimeStamp This Packet (TSTP)

A 64-bit unsigned integer field containing a timestamp that this packet was sent by the source node. The value indicates the number of seconds since January 1, 1970, 00:00 UTC, by using a fixed point format. In this format, the integer number of seconds is contained in the first 32 bits of the field, and the remaining 32 bits resolve to picoseconds.

This follows timestamp formats used in Network Time Protocol (NTP) [RFC5905] and SEND [RFC3971]. A discussion of how to implement NTP for use with PDM header type 1 is in draft-ackermann- ntp-pdm-ntp-usage-00 [NTPPDM].

Implementation note: This format is compatible with the usual representation of time under UNIX, although the number of bits available for the integer and fraction parts in different Unix implementations vary.

Packet Sequence Number Last Received (PSNLR)

16-bit unsigned integer. This is the PSN of the packet last received on the 5-tuple.

TimeStamp Last Received (TSLR)

A 64-bit unsigned integer field containing a timestamp. This is the timestamp of the packet last received on the 5-tuple. Format is the same as TSTP.

5.4 Performance and Diagnostic Metrics Destination Option (Type 2)

The second type of IPv6 Performance and Diagnostic Metrics Destination Option (PDM) is as follows. PDM type 1 and PDM type 2 are mutually exclusive. That is, a 5-tuple can either both send PDM type 1 or both send PDM type 2.

PDM type 2 contains the following fields:

PSNTP	:	Packet Sequence Number This Packet
PSNLR	:	Packet Sequence Number Last Received
DELTALR	:	Delta Last Received
PSNLS	:	Packet Sequence Number Last Sent
DELTALS	:	Delta Last Sent

PDM destination option type 2 is encoded in type-length-value (TLV) format as follows:

```
   0                   1                   2                   3
   0 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Option Type  | Option Length | PSN This Packet               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   PSN Last Received           |  PSN Last Sent                |
   |-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Delta Last Received         |  Delta Last Sent              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | TType |
   +-+-+-+-+
```

Option Type

TBD = 0xXX (TBD) [To be assigned by IANA] [RFC2780]

Option Length

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. This field MUST be set to 22.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for human use.

Initialized at a random number and monotonically incremented for packet on the 5-tuple. The 5-tuple consists of the source and destination IP addresses, the source and destination ports, and the upper layer protocol (ex. TCP, ICMP, etc).

Operating systems MUST implement a separate packet sequence number counter per 5-tuple. Operating systems MUST NOT implement a single counter for all connections.

Note: This is consistent with the current implementation of the IPID field in IPv4 for many, but not all, stacks.

Packet Sequence Number Last Received (PSNLR)

16-bit unsigned integer. This is the PSN of the packet last received on the 5-tuple.

Packet Sequence Number Last Sent (PSNLS)

16-bit unsigned integer. This is the PSN of the packet last sent on the 5-tuple.

Delta TimeStamp Type (TIMETYPE)

4-bit unsigned integer. This is the type of time contained in the delta fields below.

0 - unknown
1 - time is in units of nanoseconds
2 - time is in units microseconds
3 - time is in units of milliseconds
4 - time is in units of seconds
5 - time is in units of minutes
6 - time is in units of hours
7 - time is in units of days

The values 5 - 7 are relevant for Delay Tolerant Networks (DTN) which may operate with long delays between packets.

Delta Last Received (DELTALR)

A 16-bit unsigned integer field. This is server delay.

DELTALR = Send time packet 2 - Receive time packet 1

The value is according to the scale in TIMETYPE.

Delta Last Sent (DELTALS)

A 16-bit unsigned integer field. This is round trip or end-to-end time.

Delta Last Sent = Receive time packet 2 - Send time packet 1

The value is in according to the scale in TIMETYPE.

Option Type

The two highest-order bits of the Option Type field are encoded to indicate specific processing of the option; for the PDM destination option, these two bits MUST be set to 00. This indicates the following processing requirements:

00 - skip over this option and continue processing the header.

RFC2460 [RFC2460] defines other values for the Option Type field. These MUST NOT be used in the PDM. The other values are as follows:

01 - discard the packet.

10 - discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

In keeping with RFC2460 [RFC2460], the third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination.

In the PDM, the value of the third-highest-order bit MUST be 0. The possible values are as follows:

0 - Option Data does not change en-route

1 - Option Data may change en-route

The three high-order bits described above are to be treated as part of the Option Type, not independent of the Option Type. That is, a particular option is identified by a full 8-bit Option Type, not just the low-order 5 bits of an Option Type.

6 Use of the PDM

6.1 Packet Identification Data

Each packet contains information about the sender and receiver. In IP protocol the identifying information is called a "5-tuple". The flows described below are for the set of packets flowing between A and B without consideration of any other packets sent to any other device from Host A or Host B.

The 5-tuple consists of:

```
SADDR : IP address of the sender
SPORT : Port for sender
DADDR : IP address of the destination
DPORT : Port for destination
PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP, etc.)
```

6.2 Data in the PDM Destination Option Headers

The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) is an implementation of the Destination Options Header (Next Header value = 60). Two types of PDM are defined. PDM type 1 requires time synchronization. PDM type 2 does not require time synchronization.

PDM type 1 and PDM type 2 are mutually exclusive. That is, a 5-tuple can either both send PDM type 1 or both send PDM type 2.

PDM type 1 contains the following fields:

```
PSNTP : Packet Sequence Number This Packet
TSTP  : Timestamp This Packet
PSNLR : Packet Sequence Number Last Received
TSLR  : Timestamp Last Received
```

PDM type 2 contains the following fields:

```
PSNTP    : Packet Sequence Number This Packet
```

PSNLR	:	Packet Sequence Number Last Received
DELTALR	:	Delta Last Received
PSNLS	:	Packet Sequence Number Last Sent
DELTALS	:	Delta Last Sent

The metrics which may be derived from these fields will be discussed in the following sections.

7 Metrics Derived from the PDM Destination Options

A number of metrics may be derived from the data contained in the PDM. Some are relationships between two packets, others require analysis of multiple packets or multiple protocols.

These metrics fall into the following categories:

1. Base derived metrics
2. Metrics used for triage
3. Metrics used for network diagnostics
4. Metrics used for session classification
5. Metrics used for end user performance optimization

It must be understood that when a metric is discussed, it includes the average, median, and other statistical variations of that metric.

In the next section, we will discuss the base metrics. In later sections, we will discuss the more advanced metrics and their uses.

8 Base Derived Metrics

The base metrics which may be derived from the PDM are:

1. One-way delay
2. Round-trip delay
3. Server delay

8.1 One-Way Delay

One-way delay is the time taken to traverse the path one way between one network device to another. The path from A to B is distinguished from the path from B to A. For many reasons, the paths may have different characteristics and may have different delays. One-way delay is discussed in "A One-way Delay Metric for IPPM" [RFC2679].

8.2 Round-Trip Delay

Round-trip delay is the time taken to traverse the path both ways between one network device to another. The entire delay to travel

from A to B and B to A is used. Round-trip delay cannot tell if one path is quite different from another. Round-trip delay is discussed in "A Round-trip Delay Metric for IPPM" [RFC2681].

8.3 Server Delay

Server delay is the interval between when a packet is received by a device and a subsequent packet is sent back in response. This may be "Server Processing Time". It may also be a delay caused by acknowledgements. Server processing time includes the time taken by the combination of the stack and application to return the response.

9 Sample Implementation Flow (PDM Type 1)

Following is a sample simple flow with one packet sent from Host A and one packet received by Host B.

Time synchronization is required between Host A and Host B. See draft-ackermann-ntp-pdm-ntp-usage-00 [NTPPDM] for a description of how an NTP implementation may be set up to achieve good time synchronization.

Each packet, in addition to the PDM, contains information on the sender and receiver. This is the 5-tuple consisting of:

```
SADDR : IP address of the sender
SPORT : Port for sender
DADDR : IP address of the destination
DPORT : Port for destination
PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP, etc.)
```

It should be understood that the packet identification information is in each packet. We will not repeat that in each of the following steps.

9.1 Step 1 (PDM Type 1)

Packet 1 is sent from Host A to Host B. The time for Host A is set initially to 10:00AM.

The timestamp and packet sequence number are sent in the PDM.

The initial PSNTP from Host A starts at a random number. In this case, 25. The sub-second portion of the timestamp has been omitted for the sake of simplicity.

```
            Packet 1

            +----------+             +----------+
            |          |             |          |
            |   Host   | ----------> |   Host   |
            |    A     |             |    B     |
            |          |             |          |
            +----------+             +----------+

            PDM Contents:
            PSNTP : Packet Sequence Number This Packet:   25
            TSTP  : Timestamp This Packet:                10:00:00
            PSNLR : Packet Sequence Number Last Received: -
            TSLR  : Timestamp Last Received:              -
```

There are no derived statistics after packet 1.

9.2 Step 2 (PDM Type 1)

Packet 1 is received by Host B. The time for Host B was synchronized with Host A. Both were set initially to 10:00AM.

The timestamp and PSN for the received packet are placed in the PSNLR and TSLR fields. These are from the point of view of B. That is, they indicate when the packet from A was received and which packet it was.

The PDM is not sent at this point. It is only prepared. It will be sent when the response to packet 1 is sent by Host B.

```
            Packet 1 Received

                 +----------+             +----------+
                 |          |             |          |
                 |   Host   | ----------> |   Host   |
                 |    A     |             |    B     |
                 |          |             |          |
                 +----------+             +----------+

            PDM Contents:

            PSNTP : Packet Sequence Number This Packet:   -
            TSTP  : Timestamp This Packet:                -
            PSNLR : Packet Sequence Number Last Received: 25
            TSLR  : Timestamp Last Received:              10:00:03
```

At this point, the following metric may be derived: one-way delay. In fact, we now know the one-way delay and the path. We will call this

path 1. This will be the outbound path from the point of view of Host A and the inbound path from the point of view of Host B.

The calculation of one-way delay (path 1) is as follows:

One-way delay (path 1) = Time packet 1 was received by B - Time Packet 1 was sent by A

If we make the substitutions from our sample case above, then:

One-way delay (path 1) = 10:00:03 - 10:00:00 or 3 seconds

9.3 Step 3 (PDM Type 1)

Packet 2 is sent from Host B to Host A. The initial PSNTP from Host B starts at a random number. In this case, 12.

```
            Packet 2

                   +----------+             +----------+
                   |          |             |          |
                   |   Host   | <---------- |   Host   |
                   |    A     |             |    B     |
                   |          |             |          |
                   +----------+             +----------+
```

PDM Contents:

PSNTP	Packet Sequence Number This Packet:	12
TSTP	Timestamp This Packet:	10:00:07
PSNLR	Packet Sequence Number Last Received:	25
TSLR	Timestamp Last Received:	10:00:03

After Packet 2 is sent, the following metric may be derived: server delay.

The calculation of server delay is as follows:

Server delay = Time Packet 2 is sent by B - Time Packet 1 was received by B

Again, making the substitutions from the sample case: Server delay = 10:00:07 - 10:00:03 or 4 seconds

Further elaborations of server delay may be done by limiting the data length to be greater than 1. Some protocols, for example, TCP, have acknowledgements with a data length of 0 or keep-alive packets with a data length of 1. An ACK may preceed the actual response data

packet. Keep-alives may be interspersed within the data flow.

9.4 Step 4 (PDM Type 1)

Packet 2 is received by Host A.

The timestamp and PSN for the received packet are placed in the PSNLR and TSLR fields. These are from the point of view of A. That is, they indicate when the packet from B was received and which packet it was.

The PDM is not sent at this point. It is only prepared. It will be sent when the NEXT packet to Host B is sent by Host A.

```
              Packet 2 Received

                    +----------+             +----------+
                    |          |             |          |
                    |   Host   | <---------- |   Host   |
                    |    A     |             |    B     |
                    |          |             |          |
                    +----------+             +----------+

            PDM Contents:

            PSNTP : Packet Sequence Number This Packet:   -
            TSTP  : Timestamp This Packet:                -
            PSNLR : Packet Sequence Number Last Received: 12
            TSLR  : Timestamp Last Received:              10:00:10
```

However, at this point, the following metric may be derived: one-way delay (path 2).

The calculation of one-way delay (path 2) is as follows:

One-way delay (path 2) = Time packet 2 received by A - Time packet 2 sent by B

If we make the substitutions from our sample case above, then:

One-way delay (path 2) = 10:00:10 - 10:00:07 or 3 seconds

9.5 Step 5 (PDM Type 1)

Packet 3 is sent from Host A to Host B.

```
            Packet 3

                 +----------+             +----------+
                 |          |             |          |
                 |   Host   | ----------> |   Host   |
                 |    A     |             |    B     |
                 |          |             |          |
                 +----------+             +----------+

            PDM Contents:

            PSNTP : Packet Sequence Number This Packet:   26
            TSTP  : Timestamp This Packet:                10:00:50
            PSNLR : Packet Sequence Number Last Received: 12
            TSLR  : Timestamp Last Received:              10:00:10
```

At this point the PDM flows across the network revealing the last received timestamp and PSN.

10 Sample Implementation Flow (PDM 2)

Following is a sample simple flow for PDM type 2 with one packet sent from Host A and one packet received by Host B. PDM type 2 does not require time synchronization between Host A and Host B. The calculations to derive meaningful metrics for network diagnostics is shown below each packet sent or received.

Each packet, in addition to the PDM contains information on the sender and receiver. As discussed before, a 5- tuple consists of:

```
SADDR : IP address of the sender
SPORT : Port for sender
DADDR : IP address of the destination
DPORT : Port for destination
PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP)
```

It should be understood that the packet identification information is in each packet. We will not repeat that in each of the following steps.

10.1 Step 1 (PDM Type 2)

Packet 1 is sent from Host A to Host B. The time for Host A is set initially to 10:00AM.

The timestamp and packet sequence number are noted by the sender internally. The packet sequence number and timestamp are sent in the packet.

Packet 1

```
                +----------+             +----------+
                |          |             |          |
                |   Host   | ----------> |   Host   |
                |    A     |             |    B     |
                |          |             |          |
                +----------+             +----------+
```

PDM type 2 Contents:

PSNTP	: Packet Sequence Number This Packet:	25
PSNLR	: Packet Sequence Number Last Received:	-
DELTALR	: Delta Last Received:	-
PSNLS	: Packet Sequence Number Last Sent:	-
DELTALS	: Delta Last Sent:	-

Internally, within the sender, Host A, it must keep:

PSNTP	: Packet Sequence Number This Packet:	25
TSTP	: Timestamp This Packet:	10:00:00

Note, the initial PSNTP from Host A starts at a random number. In this case, 25. The sub-second portion of the timestamp has been omitted for the sake of simplicity.

There are no derived statistics after packet 1.

10.2 Step 2 (PDM Type 2)

Packet 1 is received at Host B. His time is set to one hour later than Host A. In this case, 11:00AM

Internally, within the receiver, Host B, it must keep:

PSNLR	: Packet Sequence Number Last Received:	25
TSLR	: Timestamp Last Received :	11:00:03

Note, this timestamp is in Host B time. It has nothing whatsoever to do with Host A time.

At this point, we have no derived statistics. In PDM type 1, the derived statistic one-way delay (path 1) could have been calculated. In PDM type 2, this is not possible because there is no time synchronization.

10.3 Step 3 (PDM Type 2)

Packet 2 is sent by Host B to Host A. Note, the initial PSNTP from Host B starts at a random number. In this case, 12. Before sending the packet, Host B does a calculation of deltas. Since Host B knows when it is sending the packet, and it knows when it received the previous packet, it can do the following calculation:

Sending time (packet 2) - receive time (packet 1)

We will call the result of this calculation: Delta Last Received.

That is:

DELTALR = Sending time (packet 2) - receive time (packet 1)

Note, both sending time and receive time are saved internally in Host B. They do not travel in the packet. Only the Delta is in the packet.

Assume that within Host B is the following:

```
            PSNLR : Packet Sequence Number Last Received:   25
            TSLR  : Timestamp Last Received            :   11:00:03
            PSNTP : Packet Sequence Number This Packet  :   12
            TSTP  : Timestamp This Packet              :   11:00:07
```

Hence, DELTALR becomes:

4 seconds = 11:00:07 - 11:00:03

Let us look at the PDM, and then we will look at the derived metrics at this point.

```
            Packet 2

                 +----------+             +----------+
                 |          |             |          |
                 |   Host   | <---------- |   Host   |
                 |    A     |             |    B     |
                 |          |             |          |
                 +----------+             +----------+
```

```
                    PDM Type 2 Contents:

               PSNTP    : Packet Sequence Number This Packet:    12
               PSNLR    : Packet Sequence Number Last Received:  25
               DELTALR  : Delta Last Received:                    4
               PSNLS    : Packet Sequence Number Last Sent:       -
               DELTALS  : Delta Last Sent:                        -
```

After Packet 2, the following metrics may be derived:

Server delay = DELTALR

Metrics left to be calculated are the path delay for path 2. This may be calculated when Packet 3 is sent. Clearly, if there is NO next packet for the 5-tuple, then this value will be missing.

10.4 Step 4 (PDM Type 2)

Packet 2 is received at Host A. Remember, its time is set to one hour earlier than Host B. It will keep internally:

```
               PSNLR : Packet Sequence Number Last Received:   12
               TSLR  : Timestamp Last Received             :   10:00:12
```

Note, this timestamp is in Host A time. It has nothing whatsoever to do with Host B time.

At this point, we have two derived metrics:

1. Two-way delay or Round Trip time
2. Total end-to-end time

The formula for end-to-time is:

Time Last Received - Time Last Sent

For example, packet 25 was sent by Host A at 10:00:00. Packet 12 was received by Host A at 10:00:12 so:

End-to-End response time = 10:00:12 - 10:00:00 or 12

This derived metric we will call DELTALS or Delta Last Sent.

To calculate two-way delay, the formula is:

Two-way delay = DELTALS - DELTALR

Or:

Two-way delay = 12 - 4 or 8

Now, the only problem is that at this point all metrics are in the Host and not exposed in a packet. To do that, we need a third packet.

10.5 Step 5 (PDM Type 2)

Packet 3 is sent from Host A to Host B.

```
            Packet 3

                    +----------+            +----------+
                    |          |            |          |
                    |   Host   | ---------> |   Host   |
                    |    A     |            |    B     |
                    |          |            |          |
                    +----------+            +----------+
```

PDM Type 2 Contents:

```
            PSNTP    : Packet Sequence Number This Packet:    26
            PSNLR    : Packet Sequence Number Last Received:  12
            DELTALR  : Delta Last Received:                   *
            PSNLS    : Packet Sequence Number Last Sent:      25
            DELTALS  : Delta Last Sent:                       12
```

11 Derived Metrics : Advanced

A number of more advanced metrics may be derived from the data contained in the PDM. Some are relationships between two packets, others require analysis of multiple packets. The more advanced metrics fall into the categories shown below:

1. Metrics used for triage
2. Metrics used for network diagnostics
3. Metrics used for session classification
4. Metrics used for end user performance optimization

We will discuss each of these in turn.

11.1 Advanced Derived Metrics : Triage

In this case, triage means to distinguish between problems occurring on the network paths or the server. The PDM provides one-way delay and server delay. This will enable distinguishing which path is a bottleneck as well as whether the server is a bottleneck.

11.2 Advanced Derived Metrics : Network Diagnostics

The data provided by the PDM may be used in combination with data fields in other protocols. We will call this Inter-Protocol Network Diagnostics (IPND).

The PDM also allows us to use only a single trace point for a number of diagnostic situations where today we need to trace at multiple points to get required data. In diagnostics, there is often the question of did the end device really send the packet and it got lost in the network or did it not send it at all.

So, what is done is that diagnostic traces are run at both client and server to get the required data. With the data provided by the PDM, in a number of the cases, this will not be necessary.

For example, taking PDM values along with data fields in the TCP protocol, the following may be found:

1. Retransmit duplication (RD)
2. ACK lag (AL)
3. Third-party connection reset (TPCR)
4. Elapsed time connection reset (ETCR)

A description of these follows.

11.2.1 Retransmit Duplication (RD)

The TCP protocol will retransmit segments given indications from the partner that it has not received them. The retransmitted segments contain the TCP sequence number and acknowledgement. The sequence number is started at a random number and increased by the amount of data sent in each packet.

Consider the following scenario.   There is a packet sequence number in the packet at the IP layer. This is in the PDM that we have defined. The TCP sequence number already exists in the protocol.

Host A sends the following packets:

```
   IP PSN 20, TCP SEQ 10
   IP PSN 21, TCP SEQ 11
   IP PSN 22, TCP SEQ 12

   Host B receives:

   IP PSN 20, TCP SEQ 10
   IP PSN 22, TCP SEQ 12
```

Host B indicates to Host A to resend packet with TCP SEQ 2. Retransmits are done at the TCP layer.

Host A sends the following packet:

IP PSN 23, TCP SEQ 11

The packet never reaches B. B waits until a timeout for retransmits expires. It asks for the packet again.

Host A sends the following packet:

IP PSN 24, TCP SEQ 11

This time, it reaches Host B. Having the combination of PSN (as provided in the PDM) and the TCP sequence number allows us to see whether the problem is that the network is losing the packet or somehow, the sender is not sending the packet correctly.

As we said before, this also allows us a single trace point rather than at the client and server to get the required data.

11.2.2 ACK Lag (AL)

Some protocols, such as TCP, acknowledge packets. The PDM will allow or a calculation of rate of ACKs. Clients can be reconfigured to optimize acknowledgements and to speed traffic flow.

11.2.3 Third-party Connection Reset (TPCR)

Connections may be aborted by a packet containing a particular flag. In the TCP protocol, this is the RESET flag. Sometimes a third-party, for example, a VPN router, will abort the connection. This may happen because the router is overloaded, the traffic is too noisy, or other reasons. This can also be quite hard to detect because the third-party will spoof the address of the sender.

Much time can be spent by the two endpoints pointing fingers at the other for having dropped the connection.

Such a third-party spoofer would likely not have the PDM Destination Option. Routers and other middle boxes are not required to support the Destination Options Extension Header. Even if a PDM DOH was generated, it would most likely violate the pattern of PSNs and time stamps being used. This would be a clue to the diagnostician that the TPCR event has occurred.

11.2.4 Potential Hang (PH)

Connections may be aborted by a packet containing a particular flag. In the TCP protocol, this is the RESET flag. Sometimes this is done because a set amount of time has elapsed without activity. The PSN in the PDM can be used to determine the last packet sent by the partner and if a response is required -- a "hang" situation.

This can be distinguished from connections which are set to be aborted after a certain period of inactivity.

11.3 Advanced Metrics : Session Classification

The PDM may be used to classify sessions as follows:

One way traffic flow
Two way traffic flow
One way traffic flow with keep-alive
Two way traffic flow with keep-alive
Multiple send traffic flow
Multiple receive traffic flow
Full duplex traffic flow
Half duplex traffic flow

Immediate ACK data flow
Delayed ACK data flow
Proxied ACK data flow

A session classification system will assist the network diagnostician. This system will also help in categorizing the server delay.

12 Use Cases

The scheme outlined above can also handle the following types of cases:

1. Host clocks not synchronized (shown above)
2. IP fragmentation
3. Multiple sends from one side (multiple segments)
4. Out of order segments
5. Retransmits
6. One-way transmit only (ex. FTP)
7. One-way transmit only
   (e.g.real time transports and streaming protocols)
8. Duplicate ACKs
9. Duplicate segments
10. Delayed ACKs

11. ACKs preceeding send for another reason
12. Proxy servers
13. Full duplex traffic
14. Keep alive (0 / 1 byte segments, larger segments)
15. No response from other side
16. Drop without retransmit (real time transports)
17. Looped packets (where the same packet may pass the same point multiple times without duplication)
18. Multihoming via SHIM6

13 Security Considerations

There are no security considerations.

14 IANA Considerations

There are no IANA considerations.

15 References

15.1 Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander,

"SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

15.2 Informative References

[NTPPDM] Ackermann, M., "draft-ackermann-ntp-pdm-ntp-usage-00", Internet Draft, January 2014.

[ELKPDM] Elkins, N., "draft-elkins-6man-ipv6-pdm-dest-option-05", Internet Draft, January 2014.

[IEEE1588] IEEE 1588-2002 standard, "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems"

16 Acknowledgments

The authors would like to thank Al Morton, Brian Trammel, David Boyes, and Rick Troth for their comments and assistance.

Authors' Addresses

Nalini Elkins
Inside Products, Inc.
36A Upper Circle
Carmel Valley, CA 93924
United States
Phone: +1 831 659 8360
Email: nalini.elkins@insidethestack.com
http://www.insidethestack.com

William Jouris
Inside Products, Inc.
36A Upper Circle
Carmel Valley, CA 93924
United States
Phone: +1 925 855 9512
Email: bill.jouris@insidethestack.com
http://www.insidethestack.com

Michael S. Ackermann
Blue Cross Blue Shield of Michigan
P.O. Box 2888
Detroit, Michigan 48231
United States

Phone: +1 310 460 4080
Email: mackermann@bcbsmi.com
http://www.bcbsmi.com

Keven Haining
US Bank
16900 W Capitol Drive
Brookfield, WI 53005
United States
Phone: +1 262 790 3551
Email: keven.haining@usbank.com
http://www.usbank.com