INTERNET-DRAFT

N. Elkins
Inside Products
R. Hamilton
Chemical Abstracts Service
M. Ackermann
BCBS Michigan

Intended Status: Proposed Standard
Expires: October 2, 2015

March 31, 2015

IPv6 Performance and Diagnostic Metrics (PDM) Destination Option
draft-elkins-ippm-6man-pdm-option-00

Table of Contents

Abstract

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. An implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header as well as the field limits, calculations, and usage of the PDM in measurement are included in this document.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

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

Copyright and License Notice

1 Background

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. An implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header has been proposed in a companion document. This document specifies the layout, field limits, calculations, and usage of the PDM in measurement.

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 or other "middle boxes". This document specifies a new destination option, the Performance and Diagnostic Metrics (PDM) destination option.

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 End User Quality of Service (QoS)

The difference between timing values in the PDM traveling along with the packet will be used to estimate QoS as experienced by an end user device.

For many applications, the key user performance indicator is response time. 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 simple solution to support measurements.

1.3 Need for a 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, in post-collection processing, the diagnostic data corresponding to each packet at different observation points must be matched for proper measurements. A sequence number in each packet provides sufficient basis for the matching process. If need be, the timing fields may be used along with the sequence number to ensure uniqueness.

This method of data collection along the path is of special use to determine where packet loss or packet corruption is happening.

The packet sequence number needs to be unique in the context of the session (5-tuple). See section 2 for a definition of 5-tuple.

1.4 Rationale for proposed solution

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

Advantages include:

1. Real measure of actual transactions.
2. Independence from transport layer protocols.
3. Ability to span organizational boundaries with consistent instrumentation
4. No time synchronization needed between session partners

The PDM provides the ability to quickly determine if the (latency) problem is in the network or in the server (application). More intermediate measurements may be needed if the host or network discrimination is not sufficient. At the client, TCP/IP stack time vs. applications time may still need to be broken out by client software.

1.5 PDM Works in Collaboration with Other Headers

The purpose of the PDM is not to supplant all the variables present in all other headers but to provide data which is not available or very difficult to get. The way PDM would be used is by a technician (or tool) looking at a packet capture. Within the packet capture, they would have available to them the layer 2 header, IP header (v6 or v4), TCP, UCP, ICMP, SCTP or other headers. All information would be looked at together to make sense of the packet flow. The technician or processing tool could analyze, report or ignore the data from PDM, as necessary.

For an example of how PDM can help with TCP retransmit problems, please look at section 8.

2 Measurement Information Derived from PDM

Each packet contains information about the sender and receiver. In IP protocol, the identifying information is called a "5-tuple".

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.)
```

The PDM contains the following base fields:

```
PSNTP    : Packet Sequence Number This Packet
PSNLR    : Packet Sequence Number Last Received
DELTATLR : Delta Time Last Received
DELTATLS : Delta Time Last Sent
```

Other fields for scaling and time base are also in the PDM and will be described in section 3.

This information, combined with the 5-tuple, allows the measurement of the following metrics:

1. Round-trip delay
2. Server delay

2.1 Round-Trip Delay

Round-trip *Network* delay is the delay for packet transfer from a

source host to a destination host and then back to the source host. This measurement has been defined, and the advantages and disadvantages discussed in "A Round-trip Delay Metric for IPPM" [RFC2681].

2.2 Server Delay

Server delay is the interval between when a packet is received by a device and the first corresponding 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. The stack delay may be related to network performance. If this aggregate time is seen as a problem, and there is a need to make a clear distinction between application processing time and stack delay, including that caused by the network, then more client based measurements are needed.

3 Performance and Diagnostic Metrics Destination Option Layout

3.1 Destination Options Header

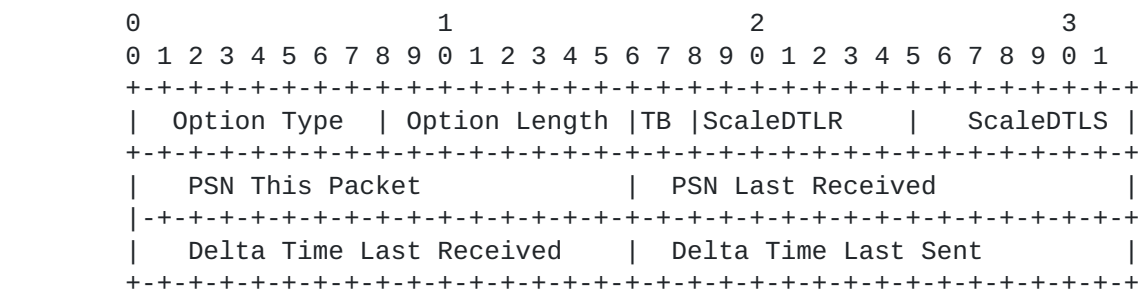
The IPv6 Destination Options Header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options Header is identified by a Next Header value of 60 in the immediately preceding header and is defined in RFC2460 [RFC2460]. The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) is an implementation of the Destination Options Header (Next Header value = 60). The PDM does not require time synchronization.

3.2 Performance and Diagnostic Metrics Destination Option

The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields:

```
TIMEBASE : Base timer unit
SCALEDTLR: Scale for Delta Time Last Received
SCALEDTLS: Scale for Delta Time Last Sent
PSNTP    : Packet Sequence Number This Packet
PSNLR    : Packet Sequence Number Last Received
DELTATLR : Delta Time Last Received
DELTATLS : Delta Time Last Sent
```

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



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 16.

Time Base

2-bit unsigned integer. It will indicate the lowest granularity possible for this device. That is, for a value of 00 in the Time Base field, a value of 1 in the DELTA fields indicates 1 microsecond.

This field is being included so that a device may choose the granularity which most suits its timer ticks. That is, so that it does not have to do more work than needed to convert values required for the PDM.

The possible values of Time Base are as follows:

00 - milliseconds
01 - microseconds
10 - nanoseconds
11 - picoseconds

Scale Delta Time Last Received (SCALEDTLR)

7-bit signed integer. This is the scaling value for the Delta Time Last Received (DELTATLR) field. The possible values are from -128 to +127. See Section 4 for further discussion on Timing Considerations and formatting of the scaling values.

Scale Delta Time Last Sent (SCALEDTLS)

7-bit signed integer. This is the scaling value for the Delta Time Last Sent (DELTATLS) field. The possible values are from -128 to +127.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for human use. That is, while to be used while analyzing packet traces.

Initialized at a random number and monotonically incremented for each 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). The random number initialization is to make it harder to spoof and insert such packets.

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

Packet Sequence Number Last Received (PSNLR)

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

Delta Time Last Received (DELTATLR)

A 16-bit unsigned integer field. The value is according to the scale in SCALEDTLR.

DELTATLR = Send time packet 2 - Receive time packet 1

Delta TimeLast Sent (DELTATLS)

A 16-bit unsigned integer field. The value is according to the scale in SCALEDTLS.

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

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.

3.3 Header Placement

The PDM destination option MUST be placed as follows:

- Before the upper-layer header. That is, this is the last extension header.

This follows the order defined in RFC2460 [RFC2460]

IPv6 header

Hop-by-Hop Options header

Destination Options header

Routing header

Fragment header

Authentication header

Encapsulating Security Payload header

Destination Options header

upper-layer header

For each IPv6 packet header, the PDM MUST NOT appear more than once. However, an encapsulated packet MAY contain a separate PDM associated with each encapsulated IPv6 header.

3.4 Implementation Considerations

The PDM destination options extension header SHOULD be turned on by each stack on a host node. It MAY also be turned on only in case of diagnostics needed for problem resolution.

3.5 Dynamic Configuration Options

If implemented, each operating system MUST have a default configuration parameter, e.g. diag_header_sys_default_value=yes/no. The operating system MAY also have a dynamic configuration option to change the configuration setting as needed.

If the PDM destination options extension header is used, then it MAY be turned on for all packets flowing through the host, applied to an upper-layer protocol (TCP, UDP, SCTP, etc), a local port, or IP address only. These are at the discretion of the implementation.

The PDM MUST NOT be changed dynamically via packet flow as this may create potential security violation or DoS attack by numerous packets turning the header on and off.

As with all other destination options extension headers, the PDM is for destination nodes only. As specified above, intermediate devices MUST neither set nor modify this field.

3.6 5-tuple Aging

Within the operating system, metrics must be kept on a 5-tuple basis.

The 5-tuple is:

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)

The question comes of when to stop keeping data or restarting the numbering for a 5-tuple. For example, in the case of TCP, at some point, the connection will terminate. Keeping data in control blocks forever, will have unfortunate consequences for the operating system.

So, the recommendation is to use a known aging parameter such as Max Segment Lifetime (MSL) as defined in Transmission Control Protocol [RFC0793] to reuse or drop the control block. The choice of aging parameter is left up to the implementation.

4 Considerations of Timing Representation

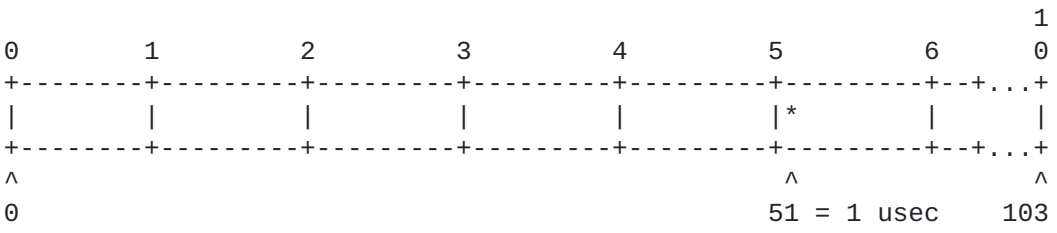
4.1 Encoding the Delta-Time Values

This section makes reference to and expands on the document "Encoding of Time Intervals for the TCP Timestamp Option" [TRAM-TCPM].

4.2 Timer registers are different on different hardware

One of the problems with timestamp recording is the variety of hardware that generates the time value to be used. Different CPUs track the time in registers of different sizes, and the most-frequently-iterated bit could be the first on the left or the first on the right. In order to generate some examples here it is necessary to indicate the type of timer register being used.

As described in the "IBM z/Architecture Principles of Operation" [IBM-POPS], the Time-Of-Day clock in a zSeries CPU is a 104-bit register, where bit 51 is incremented approximately every microsecond:



To represent these values concisely a hexadecimal representation will be used, where each digit represents 4 binary bits. Thus:

```
0000 0000 0000 0001 = 1 timer unit (2**-12 usec, or about 244 psec)
0000 0000 0000 1000 = 1 microsecond
0000 0000 003E 8000 = 1 millisecond
0000 0000 F424 0000 = 1 second
0000 0039 3870 0000 = 1 minute
0000 0D69 3A40 0000 = 1 hour
0001 41DD 7600 0000 = 1 day
```

Note that only the first 64 bits of the register are commonly represented, as that represents a count of timer units on this hardware. Commonly the first 52 bits are all that are displayed, as that represents a count of microseconds.

4.3 Timer Units on Other Systems

This encoding method works the same with other hardware clock formats. The method uses a microsecond as the basic value and allows for large time differentials.

4.4 Time Base

This specification allows for the fact that different CPU TOD clocks use different binary points. For some clocks, a value of 1 could indicate 1 microsecond, whereas other clocks could use the value 1 to indicate 1 millisecond. In the former case, the binary digits to the right of that binary point measure 2**(-n) microseconds, and in the latter case, 2**(-n) milliseconds.

The Time Base allows us to ensure we have a common reference, at the very least, common knowledge of what the binary point is for the transmitted values.

We propose a base unit for the time. This is a 2-bit integer indicating the lowest granularity possible for this device. That is, for a value of 00 in the Time Base field, a value of 1 in the DELTA fields indicates 1 picosecond.

The possible values of Time Base are as follows:

```
00 - milliseconds
01 - microseconds
10 - nanoseconds
11 - picoseconds
```

Time base is not necessarily equivalent to length of one timer tick. That is, on many, if not all, systems, the timer tick value will not be in complete units of nanoseconds, milliseconds, etc. For example, on an IBM zSeries machine, one timer tick (or clock unit) is 2 to the -12th microseconds.

Therefore, some amount of conversion may be needed to approximate Time Base units.

4.5 Timer-value scaling

As discussed in [TRAM-TCPM] we propose storing not an entire time-interval value, but just the most significant bits of that value, along with a scaling factor to indicate the magnitude of the time-interval value. In our case, we will use the high-order 16 bits. The scaling value will be the number of bits in the timer register to the right of the 16th significant bit. That is, if the timer register contains this binary value:

```
          1110100011010100101001010001000000000000
          <-16 bits     -><-24 bits             ->
```

then, the values stored would be 1110 1000 1101 0100 in binary (E8D4 hexadecimal) for the time value and 24 for the scaling value. Note that the displayed value is the binary equivalent of 1 second expressed in picoseconds.

The below table represents a device which has a TimeBase of picosecond (or 00). The smallest and simplest value to represent is 1 picosecond; the time value stored is 1, and the scaling value is 0. Using values from the table below, we have:

Delta time	Time value in picoseconds	Encoded value	Scaling decimal
1 picosecond	1	1	0
1 nanosecond	3E8	3E8	0
1 microsecond	F4240	F424	4
1 millisecond	3B9ACA00	3B9A	16
1 second	E8D4A51000	E8D4	24
1 minute	3691D6AFC000	3691	32
1 hour	cca2e51310000	CCA2	36
1 day	132f4579c980000	132F	44
365 days	1b5a660ea44b80000	1B5A	52

Sample binary values (high order 16 bits taken)

```
1 psec           1                                                0001
1 nsec         3E8                                    0011 1110 1000
1 usec       F4240                          1111 0100 0010 0100 0000
1 msec    3B9ACA00           0011 1011 1001 1010 1100 1010 0000 0000
1 sec    E8D4A51000 1110 1000 1101 0100 1010 0101 0001 0000 0000 0000
```

4.6 Limitations with this encoding method

If we follow the specification in [TRAM-TCPM], the size of one of these time-interval fields is limited to this 11-bit value and five-bit scale, so that they fit into a 16-bit space. With that limitation, the maximum value that could be stored in 16 bits is:

```
    11-bit value   Scale
    =============  ======
    1111 1111 111  1 1111
```

or an encoded value of 3FF and a scale value of 31. This value corresponds to any time differential between:

```
                |<Count of zeroes is the Scale value>|
11 1111 1111 1000 0000 0000 0000 0000 0000 0000 0000  (binary)
3  F    F    8    0    0    0    0    0    0    0     (hexadecimal)
```

and

```
11 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111  (binary)
3  F    F    F    F    F    F    F    F    F    F     (hexadecimal)
```

This time value, 3FFFFFFFFFF, converts to 50 days, 21 hours, 40 minutes and 46.511103 seconds. A time differential 1 microsecond longer won't fit into 16 bits using this encoding method.

4.7 Lack of precision induced by timer value truncation

When the bit values following the first 11 significant bits are truncated, obviously loss of precision in the value. The range of values that will be truncated to the same encoded value is 2**(Scale)-1 microseconds.

The smallest time differential value that will be truncated is

1000 0000 0000 = 2.048 msec

The value

1000 0000 0001 = 2.049 msec

will be truncated to the same encoded value, which is 400 in hex, with a scale value of 1. With the scale value of 1, the value range is calculated as 2**1 - 1, or 1 usec, which you can see is the difference between these minimum and maximum values.

With that in mind, let's look at that table of delta time values again, where the Precision is the range from the smallest value corresponding to this encoded value to the largest:

Delta time	Time value in microseconds	Encoded value	Scale	Precision
1 microsecond	1	1	0	0:00.000000
1 millisecond	38E	38E	0	0:00.000000
1 second	F4240	7A1	9	0:00.000511
1 minute	3938700	727	15	0:00.032767
1 hour	D693A400	6B4	21	0:02.097151
1 day	141DD76000	507	26	1:07.108863
Maximum value	3FFFFFFFFF	7FF	31	35:47.483647

So, when measuring the delay between transmission of two packets, or between the reception of two packets, any delay shorter than 50 days 21 hours and change can be stored in this encoded fashion within 16 bits. When you encode, for example, a DTN response time delay of 50 days, 21 hours and 40 minutes, you can be assured of accuracy within 35 minutes.

5 PDM Flow - Simple Client Server

Following is a sample simple flow for the PDM with one packet sent from Host A and one packet received by Host B. The PDM does not require time synchronization between Host A and Host B. The calculations to derive meaningful metrics for network diagnostics are 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)

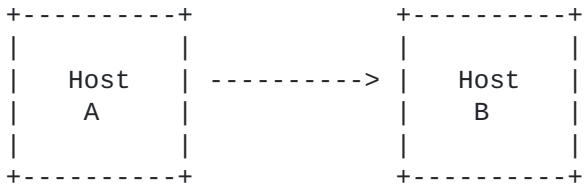
It should be understood that the packet identification information is in each packet. We will not repeat that in each of the following steps.

5.1 Step 1

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

The time and packet sequence number are saved by the sender internally. The packet sequence number and delta times are sent in the packet.

Packet 1



PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:     25
PSNLR    : Packet Sequence Number Last Received:   -
DELTATLR : Delta Time Last Received:               -
SCALEDTLR: Scale of Delta Time Last Received:      0
DELTATLS : Delta Time Last Sent:                   -
SCALEDTLS: Scale of Delta Time Last Sent:          0
TIMEBASE : Granularity of Time:                   00 (Milliseconds)
```

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

```
Packet Sequence Number of the last packet sent:     25
Time the last packet was sent:                 10:00:00
```

Note, the initial PSNTP from Host A starts at a random number. In

this case, 25. The time in these examples is shown in seconds for the sake of simplicity.

5.2 Step 2

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

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

```
Packet Sequence Number of the last packet received:    25
Time the last packet was received                 :    11:00:03
```

Note, this timestamp is in Host B time. It has nothing whatsoever to do with Host A time. The Packet Sequence Number of the last packet received will become PSNLR which will be sent out in the packet sent by Host B in the next step. The time last received will be used to calculate the DELTALR value to be sent out in the packet sent by Host B in the next step.

5.3 Step 3

Packet 2 is sent by Host B to Host A. Note, the initial packet sequence number (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 Time Last Received

That is:

DELTATLR = 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:

```
Packet Sequence Number of the last packet received:     25
Time the last packet was received:                      11:00:03
Packet Sequence Number of this packet:                  12
```

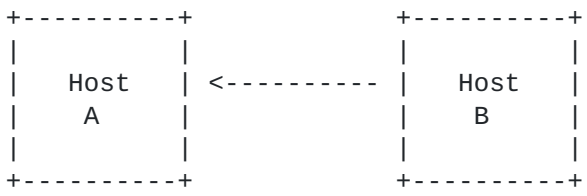
Time this packet is being sent: 11:00:07

We can now calculate a delta value to be sent out in the packet. DELTATLR becomes:

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

This is the derived metric: Server Delay. The time and scaling factor must be calculated. Then, this value, along with the packet sequence numbers will be sent to Host A as follows:

Packet 2



PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:    12
PSNLR    : Packet Sequence Number Last Received:  25
DELTATLR : Delta Time Last Received:            3A35 (4 seconds)
SCALEDTLR: Scale of Delta Time Last Received:     25
DELTATLS : Delta Time Last Sent:                   -
SCALEDTLS: Scale of Delta Time Last Sent:          0
TIMEBASE : Granularity of Time:                   00 (Milliseconds)
```

The metric left to be calculated is the Round-Trip Delay. This will be calculated by Host A when it receives Packet 2.

5.4 Step 4

Packet 2 is received at Host A. Remember, its time is set to one hour earlier than Host B. Internally, it must note:

```
Packet Sequence Number of the last packet received:    12
Time the last packet was received                :    10:00:12
```

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

So, now, Host A can calculate total end-to-end time. That is:

End-to-End Time = 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 time = 10:00:12 - 10:00:00 or 12 (Server and Network RT delay combined). This time may also be called total Overall Round-trip time (which includes Network RTT and Host Response Time).

This derived metric we will call DELTATLS or Delta Time Last Sent.

We can now also calculate round trip delay. The formula is:

Round trip delay = DELTATLS - DELTATLR

Or:

Round trip delay = 12 - 4 or 8

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

Note: this simple example assumes one send and one receive. That is done only for purposes of explaining the function of the PDM. In cases where there are multiple packets returned, one would take the time in the last packet in the sequence. The calculations of such timings and intelligent processing is the function of post-processing of the data.

5.5 Step 5

Packet 3 is sent from Host A to Host B.

PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:    26
PSNLR    : Packet Sequence Number Last Received:  12
DELTATLR : Delta Time Last Received:               0
SCALEDTLS: Scale of Delta Time Last Received       0
DELTATLS : Delta Time Last Sent:               105e (12 seconds)
SCALEDTLR: Scale of Delta Time Last Received:     26
TIMEBASE : Granularity of Time:                   00 (Milliseconds)
```

To calculate Two-Way Delay, any packet capture device may look at these packets and do what is necessary.

6 Other Flows

What we have discussed so far is a simple flow with one packet sent and one returned. Let's look at how PDM may be useful in other types of flows.

6.1 PDM Flow - One Way Traffic

The flow on a particular session may not be a send-receive paradigm. Let us consider some other situations. In the case of a one-way flow, one might see the following:

Packet	Sender	PSN This Packet	PSN Last Recvd	Delta Time Last Recvd	Delta Time Last Sent
1	Server	1	0	0	0
2	Server	2	0	0	5
3	Server	3	0	0	12
4	Server	4	0	0	20

What does this mean and how is it useful?

In a one-way flow, only the Delta Time Last Sent will be seen as used. Recall, Delta Time Last Sent is the difference between the send of one packet from a device and the next. This is a measure of throughput for the sender - according to the sender's point of view. That is, it is a measure of how fast is the application itself (with stack time included) able to send packets.

How might this be useful? If one is having a performance issue at the client and sees that packet 2, for example, is sent after 5 microseconds from the server but takes 3 minutes to arrive at the destination, then one may safely conclude that there are delays in the path other than at the server which may be causing the delivery issue of that packet. Such delays may include the network links, middle-boxes, etc.

Now, true one-way traffic is quite rare. What people often mean by "one-way" traffic is an application such as FTP where a group of packets (for example, a TCP window size worth) is sent, then the sender waits for acknowledgment. This type of flow would actually fall into the "multiple-send" traffic model.

6.2 PDM Flow - Multiple Send Traffic

Assume that two packets are sent for each ACK from the server.

Packet	Sender	PSN This Packet	PSN Last Recvd	Delta Time Last Recvd	Delta Time Last Sent
1	Server	1	0	0	0
2	Server	2	0	0	5
3	Client	1	2	20	0
4	Server	3	1	10	15

How might this be used?

Notice that in packet 3, the client has a value of Delta Time Last received of 20. Recall that Delta Time Last Received is the Send time of packet 3 - receive time of packet 2. So, what does one know now? In this case, Delta Time Last Received is the processing time for the Client to send the next packet.

How to interpret this depends on what is actually being sent. Remember, PDM is not being used in isolation, but to supplement the fields found in other headers. Let's take some examples:

1. Client is sending a standalone TCP ACK. One would find this by looking at the payload length in the IPv6 header and the TCP Acknowledgement field in the TCP header. So, in this case, the client is taking 20 units to send back the ACK. This may or may not be interesting.

2. Client is sending data with the packet. Again, one would find this by looking at the payload length in the IPv6 header and the TCP Acknowledgement field in the TCP header. So, in this case, the client is taking 20 units to send back data. This may represent "User Think Time". Again, this may or may not be interesting, in isolation. But, if there is a performance problem receiving data at the server, then taken in conjunction with RTT or other packet timing information, this information may be quite interesting.

Of course, one also needs to look at the PSN Last Received field to make sure of the interpretation of this data. That is, to make sure that the Delta Last Received corresponds to the packet of interest.

The benefits of PDM are that we have such information available in a uniform manner for all applications and all protocols without extensive changes required to applications.

6.3 PDM Flow - Multiple Send with Errors

One might wonder if all of the functions of PDM might be better suited to TCP or a TCP option. Let us take the case of how PDM may help in a case of TCP retransmissions in a way that TCP options or TCP ACK / SEQ would not.

Assume that three packets are sent with each send from the server.

From the server, this is what is seen.

Pkt	Sender	PSN This Pkt	PSN LastRecvd	Delta Time LastRecvd	Delta Time LastSent	TCP SEQ	Data Bytes
1	Server	1	0	0	0	123	100
2	Server	2	0	0	5	223	100
3	Server	3	0	0	5	333	100

The client however, does not get all the packets. From the client, this is what is seen for the packets sent from the server.

Pkt	Sender	PSN This Pkt	PSN LastRecvd	Delta Time LastRecvd	Delta Time LastSent	TCP SEQ	Data Bytes
1	Server	1	0	0	0	123	100
2	Server	3	0	0	5	333	100

Let's assume that the server now retransmits the packet. (Obviously, a duplicate acknowledgment sequence for fast retransmit or a retransmit timeout would occur. To illustrate the point, these packets are being left out.)

So, then if a TCP retransmission is done, then from the client, this is what is seen for the packets sent from the server.

Pkt	Sender	PSN This Pkt	PSN LastRecvd	Delta Time LastRecvd	Delta Time LastSent	TCP SEQ	Data Bytes
1	Server	4	0	0	30	223	100

The server has resent the old packet 2 with TCP sequence number of 223. The retransmitted packet now has a PSN This Packet value of 4. The Delta Last Sent is 30 - the time between sending the packet with PSN of 3 and this current packet.

Let's say that packet 4 STILL does not make it. Then, after some amount of time (RTO) then the packet with TCP sequence number of 223

is resent.

From the client, this is what is seen for the packets sent from the server.

Pkt	Sender	PSN This Pkt	PSN LastRecvd	Delta Time LastRecvd	Delta Time LastSent	TCP SEQ	Data Bytes
1	Server	5	0	0	60	223	100

If now, this packet makes it, one has a very good idea that packets exist which are being sent from the server as retransmissions and not making it to the client. This is because the PSN of the resent packet from the server is 5 rather than 4. If we had used TCP sequence number alone, we would never have seen this situation. Because the TCP sequence number in all situations is 223.

This situation would be experienced by the user of the application (the human being actually sitting somewhere) as a "hangs" or long delay between packets. On large networks, to diagnose problems such as these where packets are lost somewhere on the network, one has to take multiple traces to find out exactly where.

The first thing is to start with doing a trace at the client and the server. So, we can see if the server sent a particular packet and the client received it. If the client did not receive it, then we start tracking back to trace points at the router right after the server and the router right before the client. Did they get these packets which the server has sent? This is a time consuming activity.

With PDM, we can speed up the diagnostic time because we may be able to use only the trace taken at the client to see what the server is sending.

7 Potential Overhead Considerations

Questions have been posed as to the potential overhead of PDM. First, PDM is entirely optional. That is, a site may choose to implement PDM or not as they wish. If they are happy with the costs of PDM vs. the benefits, then the choice should be theirs.

Below is a table outlining the potential overhead in terms of additional time to deliver the response to the end user for various assumed RTTs.

Packet Bytes in Packet	RTT	Bytes Per Milli	Bytes in PDM	New RTT	Overhead
1000	1000 milli	1	16	1016.000	16.000 milli
1000	100 milli	10	16	101.600	1.600 milli
1000	10 milli	100	16	10.160	.160 milli
1000	1 milli	1000	16	1.016	.016 milli

Below are some examples of actual RTTs for packets traversing large enterprise networks. The first example is for packets going to multiple business partners.

Packet Bytes in Packet	RTT	Bytes Per Milli	Bytes in PDM	New RTT	Overhead
1000	17 milli	58	16	17.360	.360 milli

The second example is for packets at a large enterprise customer within a data center. Notice that the scale is now in microseconds rather than milliseconds.

Packet Bytes in Packet	RTT	Bytes Per Micro	Bytes in PDM	New RTT	Overhead
1000	20 micro	50	16	20.320	.320 micro

8 Security Considerations

The PDM MUST NOT be changed dynamically via packet flow as this creates a possibility for potential security violations or DoS attacks by numerous packets turning the header on and off.

Attackers may also send many packets from multiple ports, for example by doing a port scan. This will cause the stack to create many control blocks. This is the same problem as seen for SYN flood attacks. Similar protections should be implemented by the stack to preserve the integrity of memory.

9 IANA Considerations

Option Type to be assigned by IANA [RFC2780].

10 References

10.1 Normative References

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

[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.

[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.

10.2 Informative References

[TRAM-TCPM] Trammel, B., "Encoding of Time Intervals for the TCP Timestamp Option-01", Internet Draft, July 2013. [Work in Progress]

[IBM-POPS] IBM Corporation, "IBM z/Architecture Principles of Operation", SA22-7832, 1990-2012

11 Acknowledgments

The authors would like to thank Keven Haining, Al Morton, Brian Trammel, David Boyes, Bill Jouris, Richard Scheffenegger, 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

Robert Hamilton
Chemical Abstracts Service
A Division of the American Chemical Society
2540 Olentangy River Road
Columbus, Ohio 43202
United States
Phone: +1 614 447 3600 x2517
Email: rhamilton@cas.org
http://www.cas.org

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