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

Intended Status: Proposed Standard
Expires: November 10, 2017
May 9, 2017

IPv6 Performance and Diagnostic Metrics (PDM) Destination Option
draft-ietf-ippm-6man-pdm-option-10

Abstract

To assess performance problems, this document describes optional headers embedded in each packet that provide sequence numbers and timing information as a basis for measurements. 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

Copyright and License Notice

Table of Contents

1 Background 5
1.1 Terminology 5
1.2 Rationale for defined solution 5
1.3 IPv6 Transition Technologies 6
2 Measurement Information Derived from PDM 6
2.1 Round-Trip Delay 6
2.2 Server Delay 7
3 Performance and Diagnostic Metrics Destination Option Layout . . 7
3.1 Destination Options Header 7
3.2 Performance and Diagnostic Metrics Destination Option . . . 7
3.2.1 PDM Layout 7
3.2.2 Base Unit for Time Measurement 9
3.3 Header Placement 10
3.4 Header Placement Using IPSec ESP Mode 10
3.4.1 Using ESP Transport Mode 10
3.4.2 Using ESP Tunnel Mode 10
3.5 Implementation Considerations 11
3.5.1 PDM Activation 11
3.5.2 PDM Timestamps 11
3.6 Dynamic Configuration Options 11
3.7 Information Access and Storage 11
4 Security Considerations 12
4.1 Resource Consumption and Resource Consumption Attacks . . . 12
4.2 Pervasive monitoring 12
4.3 PDM as a Covert Channel 13
4.4 Timing Attacks 13
5 IANA Considerations 14
6 References 14
6.1 Normative References 14
6.2 Informative References 15
Appendix A: Context for PDM 15
A.1 End User Quality of Service (QoS) 15
A.2 Need for a Packet Sequence Number (PSN) 15
A.3 Rationale for Defined Solution 16
A.4 Use PDM with Other Headers 16
Appendix B : Timing Considerations 17
B.1 Timing Differential Calculations 17
B.2 Considerations of this time-differential representation . . 18
B.2.1 Limitations with this encoding method 18
B.2.2 Loss of precision induced by timer value truncation . . 19
Appendix C: Sample Packet Flows 20
C.1 PDM Flow - Simple Client Server 20
C.1.1 Step 1 21
C.1.2 Step 2 21
C.1.3 Step 3 22
C.1.4 Step 4 23

      C.1.5 Step 5 24
    C.2 Other Flows 24
      C.2.1 PDM Flow - One Way Traffic 24
      C.2.2 PDM Flow - Multiple Send Traffic 26
      C.2.3 PDM Flow - Multiple Send with Errors 27
  Appendix D: Potential Overhead Considerations 28
  Acknowledgments 29
  Authors' Addresses 30

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.

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. This document specifies the layout, field limits, calculations, and usage of the PDM in measurement.

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 Rationale for defined solution

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

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

5. Ability to handle all transport protocols (TCP, UDP, SCTP, etc) in a uniform way

The PDM provides the ability to determine quickly if the (latency) problem is in the network or in the server (application). That is, it is a fast way to do triage. For more information on background and usage of PDM, see Appendix A.

1.3 IPv6 Transition Technologies

In the path to full implementation of IPv6, transition technologies such as translation or tunneling may be employed. The PDM header is not expected to work in such scenarios. It is likely that an IPv6 packet containing PDM will be dropped if using IPv6 transition technologies.

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 storing time scaling factors 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 needs to 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. The PDM does not require time synchronization.

3.2 Performance and Diagnostic Metrics Destination Option

3.2.1 PDM Layout

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

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:

```
    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 |    ScaleDTLR  |     ScaleDTLS |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   PSN This Packet             |  PSN Last Received            |
   |-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Delta Time Last Received    |  Delta Time Last Sent         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

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

In keeping with RFC2460[RFC2460], the two high 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.

The third high 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 high order bit MUST be 0.

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.

Scale Delta Time Last Received (SCALEDTLR)

8-bit unsigned integer. This is the scaling value for the Delta Time Last Received (DELTATLR) field. The possible values are from 0-255. See Section 4 for further discussion on Timing Considerations and formatting of the scaling values.

Scale Delta Time Last Sent (SCALEDTLS)

8-bit signed integer. This is the scaling value for the Delta Time Last Sent (DELTATLS) field. The possible values are from 0 to 255.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for use while analyzing packet traces.

Initialized at a random number and incremented monotonically for each packet of the session flow of the 5-tuple. The random number initialization is intended to make it harder to spoof and insert such packets.

Operating systems MUST implement a separate packet sequence number counter per 5-tuple.

Packet Sequence Number Last Received (PSNLR)

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

This field is initialized to 0.

Delta Time Last Received (DELTATLR)

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

Delta Time Last Received = (Send time packet n - Receive time packet n-1)

Delta Time Last Sent (DELTATLS)

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

Delta Time Last Sent = (Receive time packet n - Send time packet n-1)

3.2.2 Base Unit for Time Measurement

A time differential is always a whole number in a CPU; it is the unit specification -- hours, seconds, nanoseconds -- that determine what the numeric value means. For PDM, the base time unit is 1 attosecond (asec). This allows for a common unit and scaling of the time differential among all IP stacks and hardware implementations.

Note that PDM provides the ability to measure both time differentials that are extremely small, and time differentials in a DTN-type environment where the delays may be very great. To store a time

differential in just 16 bits that must range in this way will require some scaling of the time differential value.

One issue is the conversion from the native time base in the CPU hardware of whatever device is in use to some number of attoseconds. It might seem this will be an astronomical number, but the conversion is straightforward. It involves multiplication by an appropriate power of 10 to change the value into a number of attoseconds. Then, to scale the value so that it fits into DELTATLR or DELTATLS, the value is shifted by of a number of bits, retaining the 16 high-order or most significant bits. The number of bits shifted becomes the scaling factor, stored as SCALEDTLR or SCALEDTLS, respectively. For a full description of this process, including examples, please see Appendix A.

3.3 Header Placement

The PDM Destination Option is placed as defined in RFC2460 [RFC2460]. There may be a choice of where to place the Destination Options header. If using ESP mode, please see section 3.4 of this document for placement of the PDM Destination Options 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 Header Placement Using IPSec ESP Mode

IPSec Encapsulating Security Payload (ESP) is defined in [RFC4303] and is widely used. Section 3.1.1 of [RFC4303] discusses placement of Destination Options Headers.

The placement of PDM is different depending on if ESP is used in tunnel or transport mode.

3.4.1 Using ESP Transport Mode

Note that Destination Options MAY be placed before or after ESP or both. If using PDM in ESP transport mode, PDM MUST be placed after the ESP header so as not to leak information.

3.4.2 Using ESP Tunnel Mode

Note that Destination Options MAY be placed before or after ESP or both in both the outer set of IP headers and the inner set of IP headers. A tunnel endpoint that creates a new packet may decide to use PDM independent of the use of PDM of the original packet to enable delay measurements between the two tunnel endpoints

3.5 Implementation Considerations

3.5.1 PDM Activation

An implementation should provide an interface to enable or disable the use of PDM. This specification recommends having PDM off by default.

PDM MUST NOT be turned on merely if a packet is received with a PDM header. The received packet could be spoofed by another device.

3.5.2 PDM Timestamps

The PDM timestamps are intended to isolate wire time from server or host time, but may necessarily attribute some host processing time to network latency.

RFC2330 [RFC2330] "Framework for IP Performance Metrics" describes two notions of wire time in section 10.2. These notions are only defined in terms of an Internet host H observing an Internet link L at a particular location:

+   For a given IP packet P, the 'wire arrival time' of P at H on L is the first time T at which any bit of P has appeared at H's observational position on L.

+   For a given IP packet P, the 'wire exit time' of P at H on L is the first time T at which all the bits of P have appeared at H's observational position on L.

This specification does not define the exact H's observing position on L. That is left for the deployment setups to define. However, the position where PDM timestamps are taken SHOULD be as close to the physical network interface as possible. Not all implementations will be able to achieve the ideal level of measurement.

3.6 Dynamic Configuration Options

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.

3.7 Information Access and Storage

Measurement information provided by PDM may be made accessible for higher layers or the user itself. Similar to activating the use of

PDM, the implementation may also provide an interface to indicate if received

PDM information may be stored, if desired. If a packet with PDM information is received and the information should be stored, the upper layers may be notified. Furthermore, the implementation should define a configurable maximum lifetime after which the information can be removed as well as a configurable maximum amount of memory that should be allocated for PDM information.

4 Security Considerations

PDM may introduce some new security weaknesses.

4.1 Resource Consumption and Resource Consumption Attacks

PDM needs to calculate the deltas for time and keep track of the sequence numbers. This means that control blocks which reside in memory may be kept at the end hosts per 5-tuple.

A limit on how much memory is being used SHOULD be implemented.

Without a memory limit, any time a control block is kept in memory, an attacker can try to mis-use the control blocks to cause excessive resource consumption. This could be used to compromise the end host.

PDM is used only at the end hosts and memory is used only at the end host and not at routers or middle boxes.

4.2 Pervasive monitoring

Since PDM passes in the clear, a concern arises as to whether the data can be used to fingerprint the system or somehow obtain information about the contents of the payload.

Let us discuss fingerprinting of the end host first. It is possible that seeing the pattern of deltas or the absolute values could give some information as to the speed of the end host - that is, if it is a very fast system or an older, slow device. This may be useful to the attacker. However, if the attacker has access to PDM, the attacker also has access to the entire packet and could make such a deduction based merely on the time frames elapsed between packets WITHOUT PDM.

As far as deducing the content of the payload, it appears to us that PDM is quite unhelpful in this regard.

4.3 PDM as a Covert Channel

PDM provides a set of fields in the packet which could be used to leak data. But, there is no real reason to suspect that PDM would be chosen rather than another part of the payload or another Extension Header.

A firewall or another device could sanity check the fields within the PDM but randomly assigned sequence numbers and delta times might be expected to vary widely. The biggest problem though is how an attacker would get access to PDM in the first place to leak data. The attacker would have to either compromise the end host or have Man in the Middle (MitM). It is possible that either one could change the fields. But, then the other end host would get sequence numbers and deltas that don't make any sense.

It is conceivable that someone could compromise an end host and make it start sending packets with PDM without the knowledge of the host. But, again, the bigger problem is the compromise of the end host. Once that is done, the attacker probably has better ways to leak data.

Having said that, if a PDM aware middle box or an implementation detects some number of "nonsensical" sequence numbers it could take action to block (or alert on) this traffic.

4.4 Timing Attacks

The fact that PDM can help in the separation of node processing time from network latency brings value to performance monitoring. Yet, it is this very characteristic of PDM which may be misused to make certain new type of timing attacks against protocols and implementations possible.

Depending on the nature of the cryptographic protocol used, it may be possible to leak the long term credentials of the device. For example, if an attacker is able to create an attack which causes the enterprise to turn on PDM to diagnose the attack, then the attacker might use PDM during that debugging time to launch a timing attack against the long term keying material used by the cryptographic protocol.

An implementation may want to be sure that PDM is enabled only for certain ip addresses, or only for some ports. Additionally, the implementation SHOULD require an explicit restart of monitoring after a certain time period (for example for 1 hour), to make sure that PDM is not accidentally left on after debugging has been done etc.

Even so, if using PDM, a user "Consent to be Measured" SHOULD be a pre-requisite for using PDM. Consent is common in enterprises and with some subscription services. The actual content of "Consent to be Measured" will differ by site but it SHOULD make clear that the traffic is being measured for quality of service and to assist in diagnostics as well as to make clear that there may be potential risks of certain vulnerabilities if the traffic is captured during a diagnostic session

5 IANA Considerations

This draft requests an Option Type assignment in the Destination Options and Hop-by-Hop Options sub-registry of Internet Protocol Version 6 (IPv6) Parameters [ref to RFCs and URL below].

http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2

Hex Value	Binary Value act chg rest	Description	Reference
TBD	TBD	Performance and Diagnostic Metrics (PDM)	[This draft]

6 References

6.1 Normative References

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.

[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.

[RFC4303] Kent, S, "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

6.2 Informative References

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

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

Appendix A: Context for PDM

A.1 End User Quality of Service (QoS)

The timing values in the PDM embedded in 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.

Low, reliable and acceptable response times are not just "nice to have". On many networks, the impact can be financial hardship or can endanger human life. In some cities, the emergency police contact system operates over IP; law enforcement, at all levels, use 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 response time measurements.

A.2 Need for a Packet Sequence Number (PSN)

While performing network diagnostics of an end-to-end connection, it often becomes necessary to isolate the factors along the network path responsible for 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).

A.3 Rationale for Defined Solution

One of the important functions of PDM is to allow you to do quickly dispatch the right set of diagnosticians. Within network or server latency, there may be many components. The job of the diagnostician is to rule each one out until the culprit is found.

How PDM fits into this diagnostic picture is that PDM will quickly tell you how to escalate. PDM will point to either the network area or the server area. Within the server latency, PDM does not tell you if the bottleneck is in the IP stack or the application or buffer allocation. Within the network latency, PDM does not tell you which of the network segments or middle boxes is at fault.

What PDM does tell you is whether the problem is in the network or the server.

A.4 Use PDM with Other Headers

For diagnostics, one my want to use PDM with other headers (L2, L3, etc). For example, if PDM is 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 Appendix C.

Appendix B : Timing Considerations

B.1 Timing Differential Calculations

The time counter in a CPU is a binary whole number, representing a number of milliseconds (msec), microseconds (usec) or even picoseconds (psec). Representing one of these values as attoseconds (asec) means multiplying by 10 raised to some exponent. Refer to this table of equalities:

Base value	= # of sec	= # of asec	1000s of asec
1 second	1 sec	10**18 asec	1000**6 asec
1 millisecond	10**-3 sec	10**15 asec	1000**5 asec
1 microsecond	10**-6 sec	10**12 asec	1000**4 asec
1 nanosecond	10**-9 sec	10**9 asec	1000**3 asec
1 picosecond	10**-12 sec	10**6 asec	1000**2 asec
1 femtosecond	10**-15 sec	10**3 asec	1000**1 asec

For example, if you have a time differential expressed in microseconds, since each microsecond is 10**12 asec, you would multiply your time value by 10**12 to obtain the number of attoseconds. If you time differential is expressed in nanoseconds, you would multiply by 10**9 to get the number of attoseconds.

The result is a binary value that will need to be shortened by a number of bits so it will fit into the 16-bit PDM DELTA field.

The next step is to divide by 2 until the value is contained in just 16 significant bits. The exponent of the value in the last column of of the table is useful here; the initial scaling factor is that exponent multiplied by 10. This is the minimum number of low-order bits to be shifted-out or discarded. It represents dividing the time value by 1024 raised to that exponent.

The resulting value may still be too large to fit into 16 bits, but can be normalized by shifting out more bits (dividing by 2) until the value fits into the 16-bit DELTA field. The number of extra bits shifted out is then added to the scaling factor. The scaling factor, the total number of low-order bits dropped, is the SCALEDTL value.

For example: say an application has these start and finish timer values (hexadecimal values, in microseconds):

```
Finish:      27C849234 usec    (02:57:58.997556)
-Start:      27C83F696 usec    (02:57:58.957718)
==========   =========         ===============
Difference   9B9E usec         00.039838 sec or 39838 usec
```

To convert this differential value to binary attoseconds, multiply the number of microseconds by 10**12. Divide by 1024**4, or simply discard 40 bits from the right. The result is 36232, or 8D88 in hex, with a scaling factor or SCALEDTL value of 40.

For another example, presume the time differential is larger, say 32.311072 seconds, which is 32311072 usec. Each microsecond is 10**12 asec, so multiply by 10**12, giving the hexadecimal value 1C067FCCAE8120000. Using the initial scaling factor of 40, drop the last 10 characters (40 bits) from that string, giving 1C067FC. This will not fit into a DELTA field, as it is 25 bits long. Shifting the value to the right another 9 bits results in a DELTA value of E033, with a resulting scaling factor of 49.

When the time differential value is a small number, regardless of the time unit, the exponent trick given above is not useful in determining the proper scaling value. For example, if the time differential is 3 seconds and you want to convert that directly, you would follow this path:

```
      3 seconds = 3*10**18 asec (decimal)
                = 29A2241AF62C0000 asec (hexadecimal)
```

If you just truncate the last 60 bits, you end up with a delta value of 2 and a scaling factor of 60, when what you really wanted was a delta value with more significant digits. The most precision with which you can store this value in 16 bits is A688, with a scaling factor of 46.

B.2 Considerations of this time-differential representation

There are a few considerations to be taken into account with this representation of a time differential. The first is whether there are any limitations on the maximum or minimum time differential that can be expressed using method of a delta value and a scaling factor. The second is the amount of imprecision introduced by this method.

B.2.1 Limitations with this encoding method

The DELTATLS and DELTATLR fields store only the 16 most-significant bits of the time differential value. Thus the range, excluding the scaling factor, is from 0 to 65535, or a maximum of 2**16-1.   This method is further described in [TRAM-TCPM].

The actual magnitude of the time differential is determined by the scaling factor. SCALEDTLR and SCALEDTLS are 8-bit unsigned integers, so the scaling factor ranges from 0 to 255. The smallest number that can be represented would have a value of 1 in the delta field and a

value of 0 in the associated scale field. This is the representation for 1 attosecond. Clearly this allows PDM to measure extremely small time differentials.

On the other end of the scale, the maximum delta value is 65535, or FFFF in hexadecimal. If the maximum scale value of 255 is used, the time differential represented is 65535*2**255, which is over 3*10**55 years, essentially, forever. So there appears to be no real limitation to the time differential that can be represented.

B.2.2 Loss of precision induced by timer value truncation

As PDM specifies the DELTATLR and DELTATLS values as 16-bit unsigned integers, any time the precision is greater than those 16 bits, there will be truncation of the trailing bits, with an accompanying loss of precision in the value.

Any time differential value smaller than 65536 asec can be stored exactly in DELTATLR or DELTATLS, because the representation of this value requires at most 16 bits.

Since the time differential values in PDM are measured in attoseconds, the range of values that would be truncated to the same encoded value is 2**(Scale)-1 asec.

For example, the smallest time differential that would be truncated to fit into a delta field is

```
      1 0000 0000 0000 0000 = 65536 asec
```

This value would be encoded as a delta value of 8000 (hexadecimal) with a scaling factor of 1. The value

```
      1 0000 0000 0000 0001 = 65537 asec
```

would also be encoded as a delta value of 8000 with a scaling factor of 1. This actually is the largest value that would be truncated to that same encoded value. When the scale value is 1, the value range is calculated as 2**1 - 1, or 1 asec, which you can see is the difference between these minimum and maximum values.

The scaling factor is defined as the number of low-order bits truncated to reduce the size of the resulting value so it fits into a 16-bit delta field. If, for example, you had to truncate 12 bits, the loss of precision would depend on the bits you truncated. The range of these values would be

```
      0000 0000 0000 = 0 asec
```

```
      to
         1111 1111 1111 = 4095 asec
```

So the minimum loss of precision would be 0 asec, where the delta value exactly represents the time differential, and the maximum loss of precision would be 4095 asec. As stated above, the scaling factor of 12 means the maximum loss of precision is 2**12-1 asec, or 4095 asec.

Compare this loss of precision to the actual time differential. The range of actual time differential values that would incur this loss of precision is from

```
1000 0000 0000 0000 0000 0000 0000 = 2**27 asec or 134217728 asec
      to
1111 1111 1111 1111 1111 1111 1111 = 2**28-1 asec or 268435455 asec
```

Granted, these are small values, but the point is, any value between these two values will have a maximum loss of precision of 4095 asec, or about 0.00305% for the first value, as encoded, and at most 0.001526% for the second. These maximum-loss percentages are consistent for all scaling values.

Appendix C: Sample Packet Flows

C.1 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.

C.1.1 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

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

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
```

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.

C.1.2 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.

C.1.3 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

The result of this calculation is called: Delta Time Last Received (DELTATLR)

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

Now a delta value to be sent out in the packet can be calculated. DELTATLR becomes:

4 seconds = 11:00:07 - 11:00:03 = 3782DACE9D900000 asec

This is the derived metric: Server Delay.   The time and scaling factor must be converted; in this case, the time differential is DE0B, and the scaling factor is 2E, or 46 in decimal. Then, these values, along with the packet sequence numbers will be sent to Host A as follows:

Packet 2

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

PDM Contents:

PSNTP	: Packet Sequence Number This Packet:	12
PSNLR	: Packet Sequence Number Last Received:	25
DELTATLR	: Delta Time Last Received:	DE0B (4 seconds)
SCALEDTLR	: Scale of Delta Time Last Received:	2E (46 decimal)
DELTATLS	: Delta Time Last Sent:	-
SCALEDTLS	: Scale of Delta Time Last Sent:	0

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

C.1.4 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 (RTT) which includes Network RTT and Host Response Time.

This derived metric we will call Delta Time Last Sent (DELTATLS)

Round trip delay can now be calculated. The formula is:

Round trip delay = (Delta Time Last Sent - Delta Time Last Received)

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.

C.1.5 Step 5

Packet 3 is sent from Host A to Host B.

```
                       +----------+            +----------+
                       |          |            |          |
                       |   Host   | ---------> |   Host   |
                       |    A     |            |    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:                 A688 (scaled value)
SCALEDTLR: Scale of Delta Time Last Received:     30 (48 decimal)
```

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

C.2 Other Flows

What has been 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.

C.2.1 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:

Note: The time is expressed in generic units for simplicity. That is, these values do not represent a number of attoseconds, microseconds or any other real units of time.

Packet	Sender	PSN	PSN	Delta Time	Delta Time
		This Packet	Last Recvd	Last Recvd	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 time units from the server but takes 10 times that long 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.

C.2.2 PDM Flow - Multiple Send Traffic

Assume that two packets are sent for each ACK from the server. For example, a TCP flow will do this, per RFC1122 [RFC1122] Section-4.2.3.

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 such information is now available in a uniform manner for all applications and all protocols without

extensive changes required to applications.

C.2.3 PDM Flow - Multiple Send with Errors

Let us now look at a case of how PDM may be able to help in a case of TCP retransmission and add to the information that is sent in the TCP header.

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

From the server, this is what is seen.

Pkt	Sender	PSN	PSN	Delta Time	Delta Time	TCP	Data
		This Pkt	LastRecvd	LastRecvd	LastSent	SEQ	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 receive all the packets. From the client, this is what is seen for the packets sent from the server.

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

If now, this packet arrives at the destination, one has a very good idea that packets exist which are being sent from the server as retransmissions and not arriving at 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. 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.

Appendix D: Potential Overhead Considerations

One might wonder 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.

Bytes	RTT	Bytes	Bytes	New	Overhead
in Packet		Per Millisec	in PDM	RTT	
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.

Bytes	RTT	Bytes	Bytes	New	Overhead
in Packet		Per Millisec	in PDM	RTT	
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.

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

As with other diagnostic tools, such as packet traces, a certain amount of processing time will be required to create and process PDM. Since PDM is lightweight (has only a few variables), we expect the processing time to be minimal.

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. We would also like to thank Tero Kivinen and Jouni Korhonen for their detailed and perceptive reviews.

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 M. 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@bcbsm.com
http://www.bcbsm.com