Design and Implementation of a Flexible Measurement Tool Exploiting IPv6 Features

University Of Liège
Faculty Of Applied Sciences

Promoter *Professor G. LEDUC*
Supervisor *E. TYCHON*

Paganessi François 2nd LINF
May 2007

Acknowledgments

I would like to express my gratitude to my promoter Professor Guy Leduc, and my supervisor Emmanuel Tychon for their time spent to guide me through this work, and for their wise and precious advices.
I wish to thank all my friends, my girl friend and my family for their support and optimism during my work. And especially Quentin Chevalier for his time, support and help.

Abstract

Quality of Service (QOS) monitoring regroups different measurement functions that mostly have a diagnostic role and allow for an end-to-end monitoring.

In the IPv4 Internet, measurement techniques are popular and widespread.

However IPv6 has been designed to slowly replace IPv4. Accordingly, it is important to study whether or not the Next Generation Internet Protocol will allow the performing of QOS measurement.

It is necessary to analyse IPv6 to discover if performance measurement will still be possible.

This work shows that IPv6 permits the design of measurement functions in a natural way, very similar to the design of some IPv4 functions. The design and implementation of a basic prototype function are given as an example.

But IPv6 brings many new features, among which some of them could be exploited in a way to find a smarter way to design measurement functions for IPV6. A global survey of IPv6 is made, in order to determine which features might be interesting for this purpose.

Evoked solutions are compared, and the design and implementation of an experimental measurement function based on an IPv6 interesting chosen facet are tested. All the implemented functions are then regrouped in a tool.

Contents

List of Figures

Introduction

0.1 Overview

Nowadays traffic engineering has become an essential tool for every actor of the network domain. For service providers, traffic engineering consists of a set of methods used to monitor some metrics in order to maximize their network resource utilization. In a customer view, traffic engineering techniques are needed to meet the customers' demands specified in Service Level Agreements (SLA). In that case, the role of traffic engineering is to assure a certain quality of service (QOS) to a traffic class. It is therefore necessary for customers, as well as for Internet Service Providers (ISP), to be able to compare the QOS given to a real flow against the expected performance specified in the SLA, and detect possible QOS degradation. It is achieved by the QOS monitoring service.

QOS monitoring service is part of Performance Management service, it regroups several measurement functions that mostly have a diagnostic role and allow for an end-to-end monitoring. In this work we will often use the term monitoring or measurement as a shortcut for QOS monitoring. QOS monitoring can occur at different levels of abstraction, it can be used at many ISO layers such as at network layer, transport layer, application layer. Measurements techniques are popular and widespread over the internet, which is still running over IPv4.

However, in the early 1990's, it has been said that IPv4 will have to be replaced because it has reached its limits. IPv4 has been used for almost thirty years and suffers from different shortcomings. These shortcomings are mainly due to the evolution of the internet. In its early years, the internet was a wide area network established between non profit organisations. Internet Protocol version 4 was designed in this context, a 32-bit internet address was chosen (which allows for up to 4 billion inter-connected nodes) and concerns as authentication and security were not considered as critical. Today the internet represents a major economic agent connecting million users, customers and companies.

This fast growth has shown up the need for some new internet protocol requirements. Sooner or later, we will reach the maximum addresses number that can be assigned by IPv4. IPv6 has been developed by The Internet Engineering Task Force as the successor protocol to IPv4, bringing a lot of changes and evolutions in order to fix shortcomings in IPv4. The main change is the address space expansion but there are a lot of other new features.

One can be convinced or not that IPv6 will one day replace IPv4 as the protocol running on the whole Internet, but one thing is certain: IPv6 has been developed to fix every IPv4 weakness, a lot of work has been achieved on IPv6 for more than 10 years, gathering many new mechanisms and advances presented in the network world. Thus enterprise networks will probably discover an interesting facet of IPv6 from which they could take an advantage, in terms of costs or efficiency. An IPv6 niche deployment scenario is perfectly described in Jeff Young paper about IPv6 technologies [15]. According to J. Young, once that enterprises will have found a particular interest in IPv6, the niche deployment will expand, requiring a wider connectivity with other enterprises, and perhaps even a global connectivity.

In a QOS monitoring point of view, mechanisms and concepts are well known for IPv4, but what about IPv6? Will QOS monitoring still be possible over IPv6? The aim of this work is to study how we could design some measurement functions for IPv6. Based on the observation of several existing functions in the IPv4 world, we will try to see if there is a natural and basic way to design similar measurement functions in IPv6. However IPv6 brings many changes and new features, therefore another approach will be a global survey of the most interesting differences between the new Internet protocol version and the old one in the search of a smarter way to perform some measurements.

0.1.1 Objective of the work:

The first objective of the work is to have a global overview of the new IPv6 mechanisms, and the way we could take advantage of them in a monitoring point of view. We will have to determine whether IPv6 will change the way measurement is performed or not, by analyzing the new offered possibilities. During our survey, we will focus on the most interesting features for our purpose, and quickly discard some others. We will also describe the solutions that have already been proposed and the different works achieved on the subject.

The second objective of the work is the design of a prototype measurement function for IPv6. This measurement function will have to be capable of measuring a particular chosen metric. The objective is double :

- Show by an example that the design of IPv6 measurement functions can be achieved in a natural and basic way, very similarly to several IPv4 measurement methods described previously.
- Gather the solutions evoked in the IPv6 survey, compare their advantages and weaknesses, and choose the more suitable in order to design a smarter measurement function, taking advantage of an IPv6 specificity.

The final objective of this work is to implement an experimental tool regrouping the proposed functions and show that it can perform the expected measurements.

0.1.2 General structure of the work

The work is divided into five parts :

- Part one is an introduction of the concepts used in this work. A short presentation of terms such as metrics or monitoring, some definitions of the metrics we have chosen to focus on, and a fast presentation of some existing IPv4 measurement methods for these metrics : Cisco Systems IPSLA® functions, OWAMP, and IPMP.
- Part two of the work consists in a global survey of IPv6 new features, examining the differences with IPv4 and how we could take advantage of them in a measurement purpose. The survey is divided into two parts :
    1. We examine the different features that could be used in order to perform a semi-active monitoring over IPv6, exploiting IPv6 features. It is organized in a Bottom-up analysis of every change brought in an IPv6 packet compared to an IPv4. It starts by the IPv6 main header, analyze the different Options, and finally explores the new mechanisms.
    2. Active monitoring requires the use of a transport protocol; in this section we describe the different changes due to IPv6 in the main transport protocols.
- Part three is devoted to the design of some prototype functions performing active measurements.
    - the design of a basic UDP based function allowing for a Round-trip Delay measurement over IPv6 : UDP-Echo.
    - An interesting way to perform an automated multiple Round-trip Delay measurement combining the first solution with IPv6 Native multicast.
    - And finally we discuss the solutions exploiting IPv6 specificities evoked in the survey part, and choose the most suitable of them. This solution is then combined to the basic function prototype in order to obtain a smarter active measurement function over IPv6.
- Part four part contains the implementation details, such as the chosen environment, the structure of the application, the possible measurement options and the way to run it. It also contains the methodology used to test the program.
- Part five is a conclusion about the work, the objectives met and future possible improvements.

1 Prelude

1.1 Measurements over IPv4 : overview of some current methods.

Measurement methods can occur at different layers of the OSI model : network, transport or application layer. IPv6 will replace the existing network layer protocol and bring several changes to the transport layer. For this reason we will focus on these two layers.

There are mainly two different techniques to perform some performance measurements : active monitoring and passive monitoring. Both have their advantages and weaknesses, in this framework we will focus on active monitoring and on a combination of the two types called semi-passive monitoring that will be described in the IPv6 Survey part. Both active and passive measurement functions can monitor different metrics, in the next part we will define the word "metric" and describe two particular metrics on which we will focus all along this work.

1.1.1 Metrics

The aim of performance monitoring is to be capable of measuring some performance properties between different points in the network. These measurements consist of observing some performance indicators, also called performance metrics, between the chosen points. IP performance metrics are defined in the RFC 2330 [26] as some carefully specified quantities related to the performance and reliability of a network.
The IP Performance Metrics Working Group (IPPM WG) is an Internet Engineering Task Force (IETF) Working Group responsible of the development of a set of standard metrics that can be measured by performance monitoring methods. These metrics were defined by the IPPM to provide "unbiased quantitative measures of performance".

IPPM has produced many documents specifying different well known metrics such as :

- Connectivity
- Round-trip Delay
- One-way Delay
- Round-trip and One-way loss
- Delay Variation
- ...

The aim of this work is mainly to explore IPv6 new features, not to implements as many metrics measurement functions as possible.

In our goal to design some performance monitoring functions exploiting IPv6 new features, we had to make a choice on the metrics that would be measured. We have chosen to focus on two well known metrics as they are defined by the IPPM : The Round-trip delay (RTT) and the One-way delay.

1.2.1.1 Round-trip delay

The Round-trip delay or Round-trip time (RTT) of a packet from a source host to a destination host is the time required for a packet to travel from the source to the destination and back again.

Round-trip delay metric for IPPM is described in the RFC 2681 ([28]). The RFC defines a singleton for Round-trip delay called "Type-P-Round-trip-Delay". This singleton contains three parameters :

- **Src**, the IP address of a host
- **Dst**, the IP address of a host
- T, a time

The value of a Type-P-Round-trip-Delay singleton can be a real, or an undefined number of seconds :

1. For a real number of seconds dT :
   A Type-P-Round-trip-Delay from the source to the destination at time T of value dT means that the source sent the first bit of a Type-P packet to the destination at an absolute time T, the destination received that packet, then immediately sent a Type-P packet back to the Source, and that Source received the last bit of that packet at the absolute time $T+dT$.

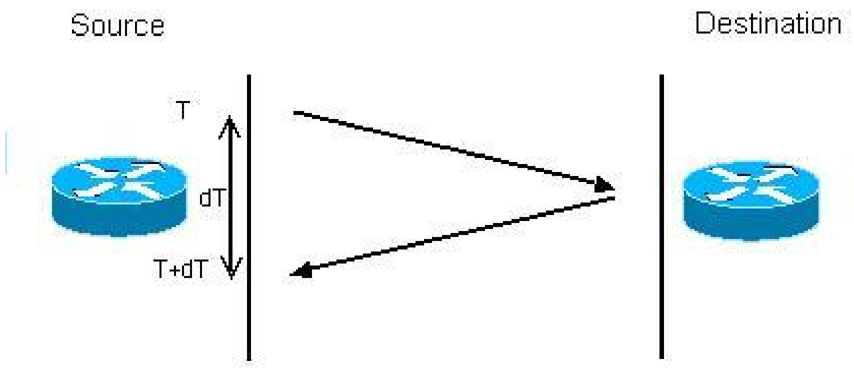


Figure 1: Round-trip Delay Calculation

2. For an undefined number of seconds :
   An undefined Type-P-Round-trip-Delay from the source to Dst at T means that the source sent the first bit of a Type-P packet to the destination at a time T and either the destination did not receive the packet, or because the destination did not send a Type-P packet in response, or the source did not receive that response packet .

In practice, the source and the destination are network entities with limited resources. These resources can be more or less used depending on the traffic load transiting through the entity. The processing time at an end-system is the time difference measured between the arrival of a packet at an entity interface and its treatment by the entity. In order to respect the definition of the Round-trip Delay metric it is important to remove the processing time of the Type-P-One-way-Delay packet at source (see further examples in section 1.1.3).

1.2.1.2 One-way Delay

The One-way Delay of a packet from a source host to a destination host is the time required for a packet to travel from the source to the destination. One-way Delay metric for IPPM is described in the RFC 2679 ([29]). This RFC defines a singleton for One-way Delay called Type-P-One-way-Delay. This singleton contains three parameters :

- **Src**, the IP address of a host
- **Dst**, the IP address of a host

- T, a time

The value of a Type-P-One-way-Delay singleton can be a real, or an undefined number of seconds :

1. For a real number of seconds dT :
   A Type-P-One-way-Delay from the source to the destination at time T of value dT means that the source sent the first bit of a Type-P packet to the destination at an absolute time T, the destination received the last bit of that packet at the absolute time $T + dT$.

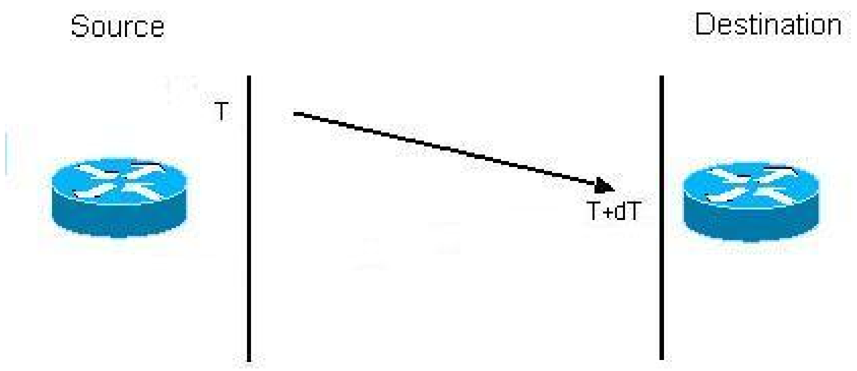


Figure 2: One-way Delay Calculation

2. For an undefined number of seconds :
   An undefined Type-P-One-way-Delay from the source to Dst at T means that the source sent the first bit of a Type-P packet to the destination at a time T and the destination did not receive the packet.

In order to respect the definition of the One-way Delay metric it is important to remove the processing time of the Type-P-One-way-Delay packet at both ends. A difference with the Round-trip Delay calculation is that One-way Delay calculation requires perfect time synchronization between both end-systems. Indeed, for Round-trip Delay it is only necessary to compute a relative time between the departure and the arrival of the Type-P-Round-trip at the source entity. But One-way Delay requires an absolute timestamp at the packet departure from the source and an absolute timestamp at the arrival at the destination, in order to accurately measure their difference.

1.1.2 Passive Monitoring

The first technique to perform performance measurements is a passive approach, it consists to collect locally all data transiting by a node in order to analyse data traffic transmitted through the network. There is no feedback from the network; it does not require any co-operation of end-points. The passive monitoring devices are polled periodically and information is collected (in the case of SNMP devices the data is extract from Management Information Bases (MIB)) to assess network performance and status. The quality of analysed information depends on the granularity and integrity of collected data.

The passive approach has a major advantage: it does not increase the traffic on the network.

However, the polling required to collect the data and the traps and alarms all generate network traffic, which can be substantial. Another weakness is that it can only measure the performance experienced by the carried traffic. If there is no traffic, the passive approach cannot conclude whether there is a connectivity concern or not.

An example of a passive monitoring tool is Cisco Systems IOS ® Netflow. For more information about what a tool such as Netflow can provide see [30].
In this framework we will not focus on passive monitoring, but rather on active monitoring and a combination of passive and active.

1.1.3 Active Monitoring

Active monitoring can be defined as active techniques that consist in injecting additional traffic with known characteristics into the network to test particular attributes of a service [11]. Active monitoring requires co-operation from both measurement end-points, it allows an end-to-end calculation of different variables such as delay and jitter. Active monitoring allows to measure network service performances even if there is no existing traffic on the network. The accuracy of the performance test depends on the frequency used to inject some test traffic. However test traffic requires some resources from the network. Active monitoring is thus a compromise between better accuracy due to faster probing and at the same time not disturb the real traffic on the network by injecting too much additional load.
Active monitoring can be used at different OSI layers, each use at a particular layer has its own specificities :

1. Network layer
   Measurements at this layer allow gathering information about the network state. They may be used for many purposes such as QOS monitoring or network layer problems troubleshooting.

2. Transport layer
   Measurements at this layer is used for end-to-end measurements between two transport entities. It can only monitor metrics on a full path. It produces better estimations of the application performances than at the network layer and it takes into account transport protocol effects on performances.

3. Application layer
   It is useful for measuring specific application(s) performances over the network. As application performances depend on the lower layers performances, it also gives a good feedback of the network and transport layer. It does not permit to get detailed information about these layers but good performances at the application layer mean good performance at lower layers.

1.2.3.1 Examples of measurement functions for IPv4 :

Before going further in our analysis we will have an overview of different measurement functions working on IPv4. A lot of different metrics can be measured with a monitoring function. Most IPv4 measurement functions are based on the transport layer. In particular, at the transport layer, we can measure the metrics we have chosen to focus on : Round-trip Delay and One-way Delay.
In this section we will briefly describe some IPv4 measurement methods for these metrics, which are part of the Cisco Systems IPSLA® tool. They all rely to the transport layer.

IPSLA® is a network end-to-end IP Service levels monitoring tool, it regroups many different measurement functions that are typically used to "verify service guarantees, validate network performance, identify network issues and react to performance metrics" ([32]).

In order to perform an accurate measurement of a singleton metric such as Round-trip Delay as it is described by the IPPM, the measurement functions have to take into account the processing times at both ends. For this purpose, every IP SLA operation designed to measure global delay will use different timestamps. One timestamp will indicate the time of arrival of the packet at the interface, and the other will contain the time when the packet is treated.

On the figure 3, timestamps TS2 and TS3 allow to remove the target processing time, and timestamps TS4 and TS5 the sender processing time. The Round-trip Delay metric between the source and the destination can be measured by $TS4-(TS3-TS2)-TS1$and the One-way Delay can be measured by $TS2-TS1$.

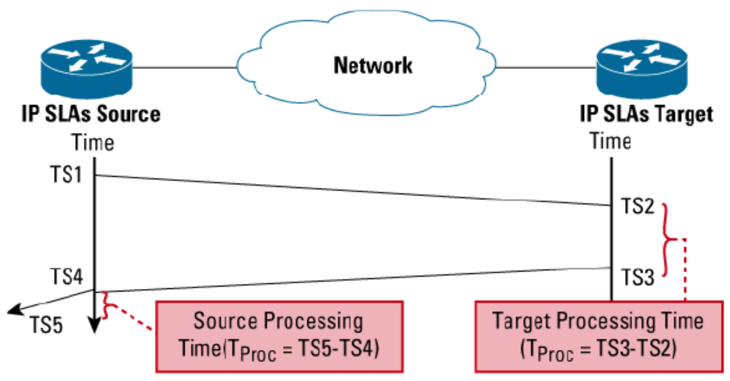

Network
IP SLAs Source
Time
IP SLAs Target
Time
TS1
TS2
TS3
TS4
TS5
Source Processing Time(TProc = TS5-TS4)
Target Processing Time (TProc = TS3-TS2)

Figure 3: Cisco IP SLA Global Delay and One-way Delay general measurement method - Source [5]

This measurement scheme requires a listening entity on the target host in order to compute the processing times of the target and send them back to the source. As seen above, measuring the One-way Delay has another pre-requirement : the synchronization of the end-system clocks. This synchronization can be achieved by protocols such as Global Positioning System or other methods like Network Time Protocol (NTP) [31] .

Based on these general schemes, Cisco Systems IPSLA® provides different measurement functions to compute Delay or One-way Delay. The differences between these functions are that they perform Delay or One-way Delay measurements over different transport protocols. Indeed the objective of performance monitoring is to evaluate the performance encountered by real data traffic. Data traffic can be of different types, depending on the service it is providing. Services can use connection oriented or connection-less transport protocols. The main three types of transport protocols used by these services are : UDP, TCP and ICMP. IPSLA allows to perform measurements for these three protocols thanks to three different operations :

1. IPSLA : ICMP Echo

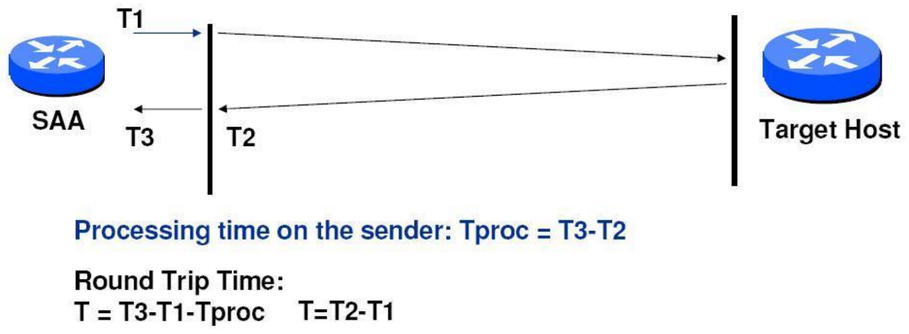


Figure 4: Cisco IP SLA ICMP ECHO Operation - Source [5]

ICMP Echo uses ICMP Echo request and Echo reply messages to measure the Round-trip Delay metric between two entities. The operation permits the removal of the source processing time for a more accurate result. The big advantage of ICMP Echo is that it can be performed toward any IP host.

2. IPSLA : UDP Echo

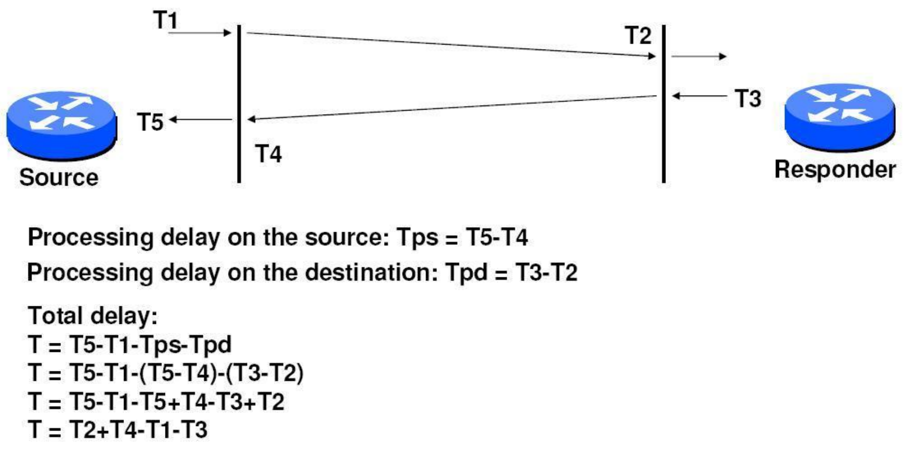

T1
T2
T3
T5
T4
Source
Responder
Processing delay on the source: Tps = T5-T4
Processing delay on the destination: Tpd = T3-T2
Total delay:
T = T5-T1-Tps-Tpd
T = T5-T1-(T5-T4)-(T3-T2)
T = T5-T1-T5+T4-T3+T2
T = T2+T4-T1-T3

Figure 5: Cisco IP SLA UDP ECHO Operation - Source [5]

UDP Echo is based on UDP packets, it allows for a Round-trip Delay and also a One-way Delay calculation. A dedicated entity listening on the destination host is required for the One-way Delay measurement; it is called a **Responder**. The role of the responder is to listen on a particular UDP custom port for incoming UDP Echo requests, and answer to them with the addition of two timestamps (as seen in the figure).

3. IPSLA : TCP Connect

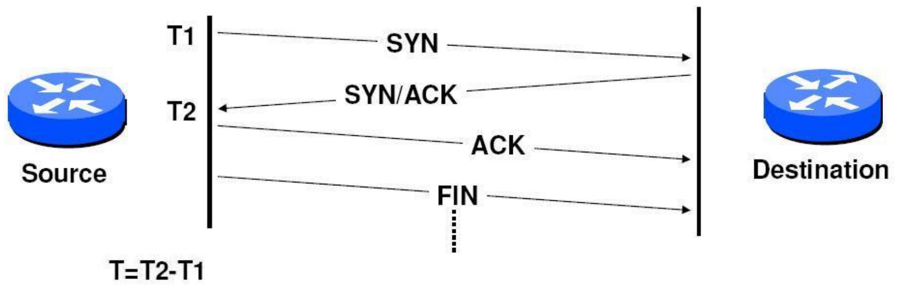


Figure 6: Cisco IP SLA TCP Connect Operation - Source [5]

TCP Connect is an operation that can measure the time taken by the source to establish a TCP Connect operation to any destination host. Here a responder entity listening on the destination is not required, any TCP listening service can be tested.

4. IPSLA : ICMP Echo Path
ICMP Echo Path consists of a measurement of the Hop-by-Hop response time between a source and a destination. For this purpose the operation will first discover the path using *traceroute*, and then perform an individual ICMP Echo operation toward each intermediary hop on the path to the destination.

1.2.3.2 One-way Active Measurement Protocol (OWAMP)

OWAMP is a very complete protocol that allows measuring unidirectional metrics such as One-way Delay and One-way Loss. High accuracy in the measures requires a perfect synchronization between the end systems (using GPS for example). The idea in OWAMP design was to create a technique allowing singleton metrics measurements, based on the initiation of test packets exchange. OWAMP designers wanted to constitute a standard for collecting metrics across a wide mesh of Internet path.

OWAMP consists of two protocols : a control protocol and a test protocol.

- The control protocol is based on TCP. It is used to initiate, start and stop test sessions and to fetch the results.

- The test protocol is used to exchange test data contained in UDP packets between measurement nodes.

OWAMP has interesting characteristics :

- It is hard to detect because it is based on UDP packets, and because test sessions are dynamically negotiated (not on a static port).
- It supports security features, there are three different modes : unauthenticated, authenticated, and encrypted modes.
- It can be used to collect every IPPM metric.

For more information about OWAMP see the RFC 4656 [17].

1.2.3.3 Internet Protocol Measurement Protocol (IPMP)

Internet Protocol Measurement Protocol is another active monitoring protocol. It has been designed by an IETF working group [17] as an evolved echo request - echo reply protocol. It is also designed to allow intermediate routers to piggyback additional information in the request, in order to record delivery timestamps along the path.

Typically, a host that wants to perform a measurement via IPMP will send an IPMP Echo Request packet to the desired target, and the receiver will answer with an echo reply. Every IPMP capable routers on the path will be piggyback an additional indicator to the request : a timestamp. The timestamp will allow the calculation of the one-way delay between each node on the path.

Another IPMP features are IPMP Information request and reply packets, which allow to calculate the inference between a node's clock time and the real time.

However, IPMP is not a very popular and widespread protocol. One main reason is that IPMP requires the definition and standardization of several new type of protocols : IPMP Echo request and reply, IPMP Information request and reply. It requires the assignation of four new Internet Protocol types by the Internet Assigned Numbers Authority (IANA). IANA is the authority responsible for every internet protocol assignment, thus a worldwide consensus is needed for an assignment. We can keep this concern in mind in our need to find a good way to perform some performance monitoring over IPv6. As we will see further, a new extension header requires the assignment of a new "next header" type by the IANA (see 2.1.3).

2 IPv6 Survey - Exploring IPv6 new features in a monitoring approach

IPv6 has been designed by the IETF in a way to ensure its interoperability with the existing protocols : different mechanisms are specified to facilitate the transition from IPv4 to IPv6. It was also a pre-requirement to leave the widespread transport protocols UDP and TCP unchanged. This will allow a natural and basic way to extend the multiple existing IPv4 measurement functions based on the transport layer to work over IPv6. For example, in the third part of this work we will describe a really basic UDP measurement function working over IPv6.

IPv6 has been designed to bring several major changes to the existing IPv4 protocol, the original set of attributes of the protocol are the following :

1. **Expanded Addressing Capabilities**
   The best known is the increase of IP address size to 128 bits in order to solve IPv4 lack of addresses in the years to come, but there are many other things like simpler addresses auto-configuration, native multicast, and anycast addresses.
2. **Header Format Simplification**
   A choice has been made to remove and replace several fields in the IPv4 and to simplify IPv6 header. This simplification permits a faster processing of the header by the intermediate nodes on a path.
3. **Improved support for Extensions and Options**
   The way Internet protocols options are carried has been totally reviewed, options are now separated from the main header and carried into Extensions Headers. The big difference is that options are no longer processed by intermediate nodes, except for one, which allows for a faster processing and a higher flexibility for new options in the future.
4. **Flow Labelling Capability**
   An additional field in the header enables to tag packets belonging to the same kind of traffic (same flows), in order to request a special handling from the network.
5. **Authentication and Privacy Capabilities**
   New options carried by specific extension headers bring authentication, data integrity and optionally data confidentiality native support to IPv6

These attributes have required many changes in the header and several new mechanisms. In this part we will examine every change and new feature in a performance monitoring view. We will try to determine if the Internet Protocol Next Generation provides a smarter or more efficient way to perform some measurements than IPv4.

This part will be divided into three sections:

- In the first, we will have a global overview of the different changes in IPv6.
- In the second, we will search for some mechanisms that could be used to set up a semi-passive monitoring operation over IPv6.
- And finally we will explore every possibility to set up an active monitoring operation over IPv6.

2.1 Semi-passive monitoring over IPv6.

2.1.1 Semi-passive monitoring.

Semi-passive monitoring combines the principles of both active and passive monitoring. We will refer to it as a monitoring methodology allowing to measure some metrics between nodes without explicitly injecting a synthetic traffic. Semi-passive monitoring uses existing data traffic as a transport mean for its monitoring purpose.
One of the advantages is to avoid introducing more overhead on the network, by using piggybacking mechanism on the traffic passing through a node. Different kinds of traffic can transit through a node :

1. The traffic is generated locally and will be sent to the next hop on the destination path.
2. The traffic comes from a source node and the destination is the local node.
3. The traffic comes from a source node and will be forwarded to the next hop on the destination path.

Typically a semi-passive monitoring agent will add monitoring information on the type 1 and 3 of these three kinds of traffic. The information will be processed either by the destination or by all the monitoring capable agents on the path to the destination.

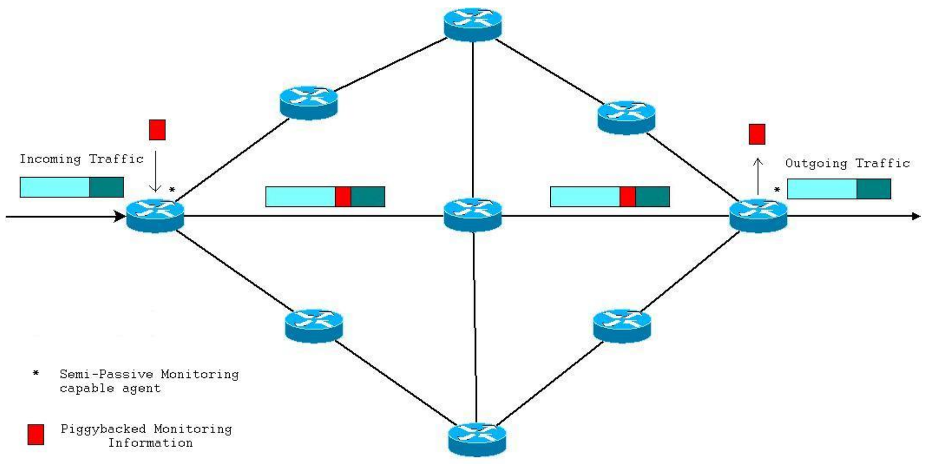

Incoming Traffic
Outgoing Traffic
*
*
* Semi-Passive Monitoring capable agent
🟥 Piggybacked Monitoring Information

Figure 7: Semi-passive Monitoring Principle

Semi-passive monitoring requires finding a well-suited field in the network layer protocol (IPv6) or in another higher level layer (transport layer) in order to add the monitoring data in the traffic. The next section will explore IPv6 possibilities to provide a semi-passive monitoring methodology. We will begin with the basic features (IPv6 main header) and move on to more sophisticated (option headers, Neighbor Discovery, ...).

2.1.2 IPv6 Header

The main difference between the IPv6 header with IPv4 is that it has a fixed 40 bytes length, it does not contain any checksum and is thus simpler and permits better performance. Let us review every field and see if it can be interesting for our objective :

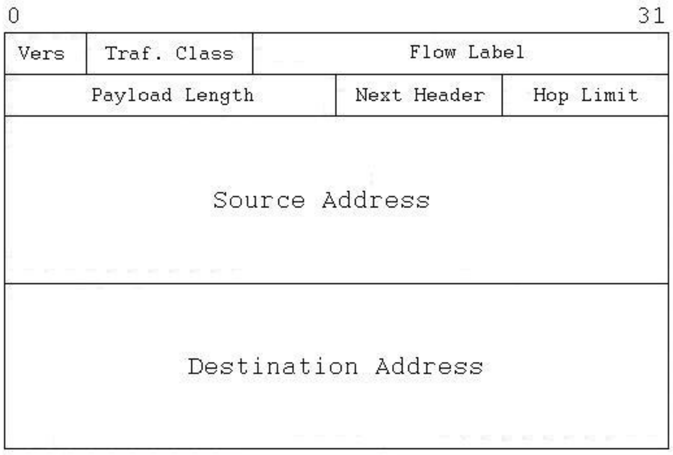


Figure 8: IPv6 Header

Version:	4-bit Internet Protocol version number = 6.
Traffic:	Class 8-bit traffic class field. This field is used by origin node or routers to identify a class of traffic and eventually to apply a priority policy to it. It cannot be used to store any useful measurement information, it could be modified by any nodes on the path and it is not designed for that purpose.
FlowLabel	20-bit Flow label. According to RFC 2460 “A flow is a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers.” The question is : could we possibly use the flow label identifier to specify a special monitoring handling on the data transiting by the intervening routers ?

On traffic type 3 2.1.1, it seems to be impracticable to give to the data flow a new flow label. The source has already specified a flow label which has a meaning on the path. It would be a non-sense to modify it. RFC 3697 is very clear on that: “The Flow Label value set by the source MUST be delivered unchanged to the destination node(s).”

On traffic type 1 2.1.1, we could make a decision to set a special flow label for each packet requiring a special monitoring treatment. But two problems would appear :

- Firstly, we would not be able to set any other special treatment for the real data payload.
  Hence, we would not be able to distinguish any different flows. For example, one flow carrying some real time data and another some best effort data would have the same treatment.

- Secondly, if we use the flow label as a solution for piggybacking, we will indirectly give to every node processing the IPv6 header an easy way to identify a measurement packet. Thus, nodes could decide to apply a special treatment to these packets. For example monitoring packets could get a higher or lower priority compared to the other flows. It would have a big influence on the metrics measured by the monitoring function which would not reflect the network state anymore.

For these reasons, the flow label which could seem to be an interesting solution at first glance, is in fact not designed for this purpose.

PayloadLength 16-bit unsigned integer. Length of the IPv6 payload in bytes. Payload Length set to zero means that the jumbogram option is used.

NextHeader 8-bit selector. Identifies the type of header immediately following the IPv6 header. IPv6 has no option fields in its header but options can be specified in extension headers. We will inspect the different encapsulated headers possibilities for monitoring in the next section.

HopLimit 8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.

Source 128-bit address of the originator of the packet.

Destination 128-bit address of the intended recipient of the packet (possibly not the ultimate recipient, if a Routing header is present).

IPv6 main header is much simpler than IPv4. As it will be processed by every node on the path, its designers have chosen to reduce its complexity in order to facilitate its treatment. It also simplifies the need of standardisation for possible options treatments that have to be implemented on every node. The few fields present in the main header do not provide any options for a measurement on actual data traffic. Actually, IPv6 options are no longer part of the main header. IPv6 Options are contained in Extension Headers that will be discussed in the next section.

2.1.3 IPv6 Extension Headers

Extension headers can be used in IPv6 to add different options to the IP packet.

But IPv4 could carry some options too, so why would IPv6 extensions be more interesting in a monitoring view than the IPv4 option fields? Because there is a big difference in the way there are handled :

In IPv4, options are a part of the IP header. Hence, it is processed by every router on a the path from the source to the destination. It requires from every node the capacity to handle all the options, it has two impacts :

1. Option fields need to be standardised in order to to be recognized and processed on every hop.
2. The processing of these options has a cost in term of time and computation power, this cost is repeated on every node.

In version 6, a choice was made to change the way IP header options were encoded. Its reasons are a more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future [1].

The options are encoded in extension headers, these headers are separated from the main header. They are used between the IPv6 main header and the upper-layer header. They are not mandatory, and they act like an intermediate layer between the network layer and the transport layer.

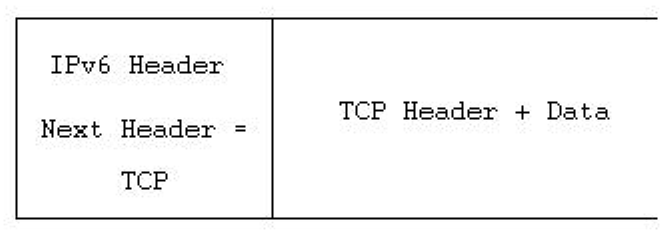


Figure 9: IPv6 Packet without Extension Headers.

There are a small number of such extension headers, each identified by a distinct Next Header value.

A full implementation of IPv6 includes implementation of the following extension headers:

- Hop-by-Hop Options
- Routing (Type 0)
- Fragment
- Destination Options
- Authentication
- Encapsulating Security Payload

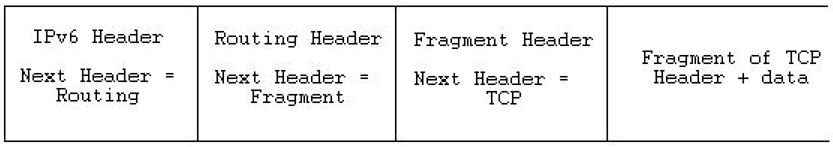


Figure 10: IPv6 Packet with two Extension Headers : Hop-by-Hop and Destination Options.

The first advantage of using an extension header as a carrier for any monitoring information is that, "with one exception, extension headers are not examined or processed by any node along a packet's delivery path, until the packet reaches the node identified in the Destination Address field of the IPv6 header"[1]. Hence, it allows a total transparency for a semi-passive monitoring operation. The nodes along the path that are not capable of handling a semi-passive monitoring treatment will not even be aware that it is contained in an extension header.
The second advantage is the flexibility brought by the separation between the main header and the extensions. There are six defined extension headers, but the number of possibilities to create a new extension header is linked to the Next Header field of the main header. The Next Header is a 8-bit field, so there is 256 possibilities for upper-layer headers among which number 136 to 254 are actually left unassigned.
That lets us two choices : Either try to use one of the defined extension headers to carry our monitoring context, or use one of the unassigned Next Header value to create a new monitoring protocol extension. We will first check if it is possible to use one of the existing defined extensions for our purpose, and thereafter compare with the second solution.

2.1.3.1 Examining the defined extension headers

We will focus on three of the defined extensions that seem to be interesting from our point of view. There are the following : Hop-by-Hop options header, Routing header, and Destination options header.

1. **Hop-by-Hop options**
   The Hop-by-Hop Options header is used to carry optional information that must be examined by every node along a packet's delivery path. It is the only header that will be processed by every node on a path. It has the following general format :

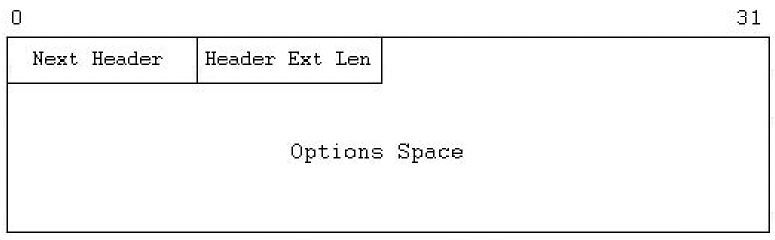


Figure 11: Hop-by-Hop Options Header Format.

The option field is a Type Length Value (TLV) encoded field which is also used in the Destination extension header. The Type Length Value format is shown in figure 12.

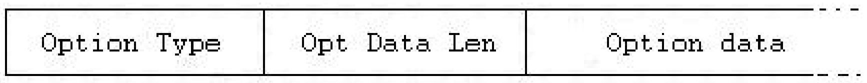


Figure 12: Hop-by-Hop and Destination Options TLV Format.

There are currently four options defined : two padding options, the jumbogram option and the router alert option. The type of an option is a 8-bit field where 2 bits are used to specify the action that must be taken if the processing IPv6 node does not recognize the option type :

(a) 00 : the router must ignore the option (skip it and continue the processing of the packet).

(b) 01 : the router must reject the packet.

(c) 10 : the router must reject the packet and send back an ICMP Destination Unreachable message.

(d) 11 : the router must reject the packet and send back an ICMP Destination Unreachable message if the destination address is not multicast.

The third bit is used to specify whether a router can change en-route the option content or not. It lets us 6 bits as an option type label, so 64 possibilities among which 60 are unassigned.
It is thus possible to create a new option type associated with a new monitoring operation in order to carry some context and process it in every node on the path.

This mechanism has already been developed, in particular in H. Kitamura's work [12]. We will briefly have an overview of his work, and we will focus on the advantages and concerns of using the Hop-by-Hop option header to implement such a mechanism.

H. Kitamura's Connection Status Investigation (CSI)

The idea of CSI is to couple a new kind of ICMP control message with a new option called CSI option in the Hop-by-Hop extension header. The operation works as an evolved and smart *traceroute*. An ICMP status request message is sent to a destination and acts as a trigger for every node on the path. In reaction to the trigger, the nodes will process the CSI option contained in the hop-by-hop extension header.

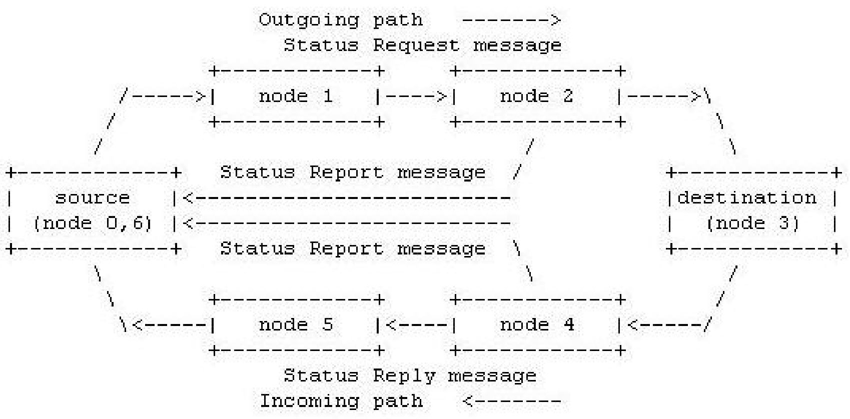
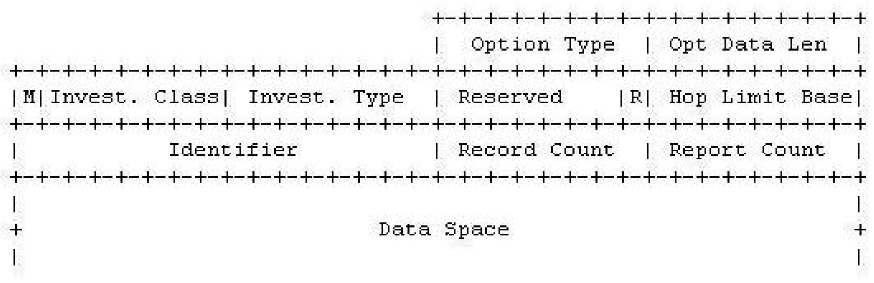


Figure 13: Connection Status Investigation (CSI) - Source [12].

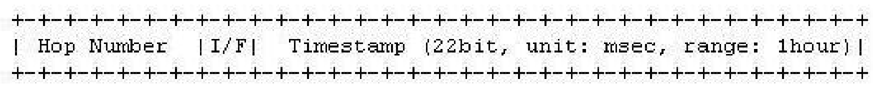
Status request, status reply and status report messages are new ICMP control messages. The status report is sent by a CSI capable router when the CSI option field in the IPv6 Hop-by-Hop extension header of the received status request is full. The two other messages work as ICMP echo and ICMP request messages, their role is to trigger the special treatment on CSI capable nodes and to carry the sub-layer headers (IPv6 and extensions).



```
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |  Option Type  | Opt Data Len  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|Invest. Class| Invest. Type  | Reserved    |R| Hop Limit Base|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Identifier           | Record Count  | Report Count  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                          Data Space                           +
|                                                               |
```

Figure 14: Hop-by-hop CSI Option format - Source [12].

Thanks to the two first bytes of the option type, the CSI option can be used on a network where not all routers are CSI capable. When a router does not know how to handle an option type field, it refers to these first bytes to take an action. If it is set to 00, the router will skip over this option and continue the processing of the header. When a CSI capable router process a Hop-by-hop extension header and recognize the CSI option type value, it will check the value of different fields in the CSI option in order to take part in measurements. Investigation class and investigation type fields specify the type of indicator that has to be piggybacked in the option data by every CSI capable node on the path delivery.



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Hop Number  |I/F|  Timestamp (22bit, unit: msec, range: 1hour)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: IPv6 Hop-by-hop CSI Option : Mandatory Data Component Record - Source [12]

The Record Count Field allows nodes to copy their indicators at the right place in the option data. If the maximum of records is reached, CSI nodes will send a status report ICMP message with a copy of the records to the source (or to the destination depending on the R bit value), clear the data records and increment the Report Count Field.

What are the advantages taken by the CSI mechanism as an option field in the Hop-by-Hop extension header ? It benefits from the extension's more interesting property, which is that it is processed by every router on the path. The TLV-encoded method allows the mechanism to be used with non CSI capable routers, that will ignore it .

There are three disadvantages in this method, the first one is directly linked with the CSI mechanism, and the two others are more general to the use of the Hop-by-Hop extension.

- First, the CSI mechanism requires three new defined and standardized of ICMP control messages : status request , status reply and status report.
  Are the two first really necessary ? They have the same behaviour than ICMP echo request and echo reply, except the difference that they act as a trigger. But according to RFC 2460 “The Hop-by-Hop Options header is used to carry optional information that *MUST* be examined by every node along a packet’s delivery path”. Thus, the CSI option should be examined and recognized by any CSI capable router, without the need of an additional trigger.
- The second disadvantage is the impact on the processing time at every step of a packet’s delivery. Each intermediary router will have to process the hop-by-hop extension header, and the CSI option type. It is a 8-bit only option type that has to be matched, thus it will not have a big incidence on the delivery global delay.
- And finally, this mechanism will encounter the same concern as described in the flow label section (2.1.2). Here, a simple 8-bit option type will let every router know that the whole packet has a special monitoring purpose. This knowledge could lead certain routers to have different priority behaviour toward packets containing a CSI option type, which would compromise the measurement function.

2. **Routing Extension Header (RH).**
   “The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet’s destination. This function is very similar to IPv4’s Loose Source and Record Route option.” [1]. The Routing Header will not be used to piggyback any monitoring context, because it cannot contain anything other than routing information.
   But an interesting property of the Routing header is that it can specify a lot of intermediary destinations. During the packet’s delivery, every destination in the Routing header list will become an IPv6 main header destination, on which the Destination header will have to be processed (see 3). The Routing extension Header in itself is not useful for our purpose, but its combination with the Destination header is a smart way to perform operations on some chosen nodes.

3. **Destination Options Header.**
   The Destination Options header contains options that should only be examined by a packet’s destination node(s). During its delivery, a packet destination can change en-route if different destinations have been recorded in the Routing Header. In this case, every intermediary destination will also process the Destination Options. Options are TLV-encoded, as in the Hop-by-Hop Extension.

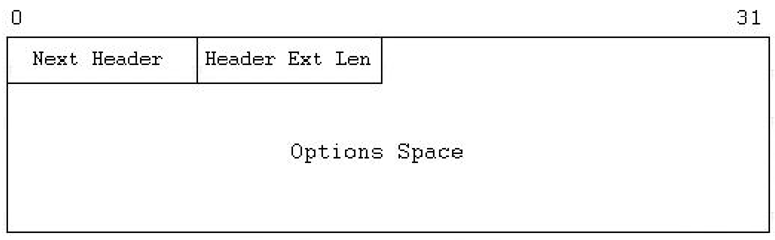


Figure 16: IPv6 Destination Header Format.

We could use the Destination Header in order to measure some metrics between two hosts. This can be done by adding a piggybacked monitoring context to existing traffic, in the IPv6 Destination extension header. It requires to create a new monitoring TLV-encoded option type (for example like the CSI option).
The main advantage by comparison to the Hop-by-Hop solution is its opacity for other nodes: the Destination extension will only be processed by the destination of the IPv6 packet. Other routers on the path will not be aware that it is running because they do not inspect the Destination Header. Hence, the operation is not detectable by the intermediary nodes, and the monitored data should not undergo any special treatment.

In his work [14] D.P. Pezaros has achieved a native in-line monitoring methodology, based on IPv6 destination header.

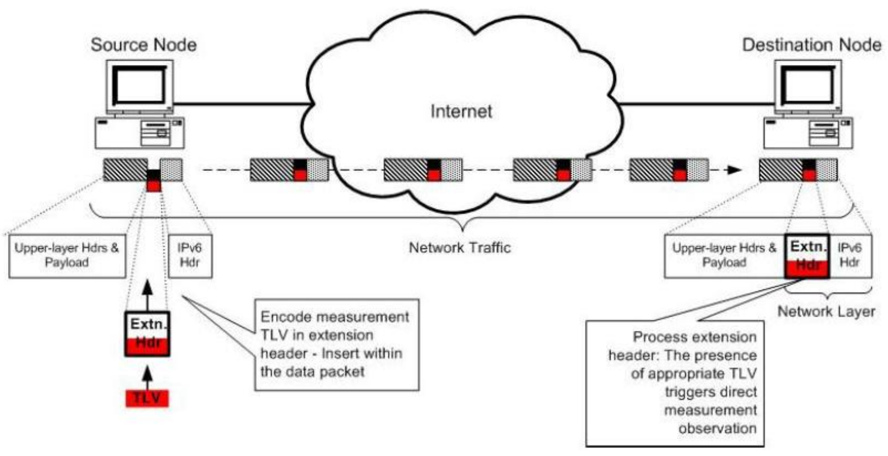
D.P. Pezaros's In-line measurements.

(a) General description of the method.
His work presents a new "in-line measurement" based on IPv6. According to D.P. Pezaros, "In-line measurement" is a new hybrid measurement technique, based on active and passive measurements properties. It is thus a new type of performance monitoring that can be classified in what we call in this framework the semi-passive monitoring family of functions. In-line measurement combines the advantages of passive and active measurement :

   i. The method, like passive monitoring methods, does not inject any synthetic traffic in the network. It operates on the actual data traffic by adding some measurement indicators. There is no feedback from the network; which makes it more complex to correlate the measurements from multiple points in the network.
   ii. Similarly to the active measurements methods family, in-line measurements allow an accurate estimation of traffic flows performances in real time. As the technique uses actual data traffic as a "piggybacker" for measurement indicators, measurements results will exactly reflect the performance experienced by data. However measurement tests do not have to be periodic as other

active measurements. The reason is that they will only be triggered by the presence of traffic type that has to be monitored. Hence, this solution does not allow measuring any service performances during lack of traffic of interest.

In-line measurements are “multi-point measurements whereby packets are tagged with measurement information at one point in the network, and this information is observed, augmented and retrieved at a points (or points) elsewhere.



Source Node
Internet
Destination Node
Network Traffic
Upper-layer Hdrs & Payload
IPv6 Hdr
Extn. Hdr
TLV
Encode measurement TLV in extension header - Insert within the data packet
Upper-layer Hdrs & Payload
Extn. Hdr
IPv6 Hdr
Network Layer
Process extension header: The presence of appropriate TLV triggers direct measurement observation

Figure 17: Inline Measurement Technique - Source [14].

A major advantage of using IPv6 rather than IPv4 in an in-line measurement goal is the new way IP options are handled. Pezaros chose to exploit the destination header as a carrier for an in-line measurement. He defined new TLV-encoded options to be inserted as part of the destination extension header within the actual data packets. These new options also allow piggybacking some measurements indicators along a packet's delivery path. Typically, a node that wants to perform an in-line test on a particular traffic type will select the traffic of interest and create the appropriate measurement option in the destination header in order to hold an indicator of the metric measured (for example a timestamp or a packet counter). At the destination node, the destination header will be processed. The presence of the TLV-encoded option will trigger a direct measurement observation.

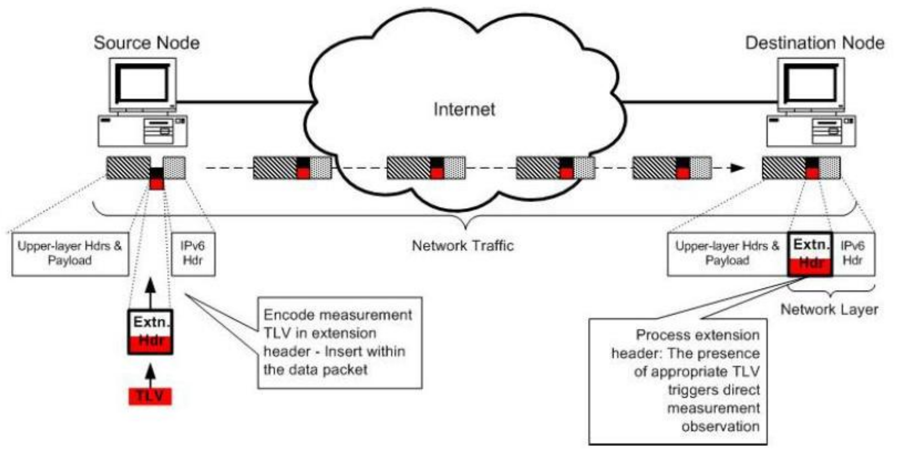

Source Node
Destination Node
Internet
Network Traffic
Upper-layer Hdrs & Payload
IPv6 Hdr
Extn. Hdr
TLV
Encode measurement TLV in extension header - Insert within the data packet
Network Layer
Process extension header: The presence of appropriate TLV triggers direct measurement observation

Figure 18: D.P. Pezaros In-line Measurement method using TLV-Encoded Options - Source [14].

(b) New defined TLV-Encoded options
He defined two new TLV-Encoded options :

i. One-way Loss (OWL) TLV-Encoded Option

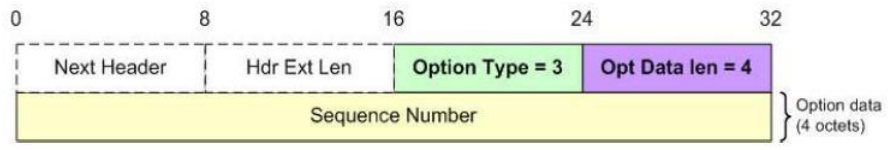


Figure 19: Destination Header encapsulating a One-way Loss TLV-Encoded Option - Source [14].

One-way Loss option (OWL) is used to measure packet loss between two points in the network. The method is really simple : The origin node choose a traffic type of interest, and piggy-back in a corresponding packet the OWL option with a sequence number based on a local policy. At an identified destination, the destination header is processed and the OWL option is extracted. The sequence number is then stored, and by observing sequence numbers of successive packets satisfying some common classification criteria the destination can compute the One-way loss.

ii. One-way Delay (OWD) TLV-Encoded Option

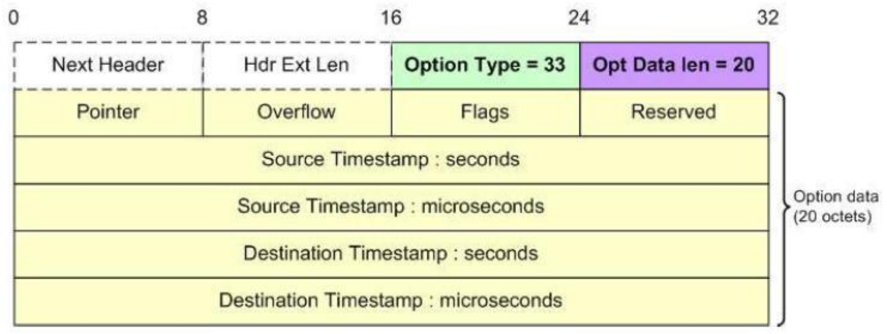

0
8
16
24
32
Next Header
Hdr Ext Len
Option Type = 33
Opt Data len = 20
Pointer
Overflow
Flags
Reserved
Source Timestamp : seconds
Source Timestamp : microseconds
Destination Timestamp : seconds
Destination Timestamp : microseconds
Option data (20 octets)

Figure 20: Destination Header encapsulating a One-way Delay TLV-Encoded Option - Source [14].

One-way Delay (OWD) TLV-Encoded option has been designed to allow the performing of a One-way delay measurement between two points in the network. Let us describe the essential Option fields :

A. **Pointer** : it indicated the octet which begins the next timestamp to be added

B. **Overflow** : it is used to indicate if nodes have tried to insert more timestamps than there are slots to accommodate them.

C. **Source Timestamp seconds/microseconds** : two 32-bit timestamps to record the departure time of the packet from the interface of the originator

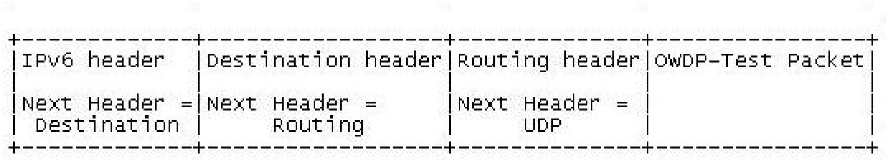
D. **Destination Timestamp seconds/microseconds** : two 32-bit timestamps to record the arrival at the interface at the node processing the OWD option.

Usage of the OWD option requires perfect time synchronization between the origin node and the destination one.

D.P. Pezaros's work is one more proof that TLV-Encoded options provide a flexible and smart way to perform measurements. His smart use of the destination header allows measurements between two points in the network. But another interesting characteristic of the destination header is that we can combine the Destination Header and the Routing Header to create a multi-point measurement. The Routing Header may specify a list of hosts that has to be visited. According to RFC 2460, each host in the Routing Header list will become a destination host during the packet's delivery, and will process the Destination Header if it is present.
The idea has been proposed in an internet draft called One-way delay Measurement using IPv6 Source Routing [33]. The draft introduces the idea of a Subpath One-way Delay measurement (SODM).

Subpath One-way Delay measurement (SODM).

The measurement packet consists of an IPv6 main header followed by three next header : two next headers are IPv6 Destination extension header and Hop-by-hop extension header, the third one is the given as a generic One-way Delay test protocol.



```
+-------------+------------------+-------------+----------------+
|IPv6 header  |Destination header|Routing header|OWDP-Test Packet|
|             |                  |             |                |
|Next Header =|Next Header =     |Next Header = |                |
| Destination |     Routing      |     UDP     |                |
+-------------+------------------+-------------+----------------+
```

Figure 21: Layout of Subpath One-way Delay Measurement Packet - Source [33]

The solution requires the definition of a specific TLV-Encoded One-way Delay measurement option in the destination header. The main difference with D.P. Pezaros's OWD option would be that the option must be large enough to contain every timestamp added by the nodes on the subpath to be measured. A special field should also be available to indicate whether the intermediate nodes were able to piggyback their timestamps or not.

Unfortunately, this solution suffers from different issues due to several Routing header concerns. These security concerns were already present in the IPv4 Loose Source and Record Route option (LSRR), and they result in the decision from many Internet actors to ignore the LSRR option. IPv6 Routing header is very similar to IPv4 LSRR. For many reasons we might think that IPv6 Routing header (RH) will be ignored too, RH brings several issues as :

(a) Avoiding destination filters
Many firewall rules are based on the destination address field, but with the use of RH, the destination address field is not always the real destination of the packet. The destination can change many times during its delivery. To solve this concern would require from firewalls to inspect deeper in the routing header extension if there is no record of forbidden destination address.

(b) Scope escape
IPv6 addresses can have four different scopes : node local, link-local, site local or global. These scope addresses must be used only in the scope perimeter, because they guarantee the address uniqueness in that scope. Routers never forward any packets where the destination scope differs from the source scope. Using the RH makes it possible to change the scope of the destination address during its travel.
Normally an IPv6 node receiving a packet on an interface whose address has a different scope than the source address of the packet, will answer to it by using its local scope. And thus the reply will be blocked by routers.

However the node could use the same RH mechanism in order to reply to the request which would create a security issue.

(c) Traffic billing
Internet Service Providers (ISP) have financial agreements with network partners (such as carriers and other ISPs). As they have to pay for the traffic passing through their peering partners, they want to have a perfect control on how the traffic is being routed. For these reasons they might not care about the presence or not of a RH. Also, a malicious partner could force the traffic to pass several times through his network in order to overbill the traffic costs.

(d) Accelerate the research of reachable nodes.
By combining an echo-request protocol with the RH, it is possible to check the availability of some nodes much faster. It consists in placing every address to be checked as a record in the RH, and to send an ICMP Echo to one of these address. If an ICMP Echo reply is received all the addresses are reachable, else an ICMP Unreachable message will be received from the last reachable address in the RH list.

The prohibition of the IPv6 Routing header extension would cause the inefficiency of any measurement method combining the Routing Header and the Destination Header, such as the subpath One-way Delay method. For now, we still do not know if managers of IPv6 networks will take this option into account or not. But based on the experience of IPv4 LSRR, there are no reason inciting IPv6 actors to care about the Routing Header. Because all encountered issues in IPv4 LSRR are still present in IPv6 RH.

2.1.3.2 A Measurement Extension Header.

IPv6 modularity and flexibility allow creating new experimental options carried into extension headers. As seen previously, there are a certain number of next header field values which are currently unassigned. This lets an opportunity to create a new type of extension, specific to monitoring operations : a measurement extension header.
There are several differences between a "dedicated" extension header or the use of a defined extension header to carry our information. Defining a new extension has a big advantage:

- a new extension header dedicated to measurements would allow to define a lot of option types specifying different possible operations (like RTT, one-way delay, loss rate, ...) where the use of the existing extension headers only permits to define several minimalist operations. It is due to the TLV-encoded format : only 6 bits specify the option type (2 exp 6 possibilities).

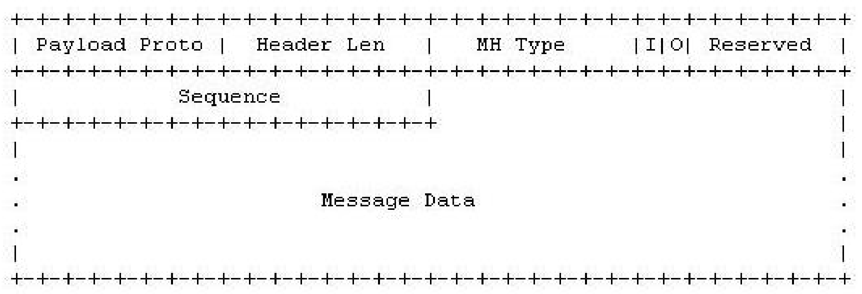
But it has some drawbacks :

- In case of unrecognized option by the destination, the two approaches (options or header) have a different behaviour :
  If a destination of a packet containing an IPv6 measurements extension header is not capable of handling this type of extension, it will normally discard the packet and send an ICMP parameter problem message to the source. Thus, the data will be lost. In the other scenario, if the destination does not recognize the option type it will skip it and process the rest of the packet.

- The new created extension will have a unique next header protocol number. It is subject to the same concern as Flow label and CSI option in the Hop-by-Hop Header : it is easily identifiable by all nodes and they can possibly adapt their behaviour while detecting a measurements extension header (see 2.1.2).
  This concern is minimized in the TVL option because it is located in a header that is not supposed to be processed by every nodes.

- A new extension requires a new protocol identifier, which has to be unique and standardised. TLV options have to be unique and standardised too, but the scope of the standardisation is local to IPv6, while the scope for a new extension is a global scope. IPMP, the complete measurement protocol described above, is a good example of a new measurement extension that has never been standardized.

The idea of a new option header dedicated to measurement is not new, it has been discussed in a draft named "IPv6 measurement header" proposed at the IETF (see [21]). The measurement header was proposed as a new IPv6 option header, thus requiring a new internet protocol type identifier. It can perform different measurements : one-way measurement, two-way measurement and allows higher layer protocol measurement.



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Payload Proto |  Header Len   |    MH Type    |I|O| Reserved  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Sequence            |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
.                                                               .
.                         Message Data                          .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 22: IPv6 Measurement Header Format - Source [21].

The message data field contains in fact the measurement options. The options are encoded in TLV format as the options in the destination header and hop-by-hop header.

```
 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 |    Option Data...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 23: Measurement Options TLV Format - Source [21].

This proposal for a new measurement header seemed to be a smart and complete measurement solution for IPv6, taking advantage of the way IPv6 options are handled option. But it has been abandoned, why ?

As we have seen above, the creation of a new internet protocol type brings several difficulties. The major concern is that the IANA authority has to be convinced of the necessity to standardize such a protocol. Actually it is not only the IANA but the whole internet community that has to find some benefits in the use of this protocol.
The measurement header possibilities are based on TLV-encoded options in the measurement header data field. We have seen in the previous section that destination header and hop-by-hop header could carry these TLV-encoded measurement options too. Thus the proposed new measurement header would have allowed the same functionalities than the use of the existing header options field. Accordingly, a new measurement header as it is described in the draft [21] would not have provided more benefits than the use of the options which encounters fewer difficulties to be developed.

It does not mean that the idea in itself of creating a new IPv6 header entirely dedicated to measurements must be forgotten. But the amount of effort needed to standardize and convince Internet actors of the necessity to widely deploy such a protocol would require from this solution to provide something really new, more complete and powerful that what the TLV-encoded options currently allow. A new extension header would have to be designed to regroup all the existing measurement features, it should handle many more options than the 2exp6 possible types allowed by the TLV options. Such a header should also be capable of measuring any metrics and be used in different modes : active, passive, semi-passive.

2.1.4 Neighbor Discovery (ND) mechanism

Neighbor Discovery (ND) is the address resolution protocol for IPv6. It is the equivalent of ARP protocol for IPv4, but it also combines a new router discovery protocol.
ND allows nodes to perform the following operations :

- Determine the link-layer addresses of the neighbours attached to the same links.

- Be aware of a cached address that becomes invalid.
- Find the different routers on the same links.
- Keep track of the neighbours that are reachable and those that are not.
- Detect changed link-layer addresses.
- When a router of the path to it fails, try to find an alternate.

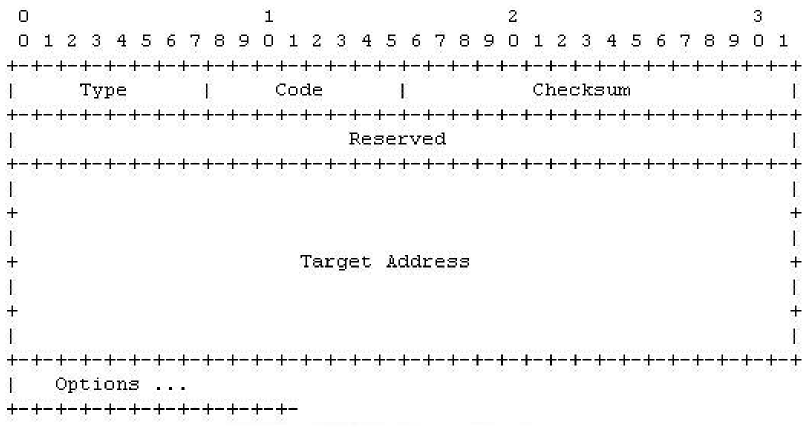
At first glance it looks like an interesting solution for our purpose, could we combine our measurement indicators with the ND mechanism ? Let us first check deeper how ND works :

The ND algorithm is based on four tables :

1. Destinations' cache
2. Neighbors' cache
3. Prefix list
4. Router list

In summary it works like this :
When a node wants to send (or to forward) a packet to a known destination :

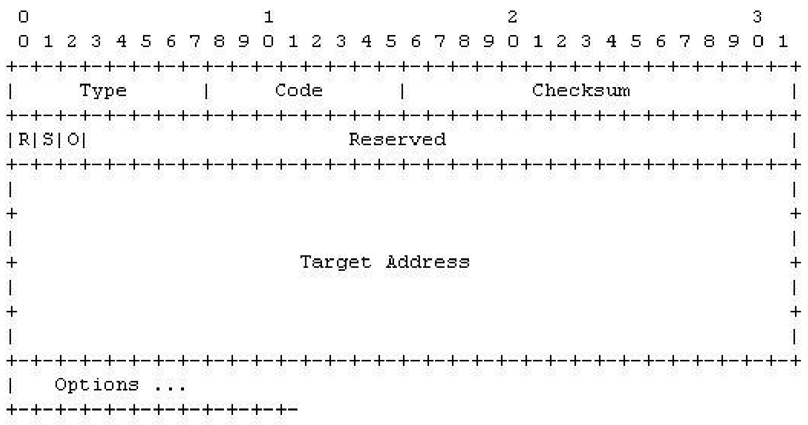
- It checks the destination cache. The destination cache associates a neighbor's address with every destination.
- The ND algorithm checks the Neighbors' cache for the corresponding Neighbor's address.
  If the Neighbors' cache does not contain the needed entry, a Neighbor Solication message is sent to discover the media layer address. These messages are ICMP control messages, they have the following format :



```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Reserved                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                       Target Address                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options ...
+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 24: Neighbor Solicitation Message Format - Source [2].

- The neighbour receives the Solication message and replies with a Neighbor Advertisement message.



```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|S|O|                     Reserved                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                       Target Address                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options ...
+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 25: Neighbor Advertisement Message Format - Source [2].

- The local nodes receives the Neighbor Advertisement message and updates its cache, it can now send its packet to the next hop on the packet's path.

The router discovery mechanism uses two other ICMP control messages : Router Solicitation message and Router Advertisement message. All these messages are part of the ICMPv6 protocol. The piggybacking of a monitoring context on these mechanisms would require finding an appropriate field in the ICMP messages. As we will see in the 2.2.2 section, the only convenient field would be the option field. Its maximum size in bytes is : 1280 - 40 (IPv6 header) - 24 (ICMP other fields) = 1216 bytes.

However RFC 2461 [2] specifies three options for Router Advertisement message and only one for the other messages. It is also said that future versions of the protocol may define new option types. As it is not clearly said that experimental options can be used in these messages it might be inappropriate and risky to introduce a new measurement option in a protocol that is not designed for it. Another reason opposed to a possible use of these new ICMPv6 messages is that ICMPv6 messages have the same general format as ICMPv4. There do not seem to be any work on ICMPv4 going in that direction, that is to say : using a new option field in an existing ICMP message in order to couple some performance measurements with the basic service provided by the ICMP message chosen.

ND is also based on another IPv6 new feature : native multicast. Indeed, ND uses ICMP messages, which is an upper-layer than IP. Thus Solicitation messages have to be sent to an IP address, and this address is a multicast IPv6 address : solicitated nodes address.
The big change compared to IP version 4 is that IPv6 defines a set of well-known multicast addresses that have to be listened to by every node. We will see if we can take advantage of this native multicast mechanism in the active monitoring section 2.2.3.

2.1.5 Path Maximum Transfer Unit (PMTU) discovery

In IPv6 fragmentation is not allowed at intermediate nodes, for a performance reason. Thus sending hosts have to perform the fragmentation if necessary because routers are not able to fragment packets that are too large for a link. The notion of Maximum Transfer Unit (MTU) is essential in IPv6 because it is important for a node to know the MTU of a path in order to use the best packet size and avoid ICMP “packet too big” error messages.

To discover a path MTU, a host will use the Path MTU (PMTU) discovery mechanism specified in the RFC 1981 [20]. If a host cannot discover a PMTU, it will have to limit the sending packet size to the lowest allowable MTU in an IPv6 network, which is 1280 bytes.

PMTU discovery mechanism is based on ICMP “packet too big” messages. Initially, the PMTU value for a path is assumed to be the (known) MTU of the first-hop link. Each time that the sending host will receive an ICMP packet too big message, it will be aware that the MTU used for this path is too big. The PMTU will be reduced until the correct delivery of the packet.
In a monitoring point of view, for many reasons PMTU discovery is obviously not suited for any special monitoring operation. Amongst other things PMTU discovery mechanism is not mandatory and its implementation can differ from a system to another. Another reason is that the only node on a path that will certainly receive the packet used to discover the PMTU is the first hop, and thus it does not provide an interesting monitoring approach of the feature.

2.1.6 Address auto-configuration and Duplicate Address Detection (DAD).

Another new IPv6 feature is the address auto-configuration. Auto-configuration allows IPv6 nodes to be Plug and Play. An IPv6 node only has to be plugged into an existing IPv6 network and it will be automatically configured without any human intervention.
IPv6 address auto-configuration aims are:

- To allow newly plugged nodes into a network to obtain a global IPv6 address.
- To allow nodes to obtain a new address in case of reassignment.

Address auto-configuration process is divided into 3 phases : generation of a link-local address, Duplicate Address Detection (DAD) mechanism in order to verify its uniqueness, and global address auto-configuration.

1. **Generation of a link-local address :**
   The node will first generate a "tentative address", this address has only a link-local scope and is generated by using the link-local prefix (FE80) combined with the interface identifier.
   The tentative address will only be partially assigned to an interface, in order to receive Neighbor Solicitation Messages and Neighbor Advertisement Messages.
2. **Duplicate Address Detection (DAD)** :
   This mechanism allows a node to check that its tentative address is not already used by another node.

   DAD is pretty simple, it is based on two types of ICMP messages : Neighbor Solicitation Message and Neighbor Advertisement Message.
   The DAD algorithm consists of sending a Neighbor Solicitation Message with an unspecified source address field to the tentative address chosen and listen to the interface for a certain time. During this time interval, three scenarios are possible :

   (a) A Neighbor Advertisement is received from the same address as the tentative address, meaning that the tentative address is another node valid address. Thus addresses are duplicated and cannot be used.
   (b) A Neighbor Solicitation Message is received, its source field allows to distinguish whether its part of the DAD mechanism or the ND. If it has an unspecified source field, it means that another node has chosen the same tentative address. Addresses are duplicated and cannot be used.
   (c) If nothing has been received for a certain time, the tentative address is considered unique and will be assigned to the interface as a valid address.

Duplicate Address Detection is a mechanism to allow nodes to obtain a locally unique address. As we have seen, nodes running DAD on an interface still have no valid address for this interface. Thus it would be pointless to try to combine any performance monitoring operation with DAD, knowing that nodes running it are still not capable of communicating with any other nodes.

Also DAD is, like Neighbor Discovery, based on ICMP messages. For the same reasons there would be no possibilities to take advantages of these messages in a monitoring purpose (see 2.1.4).

3. **Global address auto-configuration** :
   Until now, our newly connected node has acquired a new link-local unique address for its interface. The third step of the address auto-configuration process is the global address auto-configuration. There are actually two modes for global auto-configuration : stateless or statefull. The chosen mode will be dependent on the presence of a router on the link.

   (a) A router is active on the link
   When a router is present on the same link as the node expecting a global address configuration, it will communicate through Router Advertisements messages the auto-configuration mode. Special flags in the RA will specify whether the auto-configuration mode will be statefull or stateless.

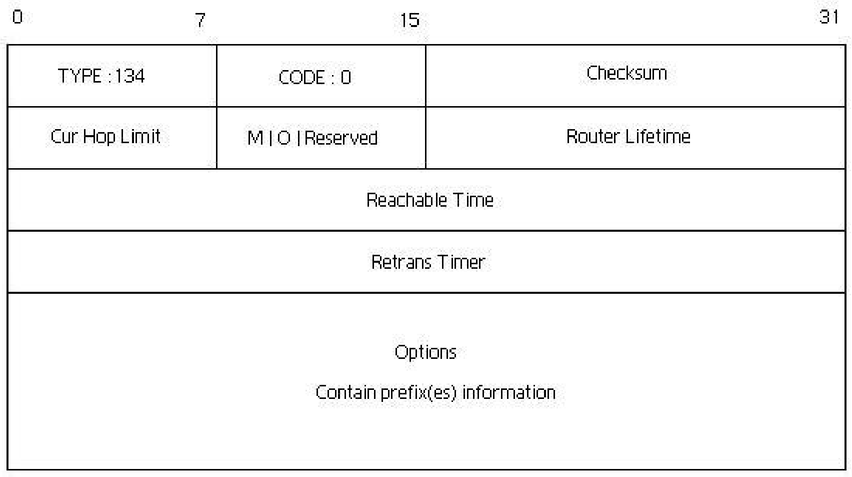


Figure 26: Router Advertisement Messages : Containing Auto-Configuration Information.

The "M" and "O" flags are respectively the ManagedFlag and the OtherConfigFlag. If the 'M' bit is set, it means that the node will have to run the statefull auto-configuration protocol. Otherwise if the 'M' bit is not set, the node will use the stateless auto-configuration

mechanism.
If the statefull mode is chosen, the node will have to run the statefull auto-configuration protocol for IPv6 : DHCPv6. The 'O' bit specifies that the node will have to use DHCPv6 to discover other parameters (but not for address configuration). In this framework we will not discuss DHCPv6, for more information about it see the RFC 3315 [25].
When the stateless mode is specified, the node will compute its global address by combining some local address information with some information about the network, respectively its link-local address with a prefix given by the routers in Router Advertisement messages (as seen in the previous figure).

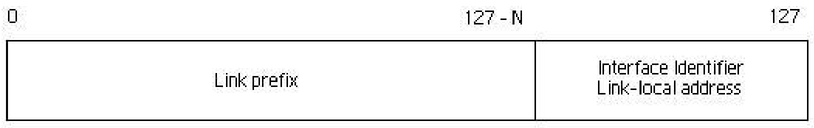


Figure 27: Stateless Auto-configuration : Global Address Construction.

For several reasons described in [22], the node could decide to ignore link prefix information. If stateless auto-configuration cannot be completed with obtaining of a global address, statefull protocol will have to be run.

(b) There is no active router on the link
If after a few solicitations the node does not receive any Router Advertisements, it means that there is no active router on the link. In that case the node will not be able to perform a stateless auto-configuration. It will then run the statefull auto-configuration protocol to try to obtain its global address. If statefull configuration fails, global address configuration will fail too, and the node will only be able to communicate on its link using its link-local address.

In a performance monitoring point of view, the analysis of the global address auto-configuration mechanism leads us to the same conclusion as for the DAD feature. Global address auto-configuration mechanism also takes place at the setup of a node on a network. At the setup time, the node still has no global address and no global visibility from the rest of the network. Thus coupling any performance monitoring methods with such a mechanism would be totally pointless.

2.2 Active monitoring over IPv6.

We have seen in the previous section 2.1.3 that it is possible to add some monitoring information in the extension headers. There are three choices which have their different advantages and concerns : the Hop-by-Hop header, the Destination header or a new extension header. In that section their purposes were to carry some information on one-way, and let the receiver collect all the piggybacked information to compute a measurement.

Active measurement can be performed the same way, the only difference is that active measurement context will be carried on synthetic traffic rather than piggybacked to the real traffic. This synthetic traffic will obviously contain the IPv6 protocol, but also a transport protocol. Most IPv4 active measurement functions, like the IP SLA measurement functions described in 1.1.3, are based on three well known transport protocols. Therefore a natural and basic approach to design measurement functions for IPv6 would be to use these transport protocols.

In this part we will discuss what IPv6 changes in the three main transport protocol : UDP, TCP and ICMP. We will also talk about IPv6 multicast, which can be used in combination of an active measurement.

2.2.1 Transport protocols : TCP and UDP

UDP and TCP are so widespread all over the world that it was an IPv6 prerequirement to left them unchanged in order to facilitate the upgrade from IPv4 to IPv6. However there are only some small modifications in both protocols. The first one is due to the suppression of the checksum in the IPv6 header. With IPv6, it has been decided that every upper-lay protocol had to compute and control a new checksum. The second change involves the new IPv6 jumbogram option. This option allows an IPv6 packet to contain more than the maximum 16-bit length fixed by the Payload Length field of the IPv6 header. But transport protocols such as UDP and TCP also have a 16-bit length field for their maximum data size. Thus they have to be able to handle the jumbogram option as well. So what changed ?

1. In UDP over IPv6 :

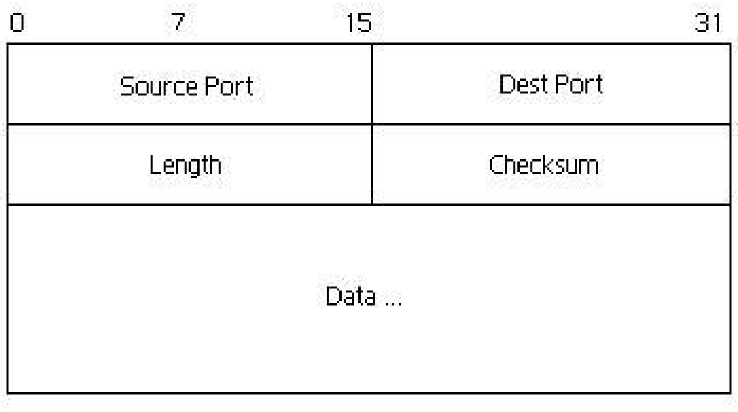


Figure 28: UDP Header Format

(a) The checksum is no longer an option, it becomes required and its method calculation is now a little different. It now takes into account a pseudo-header containing information from the IPv6 header.

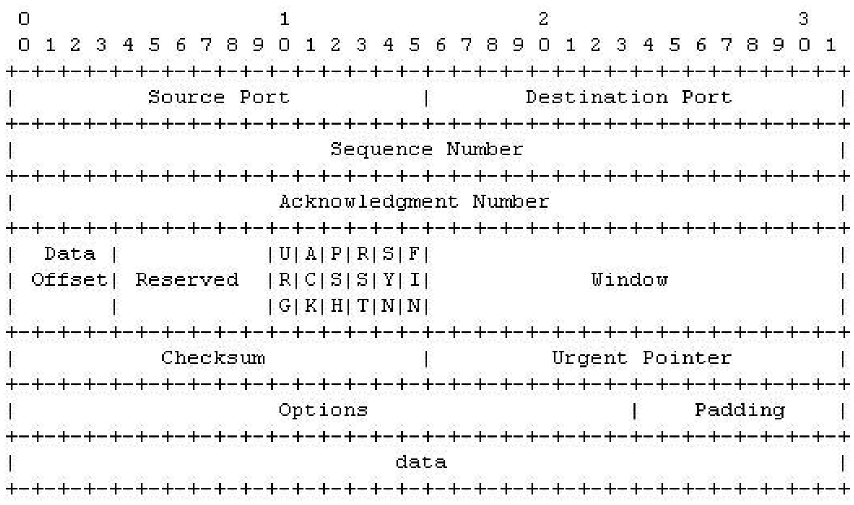
```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Upper-Layer Packet Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      zero                     |  Next Header  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 29: Pseudo-Header that has to be included in the TCP checksum for IPv6

(b) When the jumbogram option is used in the IPv6 header, UDP 16-bit length field still limit the UDP packet to 65 535 bytes.
RFC 2675 specifies the modification of UDP required by IPv6 to relax

that limit. UDP packets longer than 65 535 bytes may be sent by setting the UDP length field to zero and letting the receiver derive the actual UDP packet length from the IPv6 payload length.

2. In TCP over IPv6 :



```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 30: TCP Header Format - Source [37].

(a) In TCP the checksum was already mandatory, the only difference is that now it has to take into account the pseudo-header described in the figure above.

(b) TCP does not contain any field fixing a packet size limit, however it contains two 16-bit counters that require a special handle in case of jumbogram option.

i. The Maximum Segment Size (MSS) is negotiated at the connection establishment and is a 16-bit length field, which limits the largest TCP packet that can be sent to 65 535 bytes. When determining what MSS value to send, if the MSS is greater than 65 535, the value sent will be 65 535. When a 65 535 MSS is received, it has to be treated as infinity and the real value must be calculated thanks to the Path MTU discovery algorithm on the path to the TCP peer.

ii. The Urgent Pointer is a 16-bit offset indicating the start of urgent data when the URG bit is set. If both jumbogram option and urgent pointer are used, the RFC 2675 specified three different scenarios :

A. If the offset is less than 65 535, fill in the Urgent field and continue with the normal TCP processing.

B. If the offset is greater than 65 535, and the offset is greater than or equal to the length of the TCP data, fill in the Urgent Pointer with 65 535 and continue with the normal TCP processing.
C. Else the TCP packet must be split into two pieces. The first piece contains data up to, but not including the data pointed to by the Urgent Pointer, and the Urgent field is set to 65,535 to indicate that the Urgent Pointer is beyond the end of this packet. The second piece can then be sent with the Urgent field set normally.

The negligible changes in transport protocols allow for an easier migration for applications based on these transport protocols. This will be shown by an example in the third chapter, where we will describe the design of a simple measurement function prototype based on UDP and working over IPv6. Many IPv4 measurement functions based on these transport protocols could be extended to IPv6 in the same natural and basic way.

2.2.2 ICMPv6

Internet Control Message Protocol (ICMP) which was in version 4 has been reviewed for IPv6. It is an integral part of the IPv6 architecture and must be supported by all IPv6 implementations. ICMPv6 has been given the IP protocol number 58. The main differences with its predecessor are a better distinction between the possible ICMP messages (control or error), but also the addition of several new ICMP messages which are part of protocols such as the Multicast Listener Discover (MLD), the IPv6 successor of Internet Group Message Protocol (IGMP) in IPv4; or like Neighbor Discovery (ND, the successor of ARP.

ICMP header has remained unchanged :

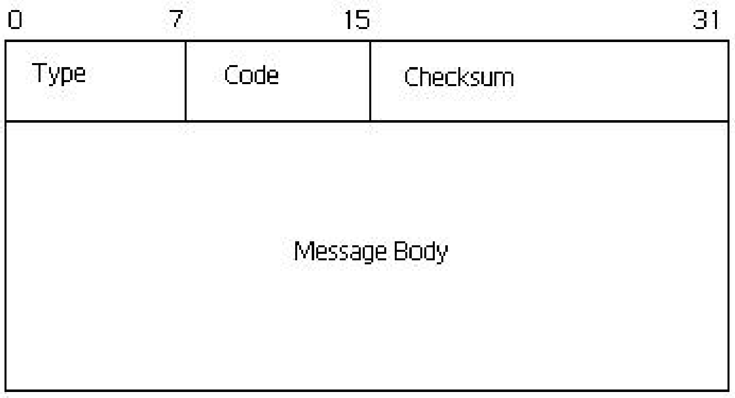
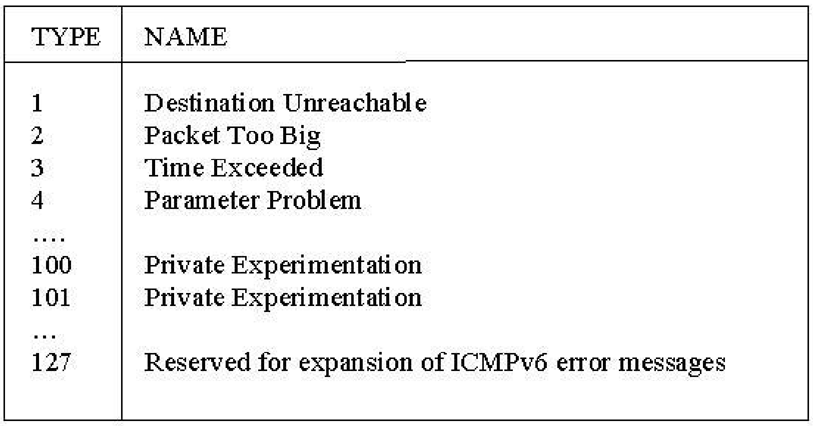


Figure 31: ICMPv6 General Format.

The type field determines the ICMP type of message, and thus the ICMP data format. There are now two types of ICMP messages :

- Error messages : Identified by a type number between 0 and 127.
- Information messages: Identified by a type number between 128 and 255.

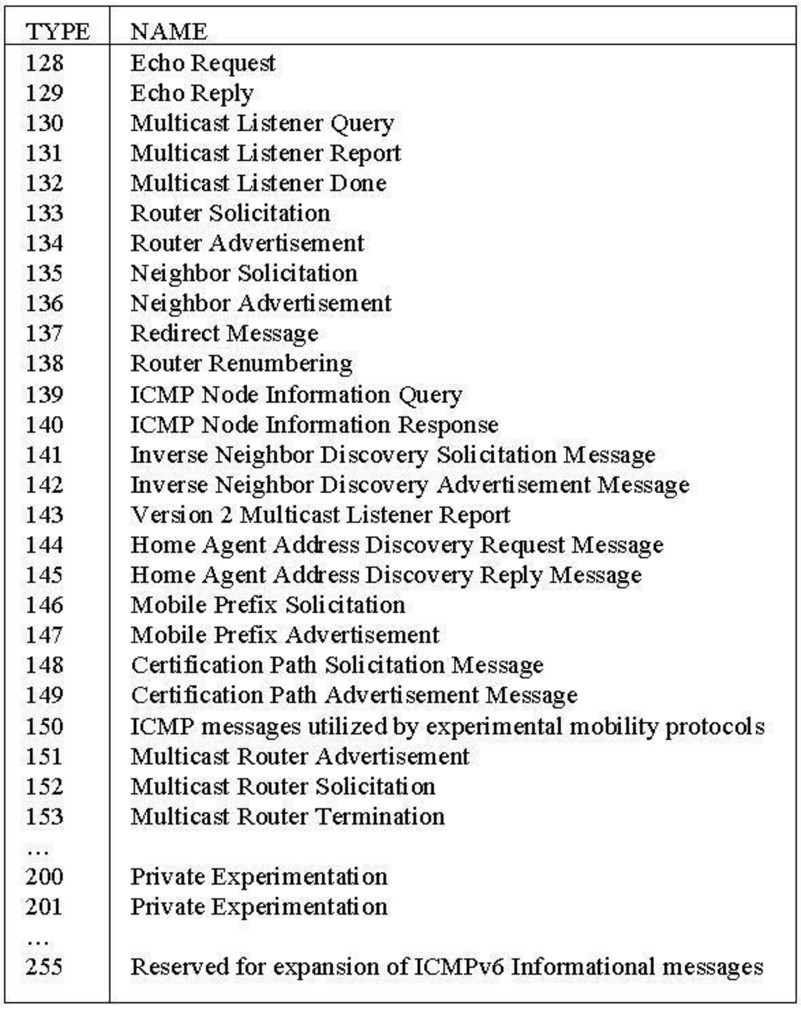
The following tables contain the currently defined ICMP type numbers [1]:



TYPE	NAME
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem
....	
100	Private Experimentation
101	Private Experimentation
...	
127	Reserved for expansion of ICMPv6 error messages

Figure 32: ICMPv6 Error Message Types.

[1]defined by the IANA and available at http://www.iana.org/assignments/icmpv6-parameters



TYPE	NAME
128	Echo Request
129	Echo Reply
130	Multicast Listener Query
131	Multicast Listener Report
132	Multicast Listener Done
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect Message
138	Router Renumbering
139	ICMP Node Information Query
140	ICMP Node Information Response
141	Inverse Neighbor Discovery Solicitation Message
142	Inverse Neighbor Discovery Advertisement Message
143	Version 2 Multicast Listener Report
144	Home Agent Address Discovery Request Message
145	Home Agent Address Discovery Reply Message
146	Mobile Prefix Solicitation
147	Mobile Prefix Advertisement
148	Certification Path Solicitation Message
149	Certification Path Advertisement Message
150	ICMP messages utilized by experimental mobility protocols
151	Multicast Router Advertisement
152	Multicast Router Solicitation
153	Multicast Router Termination
...	
200	Private Experimentation
201	Private Experimentation
...	
255	Reserved for expansion of ICMPv6 Informational messages

Figure 33: ICMPv6 Informational Message Types.

2.2.3 IPv6 native multicast addresses

In IPv6, multicast is natively supported by nodes. In a measurement point of view, native multicast should greatly facilitate the design of multicast measurement functions.
Another new feature brought by IPv6 are the native multicast pre-defined addresses. According to RFC 4291 [9] IPv6 nodes have to listen to a set of well-known pre-defined multicast addresses :

- The All-Nodes multicast addresses

- The Solicited-Node multicast address for each of its unicast and anycast address
- Multicast addresses of all other groups to which the node belongs.

IPv6 routers are required to recognize all multicast addresses that a host is required to recognize, plus the All-Routers multicast addresses.

Here are especially the more interesting pre-defined multicast addresses for our purpose :

1. All nodes address FF02:0:0:0:0:0:0:1 (local scope)
2. All routers addresses :
   FF02:0:0:0:0:0:0:2 (local scope) and
   FF05:0:0:0:0:0:0:2 (site scope)

These native multicast addresses for nodes or routers can be used to contact any nodes or routers in a local scope, or contact any routers in a site scope. In particular we could combine their use with an active measurement operation in order to query every node or every router in the local or site scope. In the next chapter we will describe the design of an UDP-echo over IPv6 operation using a multicast address.

3 Exploiting IPv6 features for designing measurement functions.

The aim of this chapter will be to describe the design of a monitoring operation measuring Round-trip Delay and One-way Delay metrics by using several specific of the previously seen new IPv6 mechanisms .
The first step of our description will be the definition of a basic UDP-Echo operation over IPv6. Then we will discuss the combination of this basic operation with the IPv6 native multicast mechanism to get an automatic multicast monitoring operation.
Finally we will talk about a smarter way to improve our UDP-Echo functionalities, by combining it with one of the new IPv6 features : TLV-Encoded options in Extension headers.

3.1 Basic solution

As said previously, one objective of this work is to show that measurement functions can be used over IPv6, and to prove that it can be done in a basic way by an example : the implementation of a simple measurement function. For this purpose we have chosen to design a simple UDP - Echo prototype operation working on IPv6. This operation will be capable of measuring one metric : the Round-Trip Delay (see 1.1.1) between two end systems . It will be an active function, working in two phases : a control phase to initiate the test phase, and a test phase to establish the measurement.
UDP - Echo function described in this section is a simple and rather naive active function, that does not take into account several issues like security issues. UDP - Echo has two roles in this work: firstly to show that existing monitoring approach on IPv4 can easily be extended to IPv6, and secondly to constitute the base function that will be combined with specific IPv6 features for our further improvements.

3.1.1 Defining a basic IPv6 UDP-echo operation

The test unfolding will be divided into two phases. The first one is a control protocol datagram exchange that will establish parameters for the second phase : UDP-echo request.

a) First phase : control protocol

The control protocol is a simple UDP exchange between a source and a destination, where the source is the host interested in performing measurements

and the destination is the target of these measurements[2][3]. The control protocol purpose is to allow a source node running a client to contact a destination running a server for a particular request.
This request will be encoded in the UDP data payload in a Type Length Value (TLV) format[4], and will specify a predefined measurement operation that the sender would like to perform.

The server entity will have to be listening on a given control port : port 5354. If a server receives a request from a client node on that port, it will have to analyse the TLV-encoded UDP data content and determines whether it is able to handle the requested operation or not. If the server entity is capable of answering the operation request, the server will reply to the control request and specify a port number that the client will have to use for the second phase : the test phase. This control protocol could be used for different operation types, but in this work it will only be used to establish one kind of measurement operation : an UDP-Echo operation.

Control datagram data will be composed of the following Type Length Value format :

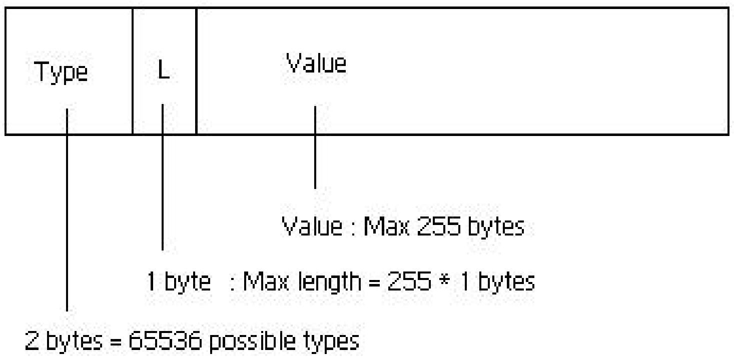

Type
L
Value
Value : Max 255 bytes
1 byte : Max length = 255 * 1 bytes
2 bytes = 65536 possible types

Figure 34: UDP Measurement Options TLV Format

The chosen TLV format will be used in both Control and Test datagrams. A 2-byte Type field allows the definition of up to 65 536 different type of options, and in this work we will define a few option types. Each Option type can have

[2]The terms "Source" and "Destination" will be used as shortcuts to refer to these definitions in the remainder of this work

[3]In the implementation part, the Source will have to run a Client application, and the Destination a Server. For these reasons the terms Client and Server might be used in place of Source and Destination in the remainder of this work.

[4]The TLV format used in the control and test datagrams is different of the TLV format used to define options in the IPv6 extension header

a maximum value length of 255 bytes, fixed by the Length field.
Two TLV option types have been defined for the Control protocol

1. Type “MC” for Measurement Control.
   Its 2-byte value is 0x4D43, which is the ASCII value of “MC”. This option will be used as the UDP datagram option content by a node establishing the control protocol with its target. The value field will specify the measurement operation that the source would like to perform with the destination of the datagram.
   In our case, an UDP-Echo operation will have a value field of “UDP-ECHO”, and thus a length field of 8.

2. Type “PN” for Port Number.
   Its 2-byte value is 0x504E, which is the ASCII value of “PN”. The option will be placed in a monitoring control reply in order to communicate a port number where the requested measurement test can be performed.

In particular, a node interested in the establishment of an UDP-ECHO measurement test with another node will have to send a datagram to that node on the port 5354. This datagram will contain a TLV-Encoded Monitoring control option, of type “MC”, its length and its value : “UDP-ECHO”.
The server listening on port 5354 will receive the request and process it. If it can handle an UDP-Echo test operation, it will send a reply to the client containing its “UDP-Echo” request, plus a TLV-Encoded Port Number option, of type “PN”, its length and the value of the port to use.

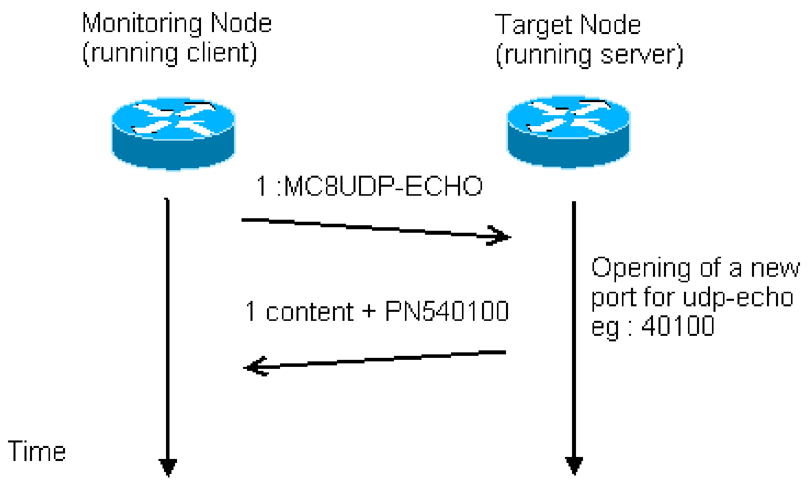


Figure 35: UDP-Echo : Control protocol.

b) Second phase : UDP-echo test

An UDP-Echo test phase consist of a two UDP datagram exchange : one echo request and one echo reply. New UDP TLV-Encoded option types have been defined for the particular UDP-Echo test operation :

1. Type "MR" for Measurement Request.
   Its 2-byte value is 0x4D52, which is the ASCII value of "MR". The option specifies the operation performed by a measurement request, to which the destination is supposed to answer. We have defined two possible values for our UDP-Echo request operation :"UEBASIC" or "UETIMESTAMPS" respectively UDP-Echo Basic, or UDP-Echo with destination timestamps in order to have a more accurate result of the Round-trip Delay.

2. Type "RE" for Request Echo.
   Its 2-byte value is 0x5245, which is the ASCII value of "RE". The aim of this option is to define the request echo type according to the measurement type asked. Possible values of an UDP Request echo are "UEBASIC" or" UETIMESTAMPS" according to the MR value. A type value of "UETIMESTAMPS" will require the presence of two more option types in the datagram : "IT" and "OT".

3. Type "IT" and Type "OT", respectively for Incoming Timestamp and Outgoing Timestamp.
   Their values are 0x4954 and 0x4F54, the ASCII values of "IT" and "OT". These options are combined to the Echo Reply of value UETIMESTAMPS in order to provide a better accuracy in the Round-trip delay calculation. "IT" and "OT" options both contains a 8-byte length value, which is a time value expressed as seconds and microseconds since 00:00 Coordinated Universal Time (UTC), January 1, 1970. The difference between these timestamps show the processing time between the reception of a Measurement Request packet, and the generation and sending of an Echo Reply packet.

More option types could have been defined in order to enhance the test function. For example :

- An option type "SN" for Sequence Number. A sequence number would identify the request in case of multiple requests. For example it would allow the calculation of the jitter metric.
- An option type "AD" for Arbitrary Data, would allow the monitoring station to specify a certain size for the test datagram.

Once it has completed the first control phase, a node interested in an UDP-Echo measurement will know if it can start or not an UDP-Echo test phase with the

chosen destination. If a control protocol reply has been received, the client has been given a port number to send its UDP-Echo request to.
The test phase will thus be initiated by the client.
The client will send a datagram with a TLV-Encoded Measurement Request Option to its target. It can choose among the two possible values : UEBASIC or UETIMESTAMPS. The datagram will be sent to the port given by the control protocol.
At the reception, the server will process UDP data and send a reply corresponding to the measurement request. The datagram will contain at least a TLV-Encoded Request Echo option with the same value than the Measurement Request.

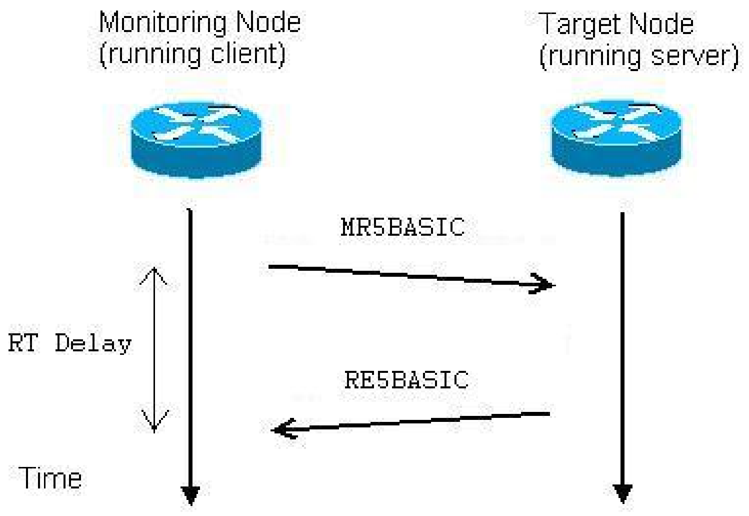


Figure 36: UDP-Echo Basic Operation

If the chosen mode is UETIMESTAMPS, the datagram will also contain the TLV-Encoded timestamps options : IT and OT.

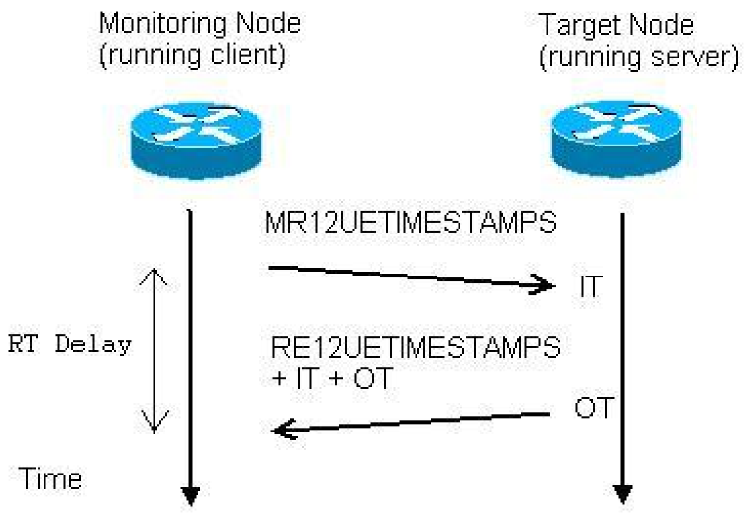


Figure 37: UDP-Echo Operation using Timestamps Option to remove the target processing time.

It is important to note that our UDP-Echo measurement function does not require any time synchronization between end-entities, the Round-trip Delay calculation will only be computed thanks to relative times on the client side. The client will measure the interval between the Measurement Request departure and the reception of the Request Echo. When Timestamps mode is used, the client will remove the target processing time of the Round-trip Delay measured by computing the difference between OT and IT.
Thanks to the timestamps option, the function could easily be modified in order to measure the One-way Delay. It could be achieved by computing the difference between the Incoming Timestamp at the destination and the packet departure time at the source, however it would require perfect time synchronization between end systems.

3.2 Combining the basic solution with native multicast

Now that we have an operational active measurement operation working over IPv6, it is interesting to combine it with the new IPv6 pre-defined multicast addresses in order to have an automatic monitoring tool for every nodes listening on the chosen multicast address.
Using these multicast addresses with our UDP-echo operation, we can perform a complete monitoring test with all nodes on the local link, or with all routers

on the local link or in the site scope. The three possible multicast addresses we can use are the following :

1. All nodes address FF02:0:0:0:0:0:0:1 (local scope)
2. All routers addresses :
   FF02:0:0:0:0:0:0:2 (local scope) and
   FF05:0:0:0:0:0:0:2 (site scope)

The multicast UDP-Echo operation is the result of the combination of the UDP-Echo operation defined in the previous section with one of these multicast addresses.

Typically a node interested in a classic UDP-Echo operation will start the control protocol, that is to say : send a Measurement Control to a chosen destination address to be measured. The multicast UDP-Echo consists in using one of the predefined multicast destination address rather than a unicast address.
For example, if a client uses the All-nodes multicast address in its Measurement Control datagram, every listening node that has a server entity active and listening on its port 5354 will answer to this request with a Measurement Control reply. The client will receive as many Measurement Control replies as there are nodes interested by measurements on the local link.

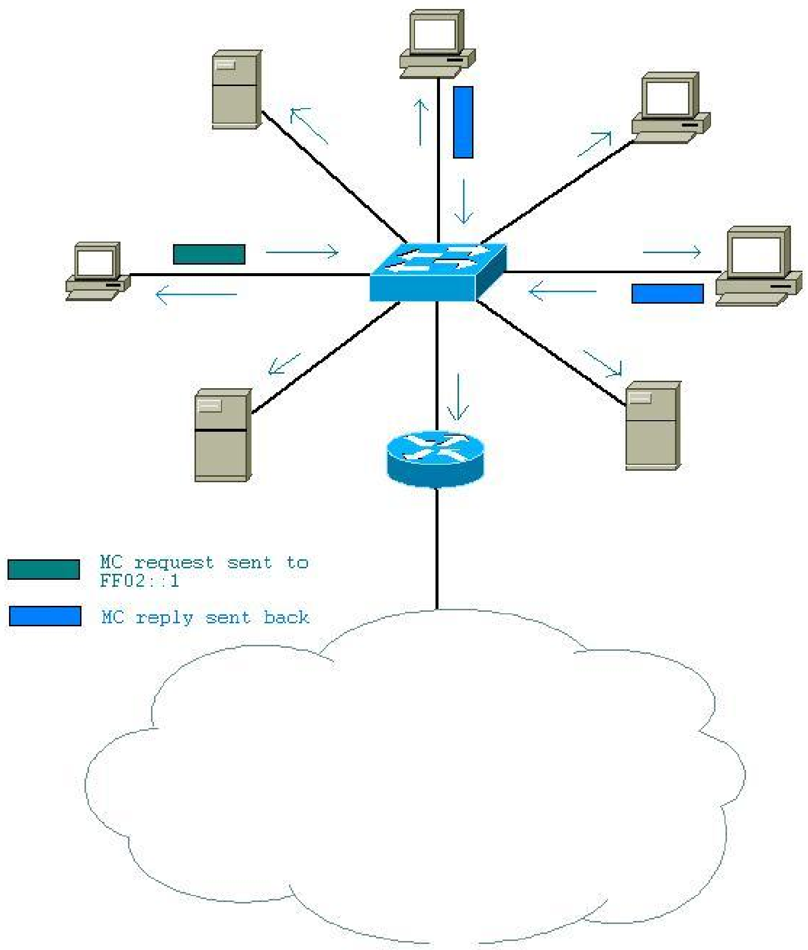


Figure 38: Multicast UDP-Echo Operation using the All-node multicast address : control phase.

Therefore it will be able to initiate the test phase with all of them, using their unicast address and the specified port contained in the Control reply.

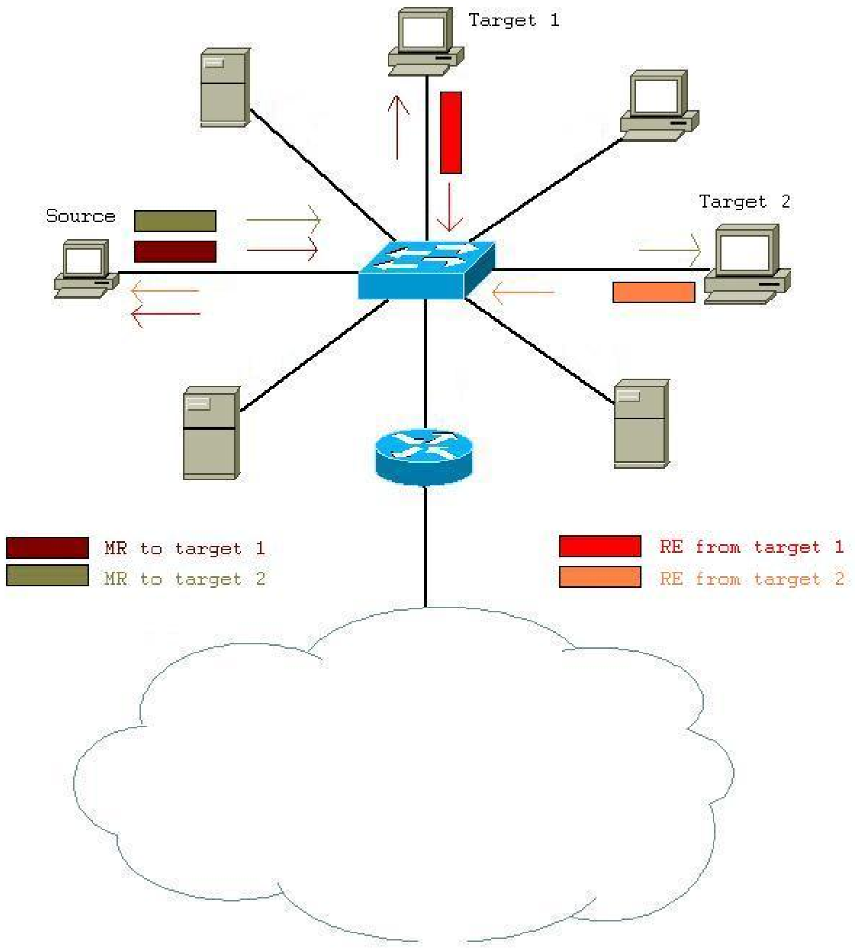


Figure 39: Multicast UDP-Echo Operation using the All-node multicast address : test phase.

3.3 Thinking of a smarter solution

Through our IPv6 survey we have distinguished two IPv6 natural mechanisms providing a smarter way to perform some performance monitoring.

The first one consists in using some existing IPv6 extension headers, and insert new TLV-Encoded option fields in it in order to carry a measurement context. Two extension headers can be used in that way : the hop-by-hop extension header and the destination option header. We have seen that some work has already been achieved on both of these headers, mainly in a semi-passive monitoring approach.

The second way would be to create a new measurement header for IPv6. We have seen that an interesting proposal has already been made, but abandoned. We have also seen that a whole new extension header dedicated to measurements might bring many more features than what the 64 different possible TLV options allow. The major difficulty of designing such a header would be to convince the whole Internet world of the necessity to standardize and deploy it. This would require from the new header to provide major advantages than the other solutions.

Both solutions have thus their advantages. The new option types in the existing Destination extension headers seem to be a smarter solution for an end-to-end two-point measurement. Theoretically, it should work in multi-point mode too by a combination with the routing header to specify different intermediary destinations, but we have seen that in practice the routing header might be ignored. Using the Hop-by-Hop Options header allows not only two-point measurement mode, but also multi-point mode. It introduces two other concerns : it is more easily identifiable because it is processed by every nodes, and it also lightly increase the processing time of every node on the path.

An interesting solution would be to gather the powerful features brought by a complete active measurement method based on the transport protocols, and the flexibility of the TLV-encoded options to carry some network indicators.
The role of the active measurement will be double : perform an active measurement of different metrics between two end-systems, and allow the collect and send back of the TLV indicators at the destination .
TLV-Encoded options interest will be to extend the active monitoring measurement functionalities from a two-point mode to a multi-point mode measurement. Combining a powerful active protocol such as the One-way Active Measurement Protocol (OWAMP) described in 1.1.3, which is already ready for IPv6, with a piggybacking system in the TLV options would permit the performing of many measurement functions, all along a delivery path. However OWAMP is a rather complex protocol, which would require a lot of time to be implemented and combined with the TLV options. That is why in this framework we have chosen to combine the UDP Echo simple active protocol with the TLV solution.

3.3.1 Combining an existing IPv6 extension header and UDP Echo

UDP-Echo is a basic prototype of a measurement function that allows performing Round-trip Delay measurements between two end-systems over an IPv6 network. So why would it be interesting to combine it with IPv6 TLV options ? Because the resulting function will have an interesting capability : it will be able to measure some metrics all along a delivery path.
In particular in this framework, it will be capable of measuring two more metrics : the One-way Delay and the One-way Path Delay.
Indeed, the use of extension headers will allow the piggybacking of some indicators at every hops on the delivery path, letting us compute the delay between each hop along the path.
Another of reason of its interest is that, though UDP Echo is a rather naive protocol, it is has a similarity with OWAMP, it is also based on transport protocols,

and divided into two phases : Control and test. Thus implementing a prototype of this smarter function will give us a good idea of what the combination of TLV options and OWAMP could provide.

3.3.2 Hop-by-hop Options or Destination Options header ?

A choice had to be made between the two possible TLV Options carriers, Hop-by-hop Options and Destination Options. This choice has taken into account the different headers properties, and in particular their advantages and weaknesses for our purpose.
The Destination header seemed to be the cleverer solution at first glance for two main reasons :

- It should only be processed by the destination of the packet.
  Hence the Destination Options header will remain an opaque data payload unit for every router on the delivery path that is not a destination of the packet . Thanks to that property, measurement packets will be far less identifiable by intermediary nodes. Another interest of that property is that intermediary nodes will not have to spend some processing time on the Destination Header, and thus not increase the global travel time.
- The Routing header can specify intermediary destinations.
  Each destination record present in the RH will become the destination of the packet through its delivery, letting intermediary destination process the Destination Options header. It allows the calculation of some metrics along a path, but it also permits the measurement of a particular subpath.

In theory the Destination Header seems to be the best choice for our purpose, nevertheless in practice the solution is confronted with a major concern.
The issue does not concern the Destination Options solution in itself, but rather the use of the Routing Header as a way to extend Destination Header solution features. Based on the experience of IPv4 LSRR header which is very similar to IPV6 RH, we have many reasons to think that IPv6 Network actors might force their routers to ignore the RH option.
If it was the case, the Destination Options solution would no longer allow the handling of multi-point measurements. However our primary interest in using TLV-Encoded options resides in the possibility of performing multi-point measurements, which can expend the features of our active measurement function. Therefore it is essential to have a certainty about the ability of the chosen mean for multi-point measurements.

For that reason the Destination Header solution had to be discarded, and the choice was made to use the Hop-by-Hop Options Header as a measurement indicators carrier. Hop-by-hop Options properties have the following advantages for our function :

- The header is processed by every intermediary node.
  It makes this solution simpler than the Destination Header. While in the DH it was required to specify each intermediary node in the Routing header, here the simple use of a TLV-Encoded option in an IPV6 Hop-by-hop Header will allow every node on the packet delivery path to be aware of the measurement context, and participate in the measurements.

- Options can be used even when nodes are unaware of their meaning :
  The solution is also flexible thanks to the first two bytes of the Option Type, unrecognized options types can be ignored by a node processing the Hop-by-hop header. Thus multi-point measurements can be performed on every path without examining whether intermediary nodes are measurements capable or not.

The first property is also the reason of some drawbacks.

- Firstly, the Option type field is visible to all nodes on the path, and so it could be an easy way to idenfity measurement flows and possibly adapt the forwarding policy in consequence.

- And secondly the addition of an option that has to be observed by every hop will lightly increase the global delivery time, which is what IPv6 designers wanted to avoid by defining the new IPv6 Extension header mechanism. However this is only a minor issue, because intermediary nodes not aware of the measurement option will only have a 8-bit Option type field to process and can then skip the rest of the option if it is not recognized. The simple observation of a 8-bit new option type is not relevant to performance issues. But the processing of the option in intermediary nodes aware of its use will have to be optimized for performances.

In conclusion, the Hop-by-hop Extension Header seems to be the more appropriate extension header to carry our measurement indicators, allowing for a One-way Path metric calculation. Its combination with an active measurement function will allow the design of a smart and flexible solution capable of measuring different metrics, not only between two end-systems but also between hops along the path.

- **Smart**
  The simple addition of a Hop-by-Hop Option to its measurement packet will greatly extend the active measurement functionalities. It will be processed by every intermediary routers, and automatically collect indicators on the measurement capable routers. The active measurement will permit the send back of the indicators to the source, in order to compute the results locally.

- **Flexible**
  The choice of an active method allows measurements at anytime, even when there is no actual data traffic between the measured entities. The

Hop-by-Hop Option can be used on any path. Even if routers are not capable of handling the Option, the two first bit of the option will let them know to ignore the option and continue the processing.

In particular the Hop-by-hop Extension Header will be perfectly suited for recording intermediary nodes timestamps needed for a One-way Path Delay calculation. The next section will describe a new TLV-Encoded option that will be used for this special purpose.

3.3.3 Using Hop-by-hop Options to carry a One-way Path Delay option.

Measuring the One-way Path Delay metric will require the definition of a new TLV-Encoded option, that we will call One-way Path Delay Option.

The One-way Path Delay Option measurement will allow the gathering of intermediary nodes timestamps along the path. These timestamps will represent the absolute time of arrival of the measured packet at the incoming interface. As some nodes might not be capable of processing such an option it is also needed to record nodes addresses in the option. The gathered records will allow the One-way Path delay Calculation at the destination system.

The definition of a new option requires the attribution by the IANA of a new Option Type number. Our new Option definition will only be used for an experimental work and will obviously not lead in the filling of a request form at the IANA. Unfortunately IANA does not specify any option type values for experimental purpose, thus the option type value chosen for our new One-way Path Delay Option (OWP[5]) will be arbitrarily chosen based on TLV options values defined in some other works encountered through our survey.

[5]OWP will be used as a shortcut to design the One-way Path Delay TLV-Encoded option in the remainder of this work

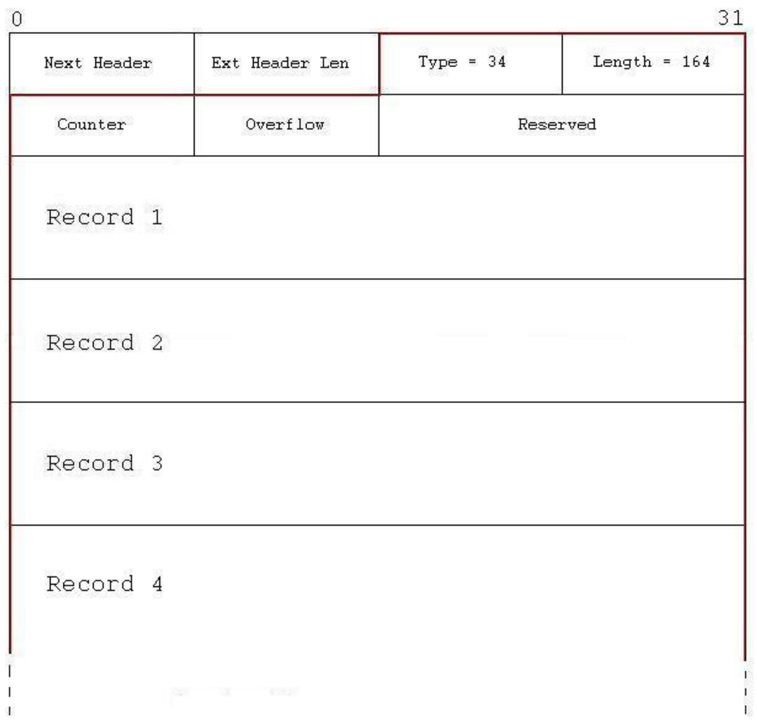


Figure 40: One-way Path Delay TLV-Encoded Option using Hop-by-hop Extension Header

The general format of the OWP option is very similar to the One-way Delay option format defined by D.P Pezaros in his work. The difference is that OWP design must allow multi-point measurements which require a record for each participating node. Here is a description of the different fields of the One-way Path Delay option:

- **Option Type = 00100010 (34)**
  The first two bytes 00 mean that the option must be ignored if it is not recognized. The third bit to 1 means that the data can change en-route. Remaining bits are set to 00010, for a value of 2, which is a simple increment of D.P Pezaros OWD option which has a value equals to 1.
- **Option Length = 164**
  The option has a fixed 164 bytes value size. 4 bytes are used for three different fields described below, and the other 160 bytes are used to store the maximum 20 possible records.
- **Counter**
  The counter field indicates the number of data records that are currently

stored in the option. It allows intermediary nodes to add their record at the right position. The counter will have a maximum value of 20.

- **Overflow**
  The overflow field indicates whether some nodes weren't able to insert their data record in the option because the maximum records number was reached.
- **Reserved**
  Reserved for future use.
- **Records :**
  The option allows to contain a maximum of 20 records, each record is a 8 bytes storage space that can contain two intermediary node indicators : two 32-bit timestamps.

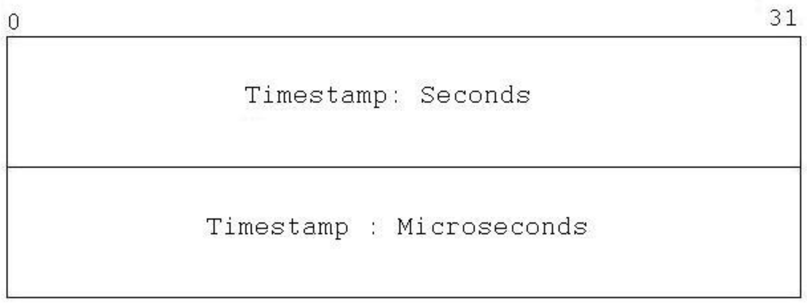


Figure 41: Option Path Delay - Record field format.

The record indicates the arrival time of the packet at the incoming interface, expressed as seconds and microseconds since 00:00 Coordinated Universal Time (UTC), January 1, 1970.
The first 32-bit field represents the seconds, and the last 32-bit the microseconds.

It would have been interesting to store one more information about intermediary nodes in the record field : their incoming interface IPv6 address. The address stored with the timestamp would give complete information about the route taken and the time elapsed between each nodes.
Without that information, two scenarios are possible:

- If all nodes on a delivery path are participating in a OWP measurement, it is possible to learn the IPv6 addresses of the nodes whose timestamps have been gathered by performing a *traceroute* on the same path.

- However if some nodes are not participating in the measurements because they are not OWP capable, the source will have to know which intermediary nodes are OWP capable to be able to find their IPv6 addresses.

These requirements decrease the flexibility of the OWP option, so why could not we simply store IPv6 addresses in the OWP data records ?
Unfortunately TLV-Encoded options have a maximum 255 bytes length, which would allow to store only ten 24-byte records containing a 8-byte timestamp and a 16-byte IPv6 address.
There are other alternatives, for example a new option dedicated to intermediary nodes address could be defined, but this would only let us a 15-record data space. Another alternative would have been to place two times the OWP option containing timestamps and addresses space, in order to carry the 20 possible records.

3.3.4 UDP-Echo with One-way Path Delay Option

The aim of this section is to describe the combination of our UDP-Echo prototype function with a One-way Path Delay Option in the Hop-by-hop header. This new function will be designed to perform an active measurement based on UDP-Echo, and carrying a One-way Path Delay Option in the Hop-by-hop Options header. We will call it the "UDP-Echo OWP" function for UDP-Echo One-way Path Delay Option. Such a function will allow the measurement of the Round-trip Delay metric, but also of the One-way Path Delay.
The basic UDP-Echo operation is divided into two phases, but only the second phase will have to be modified to insert the OWP option.

The first phase, or control phase, will be performed as it is described in the section 3.1.1. The monitoring control request value of "UDP-ECHO" will remain unchanged, which mean that a listening server will not see any differences between a basic UDP-Echo monitoring control request and an UDP-Echo OWP request. The definition of a different monitoring control value type is not necessary because, as we will see in the description of the test phase, it is the OWP option itself that will trigger a special handling from the server for its treatment.

The second phase, called test phase will be modified to take into account the new OWP option. The UDP test packet will have exactly the same format than in the basic UDP-Echo, but the client will now add the Hop-by-hop Extension Header between the IPv6 main header and the UDP test packet.
On the destination side, the server entity will have to process every Hop-by-hop options and check for an OWP option presence. If the OWP has been found, its records will have to be returned to the source thanks to the UDP reply. These records will be stored in a new TLV-Encoded option[6] called OP option.

[6]The TLV format discussed here is the experimental format described in the section 3.1.1 and NOT the TLV format for IPv6 Extension headers

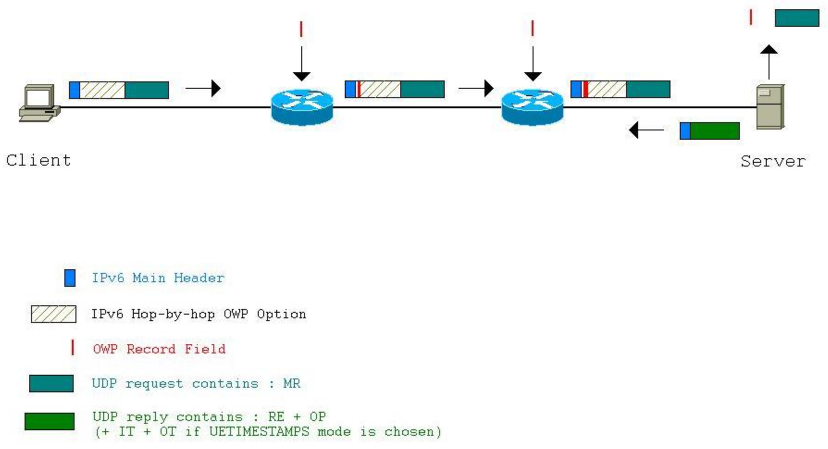

Client
Server
IPv6 Main Header
IPv6 Hop-by-hop OWP Option
OWP Record Field
UDP request contains : MR
UDP reply contains : RE + OP
(+ IT + OT if UETIMESTAMPS mode is chosen)

Figure 42: UDP-Echo OWP test phase

As shown on the figure, the OP option will be inserted in the echo reply, just next to the Request Echo option (RE), even if UETIMESTAMPS mode has been used. This choice does not seem to be important in the defined model, but it will be used further in the implemented model.
The OWP option use requires two properties of the intermediary nodes :

- First, every router on the way should be OWP capable.
  It means that intermediary routers should implement a procedure to be run when the OWP option is found in Hop-by-hop Extension Header of an incoming packet. The procedure will be the following :
  - Check the OWP counter field value.
  - If its value is less than 20, insert the packet arrival timestamps at the good position (which can be determined thanks to the counter value and the records size). Increment the counter field value.
  - Else if its value is equal to 20, increment the overflow field value.
  - Continue the processing of the Hop-by-hop Options.
- And secondly, this measurement requires the perfect time synchronization between the measured hops. However there are different ways to obtain an accurate time synchronization between hosts, and they will not be discussed here because it is outside of this work's scope.

4 Implementation and test.

4.1 Development of a measurement tool.

The next section will describe an implementation of a tool regrouping functions described in the last chapter. Many features have currently been implemented, but unfortunately not all. The tool has been divided into three entities : the client, the server, and a module for intermediary nodes participating in measurements. The client and the server have been both completely implemented, but the procedure to be run on routers among a measured path is currently not implemented.

4.1.1 Environment.

The developed tool is a Client - Server application, coded in C/Unix. It can be compiled and executed on Linux distributions. The version of the kernel must be at least 2.6.14 or newer in order to support the IPv6 advanced API.

The application is based on the basic socket API for IPv6 [38] and also on the IPv6 Advanced API [23] which has to be supported by the kernel. However the most recent RFC discussing the Advanced API, RFC 3542, does not seem to be implemented on the tested kernel (2.6.14). Thus we referred to the previous RFC, number 2292.

4.1.2 Tool features.

The developed tool regroups the following functions :

- **UDP-Echo**
  UDP-Echo can measure the Round-trip Delay between two hosts, using the basic mode or the timestamps mode (see 3.1.1).
- **Multicast UDP-Echo**
  This operation is described in the section 3.2, it allows a multicast Round-trip Delay measurement on a local network, using the All-node or All-router addresses.
- **UDP-Echo OWP**
  UDP-Echo One-way Path is a smarter function, capable of computing the One-way Path Delay metric between two systems. It is described in 3.3.4.It requires a perfect time synchronization between hosts on the measured path, and the presence of a dedicated procedure on every intermediary hosts.

4.1.3 Structure of the Implementation.

The application is divided into three 3 parts :

1. **The Client**
   The client is the application to run on a system in order to perform a particular measurement between the local host and a chosen distant host. Different parameters can be specified at the execution of the client, such as the type of measurement, the target and some other options. The usage of the possible options will be given in the next section.

   The figure 43 shows an UML Activity Diagram of the Client general behaviour:

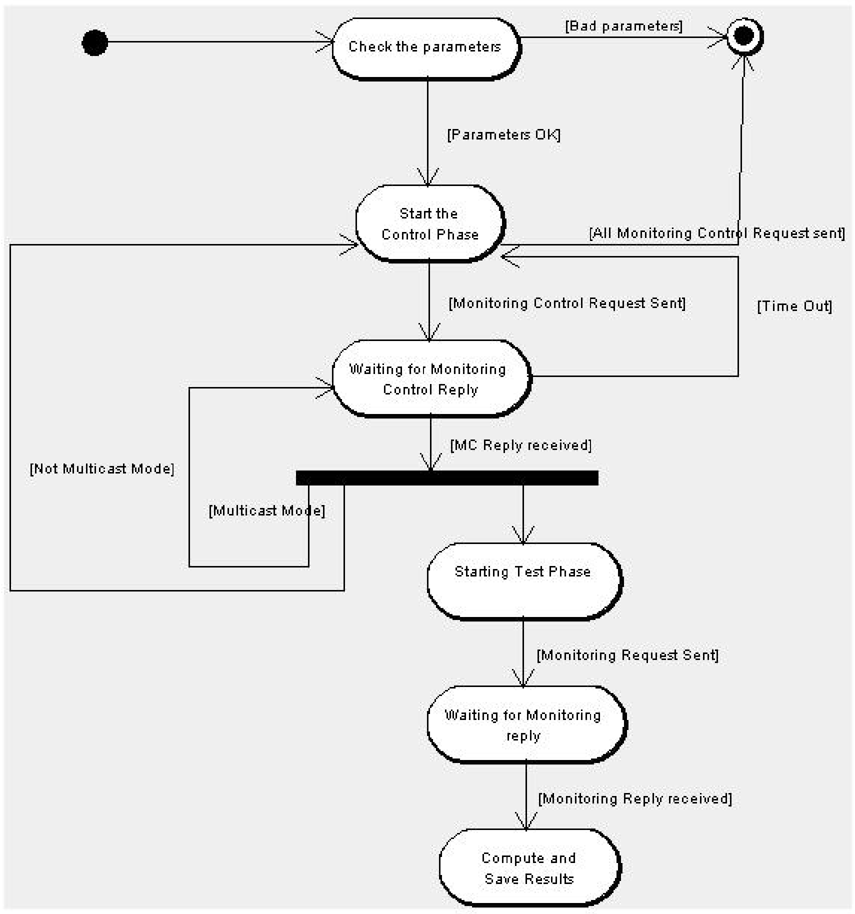

Check the parameters
[Bad parameters]
[Parameters OK]
Start the Control Phase
[All Monitoring Control Request sent]
[Monitoring Control Request Sent]
[Time Out]
Waiting for Monitoring Control Reply
[MC Reply received]
[Not Multicast Mode]
[Multicast Mode]
Starting Test Phase
[Monitoring Request Sent]
Waiting for Monitoring reply
[Monitoring Reply received]
Compute and Save Results

Figure 43: Client General Behaviour.

Processes

A client is a multi-process application where there are two types of processes : Main process and Child process.

The father process (or main process) task is to check parameters, and start the control protocol each time interval between measurements as specified in argument (see 4.1.4). After the control protocol request has been sent, it waits for a control reply from the target, and creates a new child process when it is received. If multicast mode is chosen, the main process will create a new child process for each control reply received during the time interval between two measurements. The father process will exit only when all tests have been performed and its child have been closed or when the program is closed using Ctrl+c.

The child process role is to analyse the received control reply, and perform a measurement test toward the target on its chosen port. It first sends a monitoring request of the desired type to the target, and listens for its answer. If no answer is received after a certain lap of time, the request is considered as a time out. If an answer is received, it is analyzed and the Round-trip Delay is computed and append to a file in the client directory. After the measurement has been completed or timed out, the child process shutdowns.

Files

The client application composed of two main files : main.c and client.c.

(a) **main.c**
This file contains the program entry point, its role is to check the value of parameters and then call the client.c function that will start the measurement protocol.

(b) **client.c**
This file contains every function needed to perform an UDP-Echo operation on an unicast address, or on a multicast all-node or all-router address [7]. It is also capable of starting an UDP-Echo OWP operation.

2. **The Server**
The server is the application to be run on a every host that has to be measured by client entities, the server is a multi-process application listening to the chosen measurement control port : port 5354.

The figure 44 shows an UML Activity Diagram of the Server general behaviour:

[7]when the option has been chosen (see 4.1.4)

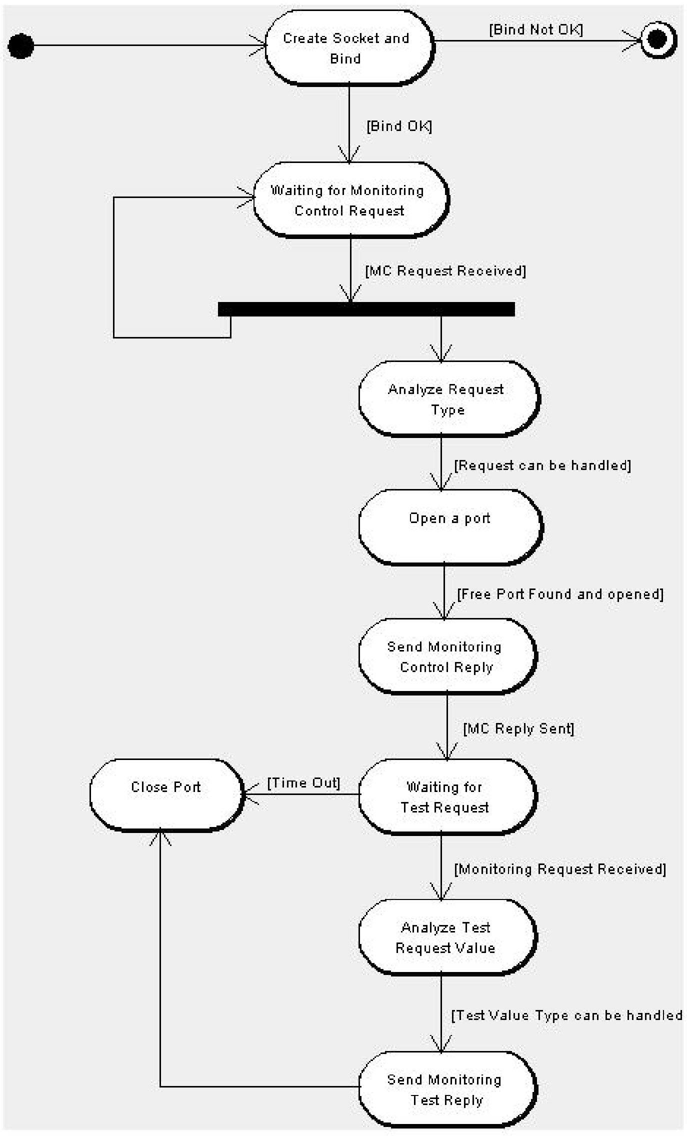

Create Socket and Bind
[Bind Not OK]
[Bind OK]
Waiting for Monitoring Control Request
[MC Request Received]
Analyze Request Type
[Request can be handled]
Open a port
[Free Port Found and opened]
Send Monitoring Control Reply
[MC Reply Sent]
Close Port
[Time Out]
Waiting for Test Request
[Monitoring Request Received]
Analyze Test Request Value
[Test Value Type can be handled
Send Monitoring Test Reply

Figure 44: Server General Behaviour.

Processes

A server is a multi-process application where there are two types of processes : Main process and Child process.

The main process task is to create a socket, bind it to the control port 5354 and listen for requests. For each monitoring control requests received, if the requested operation can be handled, the main process will check for a free port among a range of possible test ports, and create a new child process that will listen to that port. Once the port is opened, the main process role is to send back a monitoring control reply with the number of the newly opened port and continue the listening on the control port for next requests. The main process will not stop, unless it is closed using Ctrl + c.

A child process role is to receive a monitoring request (test phase), check if it is supported and answer it using the suited options. If an UDP-Echo request is received, a child process will also check for a Hop-by-Hop One-way Path Delay option in the IPv6 options in order collect and insert its records in the request echo. Once that the request echo has been sent, the child tells its father process to free the port, and exits.

Files

The server source code is composed of two main files : main.c and server.c.

(a) **main.c**
This file contains the program entry point, its role is to listen to the port 5354, and call the server.c functions for each packet received. If received packets are supported monitoring request, other server.c functions will be called in order to send a control reply and start the test phase.

(b) **server.c**
This file contains all necessary functions to analyze a packet received on the control port, check whether the requested monitoring operation is supported or not, and handle it. If the requested operation is supported, the file contains functions to open and listen to a new dedicated port and send a control reply specifying the new port.

3. The OWP Module for intermediary nodes.
Unfortunately, due to a lack of time, this module could not have been implemented.

4.1.4 Running the Application.

In order to run a measurement test between two IPv6 hosts running a Linux Kernel 2.6.14 or newer, the following procedure must be followed:

1. Copy the Client source files on the machine that will perform the measurement, and copy the Server source files on every machine that will be measured.
2. Compile both Client and server sources, with the command “make”.
3. Start every Server entities with the command “./responder”

4. Run the Client with the desired parameters using ./client Options
   Options are *[-h]-a ip6addr -p protocol [-m multicastop][-o][-i interv][-n nbtest]*:

-h	Show the Help.
-a	*ip6addr* IPv6 Address of the monitored host
-p	*protocol* :available modes are "udp" for UDP basic UDP-Echo , and "udpt" for UDP-Echo with timestamps option.
-m	*multicastop* : options are "nodes" for the All-node IPv6 multicast address or "routers" for the All-routers multicast addresses
-o	Combine the One-way Path Delay Option to the chosen mode, OWP is still in test mode because the module for intermediary nodes is missing.
-i	*interv* : specifies the interval in seconds between a test result and the next test. Minimum 5 seconds Maximum 3600 seconds. Default value is 60 seconds.
-n	*nbtest* : specifies the number of tests to perform before shutting down the client. Default value is 10 tests, Maximum value is 1000 tests.

   For example, the command :

   - *./client -a ::1 -p udp -i 10 -n 100*

     Will start an UDP-Echo operation with the local host every 10 seconds until 100 operations have been performed.
   - *./client -a ::1 -p udpt -m nodes*

     Will start an UDP-Echo with timestamps measurement (UETIMESTAMPS) on the multicast All-nodes address every 60 seconds until 10 tests have been performed. Note that the -a option is ignored if the -m option is valid.
   - *./client -a 2001:0db8:85a3:08d3:1319:8a2e:0370:7334 -p udpt -o*

     Will start an UDP-Echo with timestamps measurement (UETIMESTAMPS) and the One-way Path Delay option set in the Hop-by-hop Extension header on the given address, every 60 seconds until 10 tests have been performed.

It is important to note that the use of the OWP option (-o) requires from both Client and Server to be run as root. If it is not the case, the option cannot be set by the client in the Extension Header, and the Server will consider that it has not been used.

4.2 Test of the measurement tool.

This section will be devoted to the description of different tests measured with the Client - Server tool. Our interest is to show that the tool can perform the expected measurement functions on an IPv6 network. Because of a lack of time the measurement tool has only been tested in a Local Area Network, in order check its functionalities. A future work would be to test the tool over a wider IPv6 network. The tests have been realized between three machines running two different hosts : a Debian Sarge 3.1 distribution with a Linux Kernel 2.6.14, and a Knoppix 5.1 distribution with a Linux Kernel 2.6.19.
Due to the very short time intervals measured, the lack of accuracy in systems' clock, and the different target processing times, the results are rather inaccurate. One of the measured machines is a rather slow computer, which explains the differences in the computed metrics value.

1. UDP-Echo : Basic and Timestamps test :

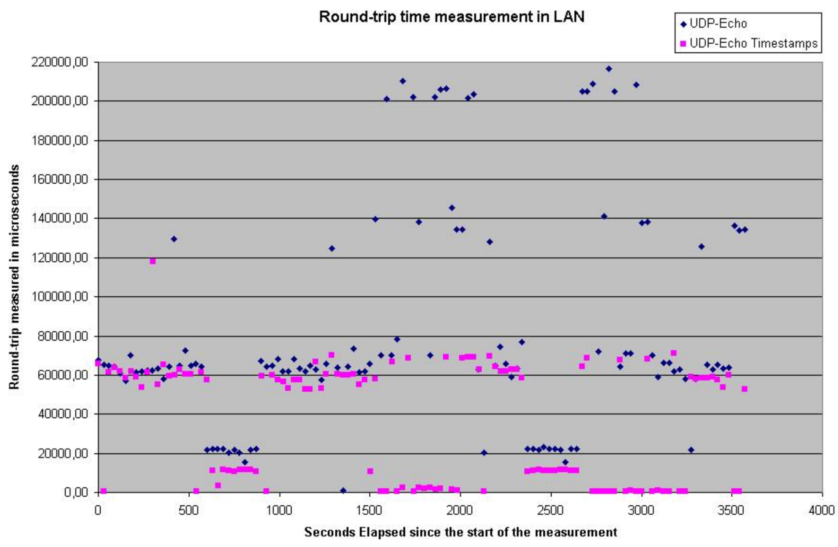


Figure 45: Test measurement of the Round-trip time use UDP-Echo and UDP-Echo Timestamps on a link-local address

The figure 45 shows the measurements of the Round-trip Delay estimation between two machines on a LAN. Two measurements have been performed toward the same target: the first one is using UDP-Echo in basic mode, and the second is using UDP-Echo in timestamps mode.
As we can see on the figure, UDP-Echo basic results are rather inaccurate. Some values are really high for a LAN measurement while others are normal. This is mainly because the chosen target is an old machine, with little memory. The processing time of the target can be very long,

depending on the order of scheduling.
UDP-Echo timestamps results constitute a better estimation of the Round-trip, removing a big part of the target processing time. However the measurement of the target processing time is performed at the application layer. It does not take into account time spent between the actual reception of the packet at the network layer by the system, and the reception of the binary data thanks to the receive function in Server's code. Also it does not taken into account the same phenomenon at the reply.
A more accurate estimation could be obtained by computing the effective Incoming and Outgoing timestamps, at the lowest level.

2. UDP-Echo via Multicast All-node Address:

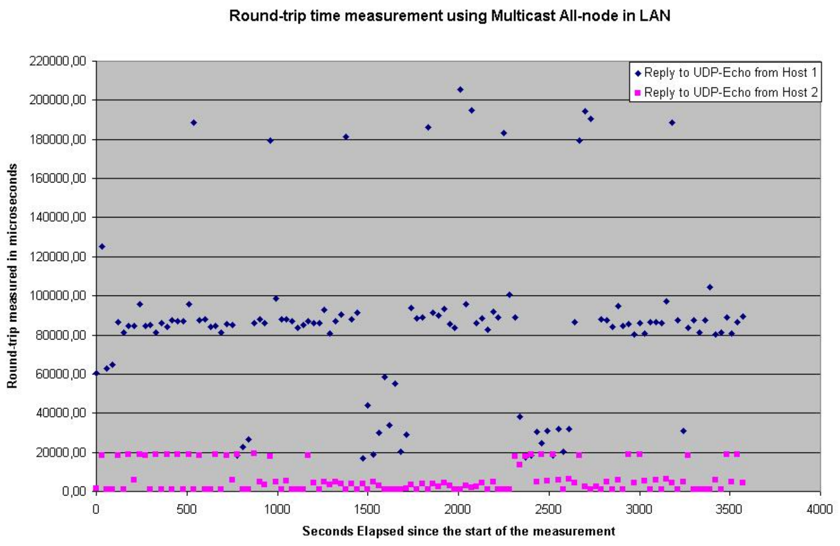


Figure 46: Test measurement of the Round-trip time using UDP-Echo on the Multicast All-node address

The figure 46 show the results obtained using UDP-Echo Multicast operation on a LAN. Two local nodes have answered to the multicast request, and participated in measurements. We can see a big difference between the two measured machines. The host number 1 is the same old machine used that in the previous test, which explains the high measured Round-trips. The host number 2 is a recent machine, and obtained results are normal for a LAN measurement.

3. UDP-Echo One-way Path Delay
The One-way Path Delay metric could not be measured, because the module required on intermediary nodes to participate in the OWP measurement has not been implemented. However the function has been tested in order to check that end systems could handle the expected behaviour.
During some UDP-Echo basic and UDP-Echo timestamps tests, the One-way Path Delay option records in the Hop-by-Hop header have been filled

with random information at the source, and compared to the OP field received in the destination reply. For each test performed, the OP field content was exactly the same than the random information inserted which prove that the mechanism is perfectly working at end-systems.

5 Conclusions

The work has shown that IPv6 will permit the design of measurement functions very similar to the existing IPv4 measurement methods. The main reason is that most of IPv4 methods are based on transport protocols, which were almost left unchanged in IPv6.
With an example, the design and implementation of an UDP-Echo function, we have seen how this natural way to build a measurement function for IPv6 could be achieved.

But more than that, IPv6 offers new possibilities to measurement methods, thanks to some new features. These features allow enhancing measurement methods capabilities, and this, in a smart and flexible way.

The Native Multicast mechanism is one of these new features. It will obviously facilitate multicast measurement methods, offering the possibility to flexibly create wide multicast groups over IPv6 networks.
IPv6 also specifies some pre-defined multicast group addresses that have to be joined by IPv6 nodes, which provides an automatic way for measuring different hosts.
In particular, the combination of UDP-Echo for IPv6 and the All-node multicast address has allowed the performing of a multicast UDP-Echo on a local network.

Another interesting facet of IPv6 is the way options are handled. Extension Headers offer new possibilities to enhance measurement methods. We have seen two different ways to exploit Extension Headers in a measurement purpose.

- Create a new Extension Header dedicated to measurements requires many standardisation efforts and need to convince IPv6 Actors of the necessity of its deployment. However this solution remains possible, but future works on such a header will have to focus on finding major advantages compared to other solutions.
- Use Type Length Value (TLV) options in existing Extension Headers in a measurement purpose.
  We have discussed two headers: The Hop-by-Hop Header and the Destination Header. Both have advantages and drawbacks, and we chose the Hop-by-Hop Option header to carry TLV measurements Options.

In our goal to propose a smarter measurement function, we wanted to find a way to enhance our active UDP-Echo function for IPv6. A good way was to combine it to Hop-by-Hop TLV options. This allowed extending an active end-to-end two-point measurement to an active end-to-end multi-point measurements.
We designed and partially implemented a combination of a One-way Path Delay (OWP) Option (in the Hop-by-Hop Header) with an UDP-Echo function for IPv6. UDP-Echo OWP is the result of this combination and is an example of a new smart way to exploit an IPv6 feature in the design of active measurements functions. Unfortunately, the complete test of the UDP-Echo OWP could not

be achieved because the OWP treatment procedure at intermediary nodes was not implemented. This could be part of a future work.

The implemented functions constitute good examples of some of the possibilities offered to performance monitoring on IPv6. They have been regrouped in an experimental tool. Future improvements of the tool could focus on increasing the number of metrics measured and options, or add other transport protocols support such as TCP or ICMP.

In our work we chose to combine our UDP-Echo prototype function to IPv6 Hop-by-Hop Options.
An interesting idea to investigate would be to combine a standardized active monitoring solution such as the One-way Active Measurement Protocol (OWAMP), which is also based on transport protocols, to the IPv6 Hop-by-Hop Options. This would create a powerful tool, capable of measuring every IPPM metric in multi-point mode.

References

[1] Deering, S., Hinden, R., Internet Protocol Version 6 Specification, IETF Network Working Group, RFC 2460, December 1998.

[2] Narten T., IBM, Nordmark E.n Sun Microsystems, Simpson W., Daydreamer, “Neighbor Discovery for IP Version 6 (IPv6)”, IETF Network Working Group, RFC 2461, December 1998

[3] Hagen, Silvia, “Ipv6 essentials” , 2nd edition, O’Reilly, May 2006

[4] Huitema Christian, “IPv6 The new internet Protocol” , Prentice Hall PTR

[5] Cisco Systems, “IOS IP Service level agreements “(User guide)

[6] Cizault, Gisèle, IPv6 - Théorie et pratique , 4ième édition , O’Reilly , November 2005

[7] Conta A., Transwitch, Deering S., Cisco Systems, Gupta M., Tropos Networks, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) : Specification”, IETF Network Working Group, RFC 4443, March 2006

[8] Postel J., Information Sciences Institute, “Internet Control Message Protocol (ICMP)”, IETF Network Working Group, RFC 792, September 1981

[9] Hinden R., Nokia, Deering S., Cisco Systems, “IP Version 6 Addressing Architecture”, IETF Network Working Group, RFC 4291, February 2006

[10] Rajahalme J., Nokia, Conta A., Transwitch, Carpenter B., IBM, Deering S., Cisco, “IPv6 Flow Label Specification”, IETF Network Working Group, RFC 3697, March 2004

[11] Paxson, V., “Towards a Framework for Defining Internet Performance Metrics”, Proceedings of INET’ 93, Montreal , June 1996.

[12] Kitamura, Hiroshi, Nec Corporation, “IPv6 New Feature: Hop-by-Hop Based Real-Time Efficient Connection/Link Status Investigation Mechanism”, Internet draft : draft-ietf-ipngwg-hbh-ext-csi-02.txt , Japan, October 1999.

[13] Pezaros, D.P, “Network Traffic Measurement for the Next Generation Internet”, Computing Department, Lancaster University, England , August 2005.

[14] Pezaros, D.P., Garcia, F.J., Gardner, R.D., Hitchison, D., Sventek, J.S., “In-line Service Measurements : Exploiting IPv6 Extension Headers”, Department of Computing Science, University of Glasgow, U.K.

[15] Young Jeff, “IPv6 Technology Overview”, Network and Telecom Strategies, Burton Group, Midvale, February 2007.

[16] Tonnes Brekne, Marius Clemetsen, Poul Heegaard, Tone Ingvaldsen, Brynjar Viken, “State of the Art in Performance Monitoring and Measurements”, Telenor R&D, 2002

[17] Shalunov S., TeitelBaum B;, Karp A., Boote J., Zekauskas M., Internet2, “ A one-way Active Measurement Protocol (OWAMP)”, IETF Network Working Group, RFC 4656, September 2006

[18] Almes G., Kalidindi S., Zekauskas M., Advanced Network & Services, “A One-way Delay Metric for IPPM”, IETF Network Working Group, RFC 2679, September 1999

[19] McGregor A., Luckie M., Waikato University, “IP Measurement Protocol (IPMP)”, IETF Network Working Group, Internet Draft, February 2004.

[20] McCann J., Digital Equipment Corporation, Deering S., Xerox PARC, Mogul J., “Path MTU Discovery for IP version 6”, IETF Network Working Group, RFC 1981, August 1996.

[21] Jian Zhang, Hongfei Chen, Huawei Technologies, “IPv6 measurement header”, IPv6 Working Group, Internet draft, April 2006

[22] Thomson S., Bellcore, Narten T., IBM, “IPv6 Stateless Address Autoconfiguration” , IETF Network Working Group, RFC 2462, December 1998

[23] Stevens W., Consultant, Thomas M., Altavista, “Advanced Sockets API for IPv6”, IETF Network Working Group, RFC 2292, February 1998

[24] Stevens W., Thomas M., Consultant, Nordmark E., Sun, Jinmei T., Toshiba, “Advanced Sockets Application Program Interface (API) for IPv6”, IETF Network Working Group, RFC 3542, May 2003

[25] Droms R., Ed., Cisco, Bound J., Hewlett Packard, Volz B., Ericsson, Lemon T., Nominum, Perkins C., Nokia Research Center, Carney M., Sun Microsystems, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, IETF Network Working Group, RFC 3315, July 2003.

[26] Paxson V., Lawrence Berkeley National Lab, Almes G., Advanced Network & Services, Mahdavi J., Mathis M., Pittsburgh Supercomputer Center, “Framework for IP Performance Metrics”, IETF Network Working Group, RFC 2330, May 1998

[27] http://www.ietf.org/html.charters/ippm-charter.html

[28] Almes G., Kalidindi S., Zekauskas M., Advanced Network & Services, “A Round-trip Delay Metric for IPPM”, IETF Network Working Group, RFC 2681, September 1999

[29] Almes G., Kalidindi S., Zekauskas M., Advanced Network & Services, “A One-way Delay Metric for IPPM”, IETF Network Working Group, RFC 2679, September 1999

[30] Cisco Systems , “Cisco IOS Netflow Overview”, February 2006, available at: www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/honf_c/onf_ov.pdf

[31] Pasztor A., Veitch D., “PC Based Precision Timing Without GPS”, in Proceeding of ACM SIGMETRICS, 2002

[32] Cisco Systems, "Cisco IOS IP Service Level Agreements White paper", 2005, available at : www.cisco.com/en/US/tech/tk648/tk362/technologies_white_paper0900aecd8017f8c9.shtml

[33] Jae-Hoon J., Jung-Soo P., Seung-Yun L., Yong-Jin K., ETRI, "One-way Delay Measurement using IPv6 Source Routing", Internet draft : draft-jeong-1way-delay-ipv6-source-rounting-00.txt, February 2002

[34] Collignon N., "IPv6 : Les dangers du Routing Header Type 0", Hervé Schauer Consultants, February 2007, available at : http://www.hsc.fr/ressources/breves/ipv6-rt0.html.fr.

[35] Borman D., Berkeley Software Design, Deering S., Cisco, Hinden R., Nokia, "IPv6 Jumbogram", IETF Network Working Group, RFC 2675, August 1999

[36] Postel J., Information Sciences Institute, "User Datagram Protocol (UDP)", RFC 768, August 1980

[37] Information Sciences Institute, University of Southern California, "Transmission Control Protocol", RFC 793, September 1981

[38] Gilligan R., Intransa, Inc., Thomson S., Cisco, Bound J., McCann J., Hewlett-Packard, Stevens. W, "Basic Socket Interface Extensions for IPv6", IETF Network Working Group, RFC 3493, February 2003.