IPPM Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2007

S. Niccolini
S. Tartarelli
J. Quittek
NEC
M. Swany
UDel

February 22, 2007

Traceroute Measurements Information Model and XML Data Model
draft-ietf-ippm-storetraceroutes-03

Status of this Memo

Copyright Notice

Abstract

This memo describes a standard way to store traceroute measurements. To better address the traceroute measurements storing issue, the authors first of all give a definition of the traceroute tool, describe the tool itself as well as its parameters and the default values on the most common operating systems and the output results that can be stored. Afterwards, the common information model with the base elements of the traceroute measurement storing is defined dividing the information elements in two semantically separated groups (configuration elements and results ones). Moreover an additional element is defined to relate configuration elements and results ones by means of a common unique identifier. On the basis of the information model a data model is then proposed in order to actually store the traceroute measurements.

Table of Contents

1. Introduction

Traceroutes are being used by lots of measurement efforts, either as an independent measurement or to get path information to support other measurement efforts. That is why there is the need to standardize the way traceroute measurements are stored and the related metrics associated with such measurements. Since traceroute is a tool that has built-in configurable mechanisms like time-outs and can experience problems related to the crossing of firewalls thus experiencing fake losses or incomplete delay information. The standard metrics defined by IPPM working group in matter of delay, connectivity and losses do not apply to the metrics returned by the traceroute tool; therefore, in order to compare results of traceroute measurements, the solution is to add to the stored results a specification of the operating system and version for the tool used. Moreover there is a need to better define the traceroute tool as well as its parameters and the results it outputs since, to the authors' knowledge, there is so far no standard describing these. These are the motivations that moved the authors to write this draft in the context of the IPPM working group even if other working groups (like DISMAN) have already addressed similar issues related to the definition of the MIB for configuring and retrieving traceroute measurements results. This draft, in order to store traceroute results and allow comparison of them, defines a standard way to store traceroute measurements using a XML schema. The draft is organised as follows: Section 2 gives the definition of traceroute used as reference in the rest of the draft as well as the traceroute configurable parameters and their default values for the most common operating systems. Section 3 reports on both traceroute accuracy of results and existing alternatives for traceroute implementations. Section 4 describes the results available from the traceroute output screen. Section 5 and Section 6 respectively describe our proposed Information model and Data model for storing traceroute measurements. Section 8 reports the differences to [RFC4560]. The draft ends with security considerations and IANA considerations in Section 9 and Section 10 respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2. Definition of Traceroute

Traceroute is a network diagnostic tool used to determine the hop by hop path from a source to a destination and the Round Trip Time (RTT) from the source to each hop. Traceroute can therefore be used to discover where and how a host is connected to the Internet and can be

usefully employed to troubleshoot network connections.

Typically, traceroute attemps to discover the path to a destination by sending UDP probes with specific time-to-live (TTL) values in the IP packet header and trying to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host.

More in detail, the host running the traceroute sends the first set of probes with TTL equal to one (some implementations allow setting the initial TTL to a value equal to "n" different from one, so that the first "n-1" hops are skipped and the first hop that will be traced is the "n-th" in the path). Upon receiving a probe, the first hop host decreases the TTL value by one. By observing a TTL value equal to zero, the host rejects the probe and typically returns an ICMP message with a TIME_EXCEEDED value. Traceroute can therefore derive the IP address of the first hop from the header of the ICMP message and evaluate the RTT between the source and the first hop. The next hops are discovered following the same procedure, taking care of increasing at each step the TTL value of the outgoing probes by one. The TTL value is increased until either an ICMP PORT_UNREACHABLE message is received, meaning that the destination has been reached, or the maximum configured number of hops has been hit.

Some implementations, use ICMP Echos, instead of UDP datagrams. However, many routers do not return ICMP messages about ICMP messages, i.e. no ICMP TIME_EXCEEDED is returned for an ICMP Echo. Therefore, in this draft we RECOMMEND to base implementations on UDP datagrams.

2.1. Traceroute Configuration Parameters

In order to define an information model for storing traceroutes, we first investigated which configuration parameters are relevant when running traceroute. We considered four major traceroute implementations and compared them based on configurable parameters and default values. The LINUX (SUSE 9.1), BSD (FreeBSD 7.0) and UNIX (SunOS 5.9) implemetations are based on UDP datagrams, while the WINDOWS (XP SP2) one uses ICMP Echos. The comparison is summarized in the following table, where an N/A in the option column, means that such parameter is not configurable for the specific implementation. A comprehensive comparison of available implementations is outside the scope of this draft; however, already by sampling a few different implementations, we can observe that they can differ quite significantly in terms of configurable parameters and also default values. Note that in the following table we reported only those options which are available in at least two of the considered implementations.

OS	Option	Description	Default
LINUX	-m	Specify the maximum TTL used in outgoing probes.	30
FreeBSD	-m		OS var
UNIX	-m		30
WINDOWS	-h		30
LINUX	-n	Display hop addresses numerically rather than simbolically.	-
FreeBSD	-n		-
UNIX	-n		-
WINDOWS	-d		-
LINUX	-w	Set the time to wait for a response to a probe.	3 sec
FreeBSD	-w		5 sec

UNIX	-w		5 sec
WINDOWS	-w		4 sec
LINUX	N/A	Specify a loose source route gateway (to direct the traceroute through routers not necessarily in the path).	-
FreeBSD	-g		-
UNIX	-g		-
WINDOWS	-g		-
LINUX	-p	Set the base UDP port number used in probes (UDP port = base + nhops - 1).	33434
FreeBSD	-p		33434
UNIX	-p		33434
WINDOWS	N/A		-
LINUX	-q	Set the number of probes per TTL.	3
FreeBSD	-q		3
UNIX	-q		3
WINDOWS	N/A		3
LINUX	-S	Set the IP source address in outgoing probes to the specified value.	IP address of the out interface
FreeBSD	-s		
UNIX	-s		
WINDOWS	N/A		
LINUX	-t	Set the type-of-service (TOS) in the probes to the specified value.	0
FreeBSD	-t		0

UNIX	-t		0
WINDOWS	N/A		0
LINUX	-v	Verbose output: received ICMP packets other than TIME_EXCEEDED and UNREACHABLE are listed.	-
FreeBSD	-v		-
UNIX	-v		-
WINDOWS	N/A		-
LINUX	N/A	Set the time (in msec) to pause between probes.	-
FreeBSD	-z		0
UNIX	-P		0
WINDOWS	N/A		-
LINUX	-r	Bypass the normal routing tables and send directly to a host on attached network.	-
FreeBSD	-r		-
UNIX	-r		-
WINDOWS	N/A		-
LINUX	-f	Set the initial TTL for the first probe.	1
FreeBSD	-f		1
UNIX	-f		1
WINDOWS	N/A		1
LINUX	-F	Set the "don't fragment" bit.	-
FreeBSD	-F		-
UNIX	-F		-

WINDOWS	N/A		-
LINUX	N/A	Enables socket level debugging.	-
FreeBSD	-d		-
UNIX	-d		-
WINDOWS	N/A		-
LINUX	N/A	Use ICMP ECHO instead of UDP datagrams.	-
FreeBSD	-I		-
UNIX	-I		-
WINDOWS	N/A		-
LINUX	-I	Specify a network interface to obtain the IP address for outgoing IP packets (alternative to option -s).	-
FreeBSD	-i		-
UNIX	-i		-
WINDOWS	N/A		-
LINUX	N/A	Toggle checksum.	-
FreeBSD	-x		-
UNIX	-x		-
WINDOWS	N/A		-
LINUX	-	As optional last paramater, LINUX, FreeBSD and UNIX implementations allow specifying the probe datagram length for outgoing probes.	Depends on implementation.
FreeBSD	-		
UNIX	-		

WINDOWS	N/A		

3. Known Problems with Traceroute

3.1. Accuracy of Results

A known inconsistency exists between the round-trip delay metric defined by the IPPM working group and the results returned by the current traceroute implementations. Unfortunately, it is unlikely that the traceroute implementations will implement the standard definition in the near future. In order to compare results of different traceroute measurements, specifications both of the operating system (name and version) and of the traceroute tool version used are added to the metadata elements in order to help in comparing metrics. Moreover, the traceroute has built-in configurable mechanisms like time-outs and can experience problems related to the crossing of firewalls; therefore some of the packets that traceroute sends out end up being time-out or filtered. As a consequence, it might not be possible to trace the path to a node or there might not be a complete set of probes describing the RTT to reach it.

3.2. Alternative traceroute Implementations

As stated above, the widespred use of firewalls might prevent UDP or ICMP based traceroutes to completely trace the path to a destination, since traceroute probes might end up being filtered. In some cases, such limitation might be overcome by sending instead TCP packets to specific ports that hosts located behind the firewall are listening for connections on. TCP based implementations use TCP SYN or FYN probes and listen for TIME_EXCEEDED messages, TCP RESET and other messages from firewalls and gateways on the path. On the other hand, some firewalls filter out TCP SYN packets to prevent denial of service attacks, therefore the actual advantage of using TCP instead of UDP traceroute depends mainly on firewall configurations, which are not known in adavance. A detailed analysis of TCP based traceroutes is outside the scope of this draft, therefore in the sequel, we will restrict our focus to the most commonly implemented UDP based traceroute.

4. Reports/results

The following list reports the information fields provided by all traceroute implementations considered. The order in which they are reported here is not relevant and it changes in different implementations. For each hop the information reported is:

- o the hop index;
- o the host symbolical address, provided that at least one of the probes received a response, the symbolic address could be resolved at the correponding host and that the option to display only numerical addresses was not set;
- o the host IP address, provided that at least one of the probes received a response;
- o the RTT for each response to a probe.

Depending on the traceroute implementation, additional information might be displayed in the output (for instance MPLS-related information).

It might happen that some probes do not receive a response within the configured time-out (for instance if the probe is filtered out by a firewall). In this case, an "*" is displayed in place of the RTT. Besides, for delays below 1 ms, some implementations reports 0 ms (f.i. UNIX and LINUX) while WINDOWS tracert reports "< 1 ms".

5. Information Model for Storing Traceroute Measurements

This section describes the information model for the traceroute measurements data storing. The information model is composed of information elements; for defining these information elements, a template is used. Such template is specified in the list below:

- o name - A unique and meaningful name for the information element. The preferred spelling for the name is to use mixed case if the name is compound, with an initial lower case letter, e.g., "sourceIpAddress".
- o description - The semantics of this information element.
- o dataType - One of the types listed in Section 5.1 of this document or in an extension of the information model. The type space for attributes is constrained to facilitate implementation.
- o units - If the element is a measure of some kind, the units identify what the measure is.
- o default value - The default value for the element (where applicable).

5.1. Data Types

This section describes the set of valid data types of the information model.

- String - The type "String" represents a finite length string of valid characters from the Unicode character encoding set. Unicode allows for ASCII and many other international character sets to be used. It is expected that strings will be encoded in UTF-8 format, which is identical in encoding for USASCII characters, but also accomodates other Unicode multibyte characters.
- InetAddressType - The type "InetAddressType" represents a type of Internet address. The allowed values are to be intended as imported from [RFC4001]; an additional allowed value is "asnumber".
- InetAddress - The type "InetAddress" denotes a generic Internet address. The allowed values are to be intended as imported from [RFC4001]; an additional allowed value is the AS number to be indicated as the actual number plus the indication how the mapping from IP address to AS number was performed.
- TruthValue - The type "TruthValue" represents a boolean value. The allowed values are to be intended as imported from [RFC2579].
- Unsigned32 - The type "Unsigned32" represents a value in the range (0..4294967295).
- InterfaceIndexOrZero - The type "InterfaceIndexOrZero" is an extension of the InterfaceIndex convention. The latter defines a greater than zero value used to identify an interface or interface sub-layer in the system. This extension permits the additional value of zero. Examples of the usage of zero might include situations where interface was unknown, or when none or all interfaces need to be referenced. The allowed values are to be intended as imported from [RFC2863].
- ProbesType - The type "ProbesType" represents a way of indicating the protocol used for the traceroute probes. Allowed values are UDP, TCP, ICMP.
- DateAndTime - The type "DateAndTime" represents a date-time specification. The allowed values are to be intended as imported from [RFC2579] apart from the fact that in this document there is the need to use a milli-second resolution instead a deci-second one.
- OperationResponseStatus - The type "OperationResponseStatus" is used to report the result of an operation. The allowed values are to be intended as imported from [RFC4560].

5.2. Information Elements

This section describes the elements of the traceroute measurement. The elements are grouped in two groups (Configuration and Results) according to their semantics. In order to relate configuration and results elements by means of a common unique identifier, an additional element is defined belonging to both the two groups.

5.2.1. Configuration Information Elements

This section describes the elements of the traceroute measurement that are specific to traceroute configuration.

5.2.1.1. CtlTargetAddressType

o name - CtlTargetAddressType
o description - Specifies the type of host address used in the traceroute command.
o dataType - InetAddressType
o units - N/A
o default value - N/A

5.2.1.2. CtlTargetAddress

o name - CtlTargetAddress
o description - Specifies the host address used in the traceroute command. The host address type can be determined by the examining the value of the corresponding CtlTargetAddressType.
o dataType - InetAddress
o units - N/A
o default value - N/A

5.2.1.3. CtlByPassRouteTable

o name - CtlByPassRouteTable
o description - Specifies if the optional bypassing of the route table was enabled or not. If enabled, the traceroute will bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to perform

the traceroute operation to a local host through an interface that has no route defined.
- o dataType - TruthValue
- o units - N/A
- o default value - false

5.2.1.4. CtlProbeDataSize

- o name - CtlProbeDataSize
- o description - Specifies the size of the data portion of a traceroute operation in octets. If the RECOMMENDED traceroute method (UDP datagrams as probes) is used, then the value contained in this object is exact. If another traceroute method is used for which the specified size is not appropriate, then the implementation should have used whatever size (appropriate to the method) is closest to the specified size. The maximum value for

this object was computed by substracting the smallest possible IP header size of 20 octets (IPv4 header with no options) and the UDP header size of 8 octets from the maximum IP packet size. An IP packet has a maximum size of 65535 octets (excluding IPv6 Jumbograms).
- o dataType - Unsigned32
- o units - octects
- o default value - 0

5.2.1.5. CtlTimeOut

- o name - CtlTimeOut
- o description - Specifies the time-out value, in seconds, for the traceroute operation.
- o dataType - Unsigned32
- o units - seconds
- o default value - 3

5.2.1.6. CtlProbesPerHop

- o name - CtlProbesPerHop
- o description - Specifies the number of times to reissue a traceroute request with the same time-to-live (TTL) value.
- o dataType - Unsigned32
- o units - probes

- default value - 3

5.2.1.7. CtlPort

- name - CtlPort
- description - Specifies the base UDP port used by the traceroute operation. Need to specify a port that is not in use at the destination (target) host. The default value for this object is the IANA assigned port, 33434, for the traceroute function.
- dataType - Unsigned32
- units - UDP Port
- default value - 33434

5.2.1.8. CtlMaxTtl

- name - CtlMaxTtl
- description - Specifies the maximum TTL value for the traceroute operation.
- dataType - Unsigned32
- units - time-to-live value
- default value - 30

5.2.1.9. CtlDSField

- name - CtlDSField
- description - Specifies the value that was stored in the Differentiated Services (DS) field in the IP packet used to encapsulate the traceroute probe. The DS Field is defined as the Type of Service (TOS) octet in a IPv4 header or as the Traffic Class octet in a IPv6 header. The value of this object must be a decimal integer in the range from 0 to 255. This option can be used to determine what effect an explicit DS field setting has on a traceroute response. Not all values are legal or meaningful. Useful TOS octet values are probably '16' (low delay) and '8' (high throughput). Further references can be found in [RFC2474] for the definition of the Differentiated Services (DS) field and to [RFC1812] Section 5.3.2 for Type of Service (TOS).
- dataType - Unsigned32
- units - N/A
- default value - 0

5.2.1.10. CtlSourceAddressType

- name - CtlSourceAddressType
- description - Specifies the type of the source address, CtlSourceAddress, used when performing the traceroute operation.
- dataType - InetAddressType
- units - N/A
- default value - N/A

5.2.1.11. CtlSourceAddress

- name - CtlSourceAddress
- description - Specifies the IP address (which has to be given as an IP number, not a hostname) as the source address used in outgoing probe packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the primary IP address of the interface the probe packet is sent on. A zero length octet string value for this object means that source address specification was disabled. The address type (InetAddressType) that relates to this object is specified by the corresponding value of CtlSourceAddressType.
- dataType - InetAddress
- units - N/A
- default value - N/A

5.2.1.12. CtlIfIndex

- name - CtlIfIndex
- description - Specifies the inferface index used in the traceroute operation for sending the traceroute probes. A value of zero for this object implies that the interface was unknown.
- dataType - InterfaceIndexOrZero
- units - N/A
- default value - 0

5.2.1.13. CtlMiscOptions

- name - CtlMiscOptions

- description - Specifies implementation dependent options.
- dataType - String
- units - N/A
- default value - N/A

5.2.1.14. CtlMaxFailures

- name - CtlMaxFailures
- description - Specifies the maximum number of consecutive timeouts allowed before terminating a traceroute operation. A value of either 255 (maximum hop count/possible TTL value) or a 0 indicates that the function of terminating a remote traceroute operation when a specific number of consecutive timeouts are detected was disabled. This element is included to give full compatibility with [RFC4560]. No known implementation of traceroute currently supports it.
- dataType - Unsigned32
- units - timeouts
- default value - 5

5.2.1.15. CtlDontFragment

- name - CtlDontFragment
- description - Specifies if the don't fragment flag (DF) in the IP header for a probe was enabled or not. Setting the DF flag can be used for performing a manual PATH MTU test.
- dataType - TruthValue
- units - N/A
- default value - false

5.2.1.16. CtlInitialTtl

- name - CtlInitialTtl
- description - Specifies the initial TTL value used in a traceroute operation. Such TTL setting is intended to bypass the initial (often well known) portion of a path.

- dataType - Unsigned32
- units - N/A
- default value - 1

5.2.1.17. CtlDescr

- o name - CtlDescr
- o description - The purpose of this element is to provide a description of the traceroute test.
- o dataType - String
- o units - N/A
- o default value - N/A

5.2.1.18. CtlType

- o name - CtlType
- o description - Specifies the implementation method used for the traceroute operation. It specifies if the traceroute is using TCP, UDP or ICMP probes.
- o dataType - ProbesType
- o units - N/A
- o default value - UDP

5.2.2. Results Information Elements

This section describes the elements of the traceroute measurement that are specific to the results of a traceroute operation.

5.2.2.1. ResultsStartDateAndTime

- o name - ResultsStartDateAndTime
- o description - Specifies the date and start time of the traceroute operation. This is the time when the first probe was seen at the sending interface.
- o dataType - DateAndTime
- o units - N/A
- o default value - N/A

5.2.2.2. ResultsIpTgtAddrType

- o name - ResultsIpTgtAddrType
- o description - Specifies the type of address stored in the corresponding ResultsIpTgtAddr element.
- o dataType - InetAddressType
- o units - N/A
- o default value - N/A

5.2.2.3. ResultsIpTgtAddr

- name - ResultsIpTgtAddr
- description - Specifies the IP address associated with a CtlTargetAddress value when the destination address is specified as a DNS name. The value of this object should be a zero length octet string when a DNS name is not specified or when a specified DNS name fails to resolve.
- dataType - InetAddress
- units - N/A
- default value - N/A

5.2.2.4. Index

- name - Index
- description - Specifies an index that consecutively numbers all probes for which a reply was received in the sequential order in which the replies were received. The maximum value for this object is CtlMaxTtl*CtlProbesPerHop.
- dataType - Unsigned32
- units - N/A
- default value - N/A

5.2.2.5. HopIndex

- name - HopIndex
- description - Specifies which hop in a traceroute path that the probe's results are for.
- dataType - Unsigned32
- units - N/A
- default value - N/A

5.2.2.6. IndexPerHop

- name - IndexPerHop
- description - Specifies the index of a probe for a particular hop in a traceroute path. The number of probes per hop is determined by the value of the corresponding CtlProbesPerHop element.
- dataType - Unsigned32
- units - N/A
- default value - N/A

5.2.2.7. HopAddrType

- name - HopAddrType
- description - Specifies the type of address stored in the corresponding HopAddr element.

- o dataType - InetAddressType
- o units - N/A
- o default value - N/A

5.2.2.8. HopAddr

- o name - HopAddr
- o description - Specifies the address of a hop in the traceroute path. This object is not allowed to be a DNS name. The value of the corresponding object, HopAddrType, indicates this object's IP address type.
- o dataType - InetAddress
- o units - N/A
- o default value - N/A

5.2.2.9. HopGeoLocation

- o name - HopGeoLocation
- o description - Specifies the geo location of a hop in the traceroute path.
- o dataType - String
- o units - N/A
- o default value - N/A

5.2.2.10. MPLSTopLabel

- o name - MPLSTopLabel
- o description - Specifies the top entry of the MPLS label stack of a probe observed when the probe arrived at the hop that replied to the probe. This object contains the top MPLS label stack entry as 32 bit value as it is observed on the MPLS label stack. Contained in this single number are the MPLS label, the Exp field, the S flag, and the MPLS TTL value as specified in RFC 3032 [RFC3032].
- o dataType - Unsigned32
- o units - N/A
- o default value - N/A

5.2.2.11. RoundTripTime

- o name - RoundTripTime

- description - Specifies the amount of time measured in milliseconds from when a probe was sent to when its response was received or when it timed out. The value of this element is reported as the truncation of the number reported by the traceroute tool (the output "< 1 ms" is therefore encoded as 0 ms). A string with the value of "RoundTripTimeNotAvailable" means either the probe was lost because of a timeout or it was not possible to transmit a probe.

- dataType - Unsigned32 or String
- units - milliseconds or N/A
- default value - N/A

5.2.2.12. ResponseStatus

- name - ResponseStatus
- description - Specifies the result of a traceroute operation made by the host for a particular probe.
- dataType - OperationResponseStatus
- units - N/A
- default value - N/A

5.2.2.13. Time

- name - Time
- description - Specifies the timestamp for when the response to the probe was received interface.
- dataType - DateAndTime
- units - N/A
- default value - N/A

5.2.2.14. ResultsHopRawOutputData

- name - ResultsHopRawOutputData
- description - Specifies the raw output data returned by the traceroute operation for a certain hop in a traceroute path.
- dataType - String
- units - N/A
- default value - N/A

5.2.2.15. ResultsEndDateAndTime

- name - ResultsEndDateAndTime
- description - Specifies the date and end time of the traceroute operation. It is either the time when the response to the last probe of the traceroute operation was received or the time when the last probe of the traceroute operation was sent plus the relative timeout (in case of missing response).
- dataType - DateAndTime
- units - N/A
- default value - N/A

5.2.3. Information Element Correlating Configuration and Results Elements

This section defines an additional element belonging to both the two previous groups. This element is defined in order to relate

configuration elements and results ones by means of a common unique identifier.

5.2.3.1. TestName

- name - TestName
- description - Specifies the name of a traceroute test. This is locally unique.
- dataType - String
- units - N/A
- default value - N/A

6. Data Model for Storing Traceroute Measurements

For storing and transmitting information according to the information model defined in the previous section, a data model is required that specifies how to encode the elements of the information model.

There are several design choices for a data model. It can use a binary or textual representation and it can be defined from scratch or use already existing frameworks and data models. In general, the use of already existing frameworks and models should be preferred.

Binary and textual representation both have advantages and disadvantages. Textual representions are (with some limitations) human readable while a binary representation consumes less resources for storing, transmitting and parsing data.

An already existing and closely related data model is the DISMAN-TRACEROUTE-MIB module [RFC4560], that specifies a BER encoding [RFC3417] used by the Simple Network Management Protocol (SNMP) [RFC3410] for transmitting traceroute information. This data model is well suited and supported within network management systems, but as a general format for storing and transmitting traceroute results it is not easily applicable.

Another binary representation would be an extension of traffic flow information encodings as specified for the IPFIX protocol [I-D.ietf-ipfix-protocol], [I-D.ietf-ipfix-info]. The IPFIX protocol is extensible. However, the architecture behind this protocol [I-D.ietf-ipfix-architecture] is targeted at exporting passively measured flow information. Therefore, some obstacles are expected when trying to use it for transmitting traceroute measurement results.

For textual representations, using the eXtensible Markup Language (XML) [XML] is an obvious choice. XML supports clean structuring of

data and syntax checking of records. With some limitations it is human readable. It is supported well by a huge pool of tools and standards for generating, transmitting, parsing and converting it to other data formats. Its disadvantages is the resource comsumption for processing, storing, and transmitting information. Since the expected data volumes of traceroute data in network operation and maintenance is not expected to be extremly high, the inefficient usage of resources is not a significant disadvantage. Therefore, XML was chosen as basis for the traceroute information model that is specified in this section.

Section 7 contains the XML schema to be used as a template for storing and/or exchanging traceroute measurements. The schema was designed in order to use an extensible approach based on templates (pretty similar to how IPFIX protocol is designed) where the traceroute configuration elements (both the requested parameters, Request, and the actual parameters used, MeasurementMetadata) are metadata to be referenced by results information elements (data) by means of the TestName element (used as unique identifier). Currently

Global Grid Forum (GGF) is also using this approach and cross-requirements have been analyzed. As a result of this analysis the XML schema contained in Section 7 is compatible with GGF schema since it was designed in a way that both limits the unecessary redundancy and a simple one-to-one trasformation between the two exist.

7. XML Schema for traceroute Measurements

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:ietf:params:xml:ns:traceroute-1.0">

  <xs:simpleType name="inetAddressType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unknown"/>
      <xs:enumeration value="ipv4"/>
      <xs:enumeration value="ipv6"/>
      <xs:enumeration value="dns"/>
      <xs:enumeration value="asnumber"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="inetAddressTypeWithoutDns">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unknown"/>
      <xs:enumeration value="ipv4"/>
      <xs:enumeration value="ipv6"/>
      <xs:enumeration value="asnumber"/>
```

```
      <xs:enumeration value="noSpecification"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="_zeroLengthString">
    <xs:restriction base="xs:string">
      <xs:maxLength value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="_inetAddressIpv4">
    <xs:restriction base="xs:string">
```

```
         <xs:pattern value="(([1-9]?[0-9]|1[0-9][0-9]|
         2[0-4][0-9]|25[0-5]).){3}([1-9]?[0-9]|1[0-9]
         [0-9]|2[0-4][0-9]|25[0-5])"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_inetAddressIpv6">
       <xs:restriction base="xs:string">
         <xs:pattern value="(([\dA-Fa-f]{1,4}:){7}[\dA-Fa-f]{1,4})
         (:([\d]{1,3}.){3}[\d]{1,3})?"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_inetAddressDns">
       <xs:restriction base="xs:string">
         <xs:maxLength value="256"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_asNumber">
       <xs:restriction base="xs:unsignedInt"/>
     </xs:simpleType>

     <xs:simpleType name="_ipASNumberMappingType">
       <xs:restriction base="xs:string">
         <xs:enumeration value="bgptables"/>
         <xs:enumeration value="routingregistries"/>
         <xs:enumeration value="nslookup"/>
         <xs:enumeration value="others"/>
         <xs:enumeration value="unknown"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="operationResponseStatus">
       <xs:restriction base="xs:string">
         <xs:enumeration value="responseReceived"/>
```

```
         <xs:enumeration value="unknown"/>
         <xs:enumeration value="internalError"/>
         <xs:enumeration value="requestTimedOut"/>
         <xs:enumeration value="unknownDestinationAddress"/>
         <xs:enumeration value="noRouteToTarget"/>
         <xs:enumeration value="interfaceInactiveToTarget"/>
         <xs:enumeration value="arpFailure"/>
         <xs:enumeration value="maxConcurrentLimitReached"/>
         <xs:enumeration value="unableToResolveDnsName"/>
         <xs:enumeration value="invalidHostAddress"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_dateAndTimeUpToSeconds">
       <xs:restriction base="xs:dateTime"/>
     </xs:simpleType>

     <xs:simpleType name="_timeMilliseconds">
       <xs:restriction base="xs:unsignedShort">
         <xs:maxExclusive value="1000"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_TestName">
       <xs:annotation>
         <xs:documentation>Specifies the name of a traceroute
         test. This is locally unique.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:string">
         <xs:maxLength value="32"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_OSName">
       <xs:annotation>
         <xs:documentation>Specifies the name of the operating
         system on which the traceroute was launched.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:string">
         <xs:maxLength value="32"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_OSVersion">
       <xs:annotation>
         <xs:documentation>Specifies the OS version on which the
```

```
    traceroute was launched.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="32"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_ToolVersion">
  <xs:annotation>
    <xs:documentation>Specifies the version of the traceroute
    tool used.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="32"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_CtlByPassRouteTable">
  <xs:annotation>
    <xs:documentation>Specifies if the optional bypassing
    of the route table was enabled or not.  If enabled,
    the traceroute will bypass the normal routing tables
    and send directly to a host on an attached network.
    If the host is not on a directly-attached network,
    an error is returned. This option can be used to
    perform the traceroute operation to a local host
    through an interface that has no route defined.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>

<xs:simpleType name="_CtlProbeDataSize">
  <xs:annotation>
    <xs:documentation>Specifies the size of the data
    portion of a traceroute operation in octets. If the
    RECOMMENDED traceroute method (UDP datagrams as probes)
    is used, then the value contained in this object is
    exact. If another traceroute method is used for which
    the specified size is not appropriate, then the
    implementation should have used whatever size (appropriate
    to the method) is closest to the specified size.
    The maximum value for this object was computed by
    substracting the smallest possible IP header size of 20
    octets (IPv4 header with no options) and the UDP header
    size of 8 octets from the maximum IP packet size. An IP
```

```
         packet has a maximum size of 65535 octets (excluding IPv6
         jumbograms). Units are: octects.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedShort">
         <xs:maxExclusive value="65508"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_CtlTimeOut">
       <xs:annotation>
         <xs:documentation>Specifies the time-out value, in
         seconds, for the traceroute operation.
         Units are: seconds.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte">
         <xs:minExclusive value="0"/>
         <xs:maxExclusive value="61"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_CtlProbesPerHop">
       <xs:annotation>
         <xs:documentation>Specifies the number of times to
         reissue a traceroute request with the same time-to-live
         (TTL) value. Units are: probes.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte">
         <xs:minExclusive value="0"/>
         <xs:maxExclusive value="11"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_CtlPort">
       <xs:annotation>
         <xs:documentation>Specifies the base UDP port used by
         the traceroute operation.  Need to specify a port that
         is not in use at the destination (target) host.
         The default value for this object is the IANA assigned
         port, 33434, for the traceroute function.
         Units are: UDP port.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedShort">
         <xs:minExclusive value="0"/>
       </xs:restriction>
```

```
     </xs:simpleType>

     <xs:simpleType name="_CtlMaxTtl">
       <xs:annotation>
         <xs:documentation>Specifies the maximum TTL value for
         the traceroute operation.
         Units are: time-to-live value.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte">
         <xs:minExclusive value="0"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_CtlDSField">
       <xs:annotation>
         <xs:documentation>Specifies the value that was stored in
         the Differentiated Services (DS) field in the IP packet
         used to encapsulate the traceroute probe. The DS Field
         is defined as the Type of Service (TOS) octet in a IPv4
         header or as the Traffic Class octet in a IPv6 header.
         The value of this object must be a decimal integer in the
         range from 0 to 255. This option can be used to determine
         what effect an explicit DS field setting has on a
         traceroute response. Not all values are legal or
         meaningful. Useful TOS octet values are probably
         '16' (low delay) and '8' (high throughput).
         Further references can be found in the
         RFC 2474 for the definition of the Differentiated
         Services (DS) field and to the RFC 1812 Section 5.3.2
         for Type of Service (TOS).
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte"/>
     </xs:simpleType>
     <xs:simpleType name="_CtlIfIndex">
       <xs:annotation>
         <xs:documentation>Specifies the inferface index used in the
           traceroute operation for sending the traceroute
           probes. A value of zero for this object implies
           that the interface was unknown.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte"/>
     </xs:simpleType>

     <xs:simpleType name="_CtlMiscOptions">
```

```
       <xs:annotation>
         <xs:documentation>Specifies implementation dependent
           options.</xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:string">
         <xs:maxLength value="100"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_CtlMaxFailures">
       <xs:annotation>
         <xs:documentation>Specifies the maximum number of
         consecutive timeouts allowed before terminating a
         traceroute operation. A value of either 255
         (maximum hop count/possible TTL value) or a 0 indicates
         that the function of terminating a remote traceroute
         operation when a specific number of consecutive
         timeouts are detected was disabled. This element is
         included to give full compatibility with DISMAN working
         group documents. No known implementation of traceroute
         currently supports it.
         Units are: timeouts.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte"/>
     </xs:simpleType>

     <xs:simpleType name="_CtlDontFragment">
```

```
        <xs:annotation>
          <xs:documentation>Specifies if the don't fragment flag
          (DF) in the IP header for a probe was enabled or not.
          Setting the DF flag can be used for performing
          a manual PATH MTU test.
          </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:boolean"/>
      </xs:simpleType>

      <xs:simpleType name="_CtlInitialTtl">
        <xs:annotation>
          <xs:documentation>Specifies the initial TTL value
          used in a traceroute operation. Such
          TTL setting is intended to bypass the initial
          (often well known) portion of a path.
          </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:unsignedByte">
          <xs:minExclusive value="0"/>
```

```
        </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="_CtlDescr">
        <xs:annotation>
          <xs:documentation>The purpose of this element is to
          provide a description of the traceroute test.
          </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
          <xs:maxLength value="100"/>
        </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="_CtlType">
        <xs:annotation>
          <xs:documentation>Specifies the implementation method
          used for the traceroute operation. It specifies if
          the traceroute is using TCP, UDP or ICMP probes.
          </xs:documentation>
        </xs:annotation>
```

```
       <xs:restriction base="xs:string">
         <xs:enumeration value="TCP"/>
         <xs:enumeration value="UDP"/>
         <xs:enumeration value="ICMP"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_Index">
       <xs:annotation>
         <xs:documentation>Specifies an index that consecutively
         numbers all probes for which a reply was received in the
         sequential order in which the replies were received.
         The maximum value for this object is
         CtlMaxTtl*CtlProbesPerHop.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedShort">
         <xs:minExclusive value="0"/>
         <xs:maxExclusive value="2551"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_HopIndex">
       <xs:annotation>
         <xs:documentation>Specifies which hop in a traceroute
         path that the probe's results are for. The value of
         this element is initially determined by the value of
```

```
         CtlInitialTtl.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte">
         <xs:minExclusive value="0"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_IndexPerHop">
       <xs:annotation>
         <xs:documentation>Specifies the index of a probe for
         a particular hop in a traceroute path. The number of
         probes per hop is determined by the value of the
         corresponding CtlProbesPerHop element.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedByte">
         <xs:minExclusive value="0"/>
         <xs:maxExclusive value="11"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_HopGeoLocation">
       <xs:annotation>
         <xs:documentation>Specifies the geo location of a hop
         in the traceroute path.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:string">
         <xs:maxLength value="100"/>
       </xs:restriction>
     </xs:simpleType>

    <xs:simpleType name="_MPLSTopLabel">
       <xs:annotation>
         <xs:documentation>Specifies the top entry of the MPLS label
         stack of a probe observed when the probe arrived at the hop
         that replied to the probe.  This object contains the top
         MPLS label stack entry as 32 bit value as it is observed on
         the MPLS label stack. Contained in this single number are
         the MPLS label, the Exp field, the S flag, and the MPLS TTL
         value as specified in RFC 3032 [RFC3032].
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:unsignedInt">
         <xs:maxExclusive value="4294967296"/>
       </xs:restriction>
     </xs:simpleType>
```

```
     <xs:simpleType name="_probeRoundTripTime">
       <xs:restriction base="xs:unsignedShort">
         <xs:maxExclusive value="60001"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_probeRoundTripTimeNotAvailable">
```

```
       <xs:restriction base="xs:string">
         <xs:enumeration value="NotAvailable"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:simpleType name="_ResultsHopRawOutputData">
       <xs:annotation>
         <xs:documentation>Specifies the raw output data returned
         by the traceroute operation for a certain hop in a
         traceroute path.
         </xs:documentation>
       </xs:annotation>
       <xs:restriction base="xs:string">
         <xs:maxLength value="200"/>
       </xs:restriction>
     </xs:simpleType>

     <xs:complexType name="_inetAddressASNumber">
       <xs:annotation>
         <xs:documentation>Specifies the AS number of a hop in the
         traceroute path as a 24 bit number and the indication how
         the mapping from IP address to AS number was performed.
         </xs:documentation>
       <xs:sequence>
         <xs:element name="asNumber"
                     type="_asNumber"/>
         <xs:element name="ipASNumberMappingType"
                     type="_ipASNumberMappingType"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="_inetAddress">
       <xs:choice>
         <xs:element name="inetAddressUnknown"
                     type="_zeroLengthString"/>
         <xs:element name="inetAddressIpv4"
                     type="_inetAddressIpv4"/>
         <xs:element name="inetAddressIpv6"
                     type="_inetAddressIpv6"/>
         <xs:element name="inetAddressDns"
                     type="_inetAddressDns"/>
```

```
    <xs:element name="inetAddressASNumber"
                type="_inetAddressASNumber"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="_inetAddressWithoutDns">
  <xs:choice>
    <xs:element name="inetAddressUnknown"
                type="_zeroLengthString"/>
    <xs:element name="inetAddressIpv4"
                type="_inetAddressIpv4"/>
    <xs:element name="inetAddressIpv6"
                type="_inetAddressIpv6"/>
    <xs:element name="inetAddressASNumber"
                type="_inetAddressASNumber"/>
    <xs:element name="zeroLengthString"
                type="_zeroLengthString"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="_dateAndTime">
  <xs:sequence>
    <xs:element name="dateAndTimeUpToSeconds"
                type="_dateAndTimeUpToSeconds"/>
    <xs:element name="timeMilliseconds"
                type="_timeMilliseconds"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_CtlTargetAddressType">
  <xs:annotation>
    <xs:documentation>Specifies the type of host address
    used in the traceroute command.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="targetAddressType"
                type="inetAddressType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_CtlTargetAddress">
  <xs:annotation>
    <xs:documentation>Specifies the host address used in
    the traceroute command. The host address type can be
    determined by the examining the value of the
    corresponding CtlTargetAddressType.
    </xs:documentation>
```

```
          </xs:annotation>
          <xs:sequence>
            <xs:element name="targetAddress" type="_inetAddress"/>
          </xs:sequence>
        </xs:complexType>

        <xs:complexType name="_CtlSourceAddressType">
          <xs:annotation>
            <xs:documentation>Specifies the type of the source address,
            CtlSourceAddress, used when performing the
            traceroute operation.
            </xs:documentation>
          </xs:annotation>
          <xs:sequence>
            <xs:element name="sourceAddressType"
                        type="inetAddressTypeWithoutDns"/>
          </xs:sequence>
        </xs:complexType>

        <xs:complexType name="_CtlSourceAddress">
          <xs:annotation>
            <xs:documentation>Specifies the IP address (which has
            to be given as an IP number, not a hostname) as the
            source address used in outgoing probe packets. On hosts
            with more than one IP address, this option can be used
            to force the source address to be something other than
            the primary IP address of the interface the probe packet
            is sent on. A zero length octet string value for this
            object means that source addres specification was disabled.
            The address type (InetAddressType) that relates to this
            object is specified by the corresponding value of
            CtlSourceAddressType.
            </xs:documentation>
          </xs:annotation>
          <xs:sequence>
            <xs:element name="sourceAddress"
                        type="_inetAddressWithoutDns"/>
          </xs:sequence>
        </xs:complexType>

        <xs:complexType name="_ResultsStartDateAndTime">
          <xs:annotation>
            <xs:documentation>Specifies the date and start time of the
```

```
         traceroute operation. This is the time when the first probe
         was sent.
         </xs:documentation>
       </xs:annotation>
       <xs:sequence>
```

```
         <xs:element name="dateAndTime" type="_dateAndTime"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="_ResultsIpTgtAddrType">
       <xs:annotation>
         <xs:documentation>Specifies the type of address stored
         in the corresponding ResultsIpTgtAddr element.
         </xs:documentation>
       </xs:annotation>
       <xs:sequence>
         <xs:element name="ipTgtAddrType"
                     type="inetAddressTypeWithoutDns"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="_ResultsIpTgtAddr">
       <xs:annotation>
         <xs:documentation>Specifies the IP address associated
         with a CtlTargetAddress value when the
         destination address is specified as a DNS name.
         The value of this object should be a zero length octet
         string when a DNS name is not specified or when a
         specified DNS name fails to resolve.
         </xs:documentation>
       </xs:annotation>
       <xs:sequence>
         <xs:element name="ipTgtAddr"
                     type="_inetAddressWithoutDns"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="_HopAddrType">
       <xs:annotation>
         <xs:documentation>Specifies the type of address stored
         in the corresponding HopAddr element.
         </xs:documentation>
       </xs:annotation>
       <xs:sequence>
         <xs:element name="probeHopAddrType"
                     type="inetAddressTypeWithoutDns"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="_HopAddr">
       <xs:annotation>
         <xs:documentation>Specifies the address of a hop in
         the traceroute path. This object is not allowed to
```

```
         be a DNS name. The value of the corresponding object,
         HopAddrType, indicates this object's IP address type.
         </xs:documentation>
       </xs:annotation>
       <xs:sequence>
         <xs:element name="probeHopAddr"
                     type="_inetAddressWithoutDns"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="_RoundTripTime">
       <xs:annotation>
         <xs:documentation>Specifies the amount of time measured
         in milliseconds from when a probe was sent to when its
         response was received or when it timed out. The value
         of this element is reported as the truncation of the
         number reported by the traceroute tool (the output
         "&lt1 ms" is therefore encoded as 0 ms).
         A string with the value of "RoundTripTimeNotAvailable"
         means either the probe was lost because of a timeout
         or it was not possible to transmit a probe.
         Units are: milliseconds.
         </xs:documentation>
       </xs:annotation>
       <xs:choice>
           <xs:element name="probeRoundTripTime"
                       type="_probeRoundTripTime"/>
           <xs:element name="probeRoundTripTimeNotAvailable"
                       type="_probeRoundTripTime"/>
```

```
        </xs:choice>
      </xs:complexType>

      <xs:complexType name="_ResponseStatus">
        <xs:annotation>
          <xs:documentation>Specifies the result of a traceroute
            operation made by the host for a particular probe.
          </xs:documentation>
        </xs:annotation>
        <xs:sequence>
          <xs:element name="probeResponseStatus"
                      type="operationResponseStatus"/>
        </xs:sequence>
      </xs:complexType>

      <xs:complexType name="_Time">
        <xs:annotation>
          <xs:documentation>Specifies the timestamp when the
            response to the probe was received.
```

```
          </xs:documentation>
        </xs:annotation>
        <xs:sequence>
          <xs:element name="dateAndTime" type="_dateAndTime"/>
        </xs:sequence>
      </xs:complexType>

      <xs:complexType name="_ResultsProbe">
        <xs:sequence>
          <xs:element name="Index"
                      type="_Index"/>
          <xs:element name="HopIndex"
                      type="_HopIndex"/>
          <xs:element name="IndexPerHop"
                      type="_IndexPerHop"/>
          <xs:element name="HopAddrType"
                      type="_HopAddrType"/>
          <xs:element name="HopAddr"
                      type="_HopAddr"/>
          <xs:element name="HopGeoLocation"
                      type="_HopGeoLocation"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="MPLSTopLabel"
                      type="_MPLSTopLabel"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="RoundTripTime"
                      type="_RoundTripTime"/>
          <xs:element name="ResponseStatus"
                      type="_ResponseStatus"/>
          <xs:element name="Time"
                      type="_Time"/>
        </xs:sequence>
      </xs:complexType>

      <xs:complexType name="_ResultsEndDateAndTime">
        <xs:annotation>
          <xs:documentation>Specifies the date and end time of
          the traceroute operation. It is either the time when
          the response to the last probe of the traceroute
          operation was received or the time when the last
          probe of the traceroute operation was sent plus the
          relative timeout (in case of missing response).
          </xs:documentation>
        </xs:annotation>
        <xs:sequence>
          <xs:element name="dateAndTime" type="_dateAndTime"/>
        </xs:sequence>
      </xs:complexType>
```

```
    <xs:complexType name="_Metadata">
      <xs:annotation>
        <xs:documentation>Specifies the metadata for a
        traceroute operation. In a request, these are the
        requested parameters. In a response, they are the
        actual parameters used.
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="TestName"
                    type="_TestName"/>
        <xs:element name="OSName"
                    type="_OSName"
                    minOccurs="0" maxOccurs="1"/>
        <xs:element name="OSVersion"
```

```
                     type="_OSVersion"
                     minOccurs="0" maxOccurs="1"/>
         <xs:element name="ToolVersion"
                     type="_ToolVersion"
                     minOccurs="0" maxOccurs="1"/>
         <xs:element name="CtlTargetAddressType"
                     type="_CtlTargetAddressType"/>
         <xs:element name="CtlTargetAddress"
                     type="_CtlTargetAddress"/>
         <xs:element name="CtlByPassRouteTable"
                     type="_CtlByPassRouteTable"
                     minOccurs="0" maxOccurs="1"
                     default="false"/>
         <xs:element name="CtlProbeDataSize"
                     type="_CtlProbeDataSize"
                     minOccurs="0" maxOccurs="1" default="0"/>
         <xs:element name="CtlTimeOut"
                     type="_CtlTimeOut" minOccurs="0"
                     maxOccurs="1" default="3"/>
         <xs:element name="CtlProbesPerHop"
                     type="_CtlProbesPerHop"
                     minOccurs="0" maxOccurs="1" default="3"/>
         <xs:element name="CtlPort"
                     type="_CtlPort" minOccurs="0"
                     maxOccurs="1" default="33434"/>
         <xs:element name="CtlMaxTtl"
                     type="_CtlMaxTtl" minOccurs="0"
                     maxOccurs="1" default="30"/>
         <xs:element name="CtlDSField"
                     type="_CtlDSField" minOccurs="0"
                     maxOccurs="1" default="0"/>
         <xs:element name="CtlSourceAddressType"
                     type="_CtlSourceAddressType"/>
```

```
         <xs:element name="CtlSourceAddress"
                     type="_CtlSourceAddress"/>
         <xs:element name="CtlIfIndex"
                     type="_CtlIfIndex" minOccurs="0"
                     maxOccurs="1" default="0"/>
         <xs:element name="CtlMiscOptions"
                     type="_CtlMiscOptions" minOccurs="0"
                     maxOccurs="1"/>
```

```
        <xs:element name="CtlMaxFailures"
                    type="_CtlMaxFailures" minOccurs="0"
                    maxOccurs="1" default="5"/>
        <xs:element name="CtlDontFragment"
                    type="_CtlDontFragment" minOccurs="0"
                    maxOccurs="1" default="false"/>
        <xs:element name="CtlInitialTtl"
                    type="_CtlInitialTtl" minOccurs="0"
                    maxOccurs="1" default="1"/>
        <xs:element name="CtlDescr"
                    type="_CtlDescr" minOccurs="0"
                    maxOccurs="1"/>
        <xs:element name="CtlType"
                    type="_CtlType" minOccurs="0"
                    maxOccurs="1"
                    default="UDP"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="_Measurement">
      <xs:annotation>
        <xs:documentation>
          Contains the actual traceroute measurement.
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="TestName"
                    type="_TestName"/>
        <xs:element name="ResultsStartDateAndTime"
                    type="_ResultsStartDateAndTime"/>
        <xs:element name="ResultsIpTgtAddrType"
                    type="_ResultsIpTgtAddrType"/>
        <xs:element name="ResultsIpTgtAddr"
                    type="_ResultsIpTgtAddr"/>
        <xs:element name="ResultsProbe"
                    type="_ResultsProbe"
                    minOccurs="1" maxOccurs="2550"/>
        <xs:element name="ResultsHopRawOutputData"
                    type="_ResultsHopRawOutputData"
                    minOccurs="0" maxOccurs="255"/>
```

```
        <xs:element name="ResultsEndDateAndTime"
```

```
                     type="_ResultsEndDateAndTime"/>
       </xs:sequence>
     </xs:complexType>

     <xs:complexType name="_traceRoute">
       <xs:choice>
         <xs:sequence>
           <xs:element name="Request"
                       type="_Metadata"/>
         </xs:sequence>
         <xs:sequence>
           <xs:element name="MeasurementMetadata"
                       type="_Metadata"/>
         </xs:sequence>
         <xs:sequence>
           <xs:element name="Measurement"
                       type="_Measurement"/>
         </xs:sequence>
         <xs:sequence>
           <xs:element name="Request"
                       type="_Metadata"/>
           <xs:element name="MeasurementMetadata"
                       type="_Metadata"/>
         </xs:sequence>
         <xs:sequence>
           <xs:element name="Request"
                       type="_Metadata"/>
           <xs:element name="Measurement"
                       type="_Measurement"/>
         </xs:sequence>
         <xs:sequence>
           <xs:element name="MeasurementMetadata"
                       type="_Metadata"/>
           <xs:element name="Measurement"
                       type="_Measurement"/>
         </xs:sequence>
         <xs:sequence>
           <xs:element name="Request"
                       type="_Metadata"/>
           <xs:element name="MeasurementMetadata"
                       type="_Metadata"/>
           <xs:element name="Measurement"
                       type="_Measurement"/>
         </xs:sequence>
       </xs:choice>
     </xs:complexType>
     <!--Reference to "traceRoute" element-->
     <xs:element name="traceRoute" type="_traceRoute"/>

   </xs:schema>
```

8. Differences to DISMAN-TRACEROUTE-MIB

For performing remote traceroute operations at managed node, the IETF has standardized the DISMAN-TRACEROUTE-MIB module in RFC 4560 [RFC4560]. This module allows:

- retrieving capability information of the traceroute implementation at the managed node,
- configuring traceroute operations to be prformed,
- retrieving information about ongoing and completed traceroute measurements,
- retrieving traceroute measurement statistics.

The traceroute storage format described in this document has significant overlaps with this MIB module. Particularly, the models for the traceroute measurement configuration and for the result from completed measurements are almost identical. But for other pats of the DISMAN-TRACEROUTE MIB module there is no need to model them in a traceroute storage format. Particularly, the capability information, information about ongoing measurements and measurement statistics are not covered by the traceroute storage model.

Concerning traceroute measurements and results, there are structural differences between the two models caused by the different choices for the encoding of the specification. For DISMAN-TRACEROUTE-MIB, the Structure of Management Information (SMIv2, STD 58, RFC 2578 [RFC2578]) was used, while for the traceroute storage format is encoded using XML.

This difference in structure implies that the DISMAN-TRACEROUTE-MIB module contains SMI-specific information element (managed objects) that concern tables of managed objects (specification, entry creation and delection, status retrieval) that are not required for the XML-encoded traceroute storage format.

But for most of the remaining information elements that concern configuration of traceroute measurements and results of completed measurements, the semantics is identical between the DISMAN-TRACEROUTE-MIB module and the traceroute storage format. There are very few exceptions to this which are listed below. Also naming of information elements is identical between both models with a few

exceptions. For the traceroute storage model, a few information elements have been added, some because of the different structure and some to provide additional information on completed measurements.

8.1. Naming

Basically, names in both models are chosen using the same naming conventions.

For the traceroute measurement configuration information all names, such as CtlProbesPerHop, are identical in both models except for the traceRoute prefix that was removed to avoid unecessary redundancy in the XML file and for CtlDataSize which was renamed to CtlProbeDataSize for clarification in the traceroute storage model.

Results of measurements in the DISMAN-TRACEROUTE-MIB modules are distributed over two tables, the traceRouteResultsTable containing mainly information about ongoing measurements and the traceRouteProbeHistoryTable containing only information about completed measurements. According to the SMIv2 naming conventions names of information elements in these tables have different prefixes (traceRouteResults and traceRouteProbeHistory). Since the traceroute storage format only reports on completed measurements, this separation is not needed anymore and the prefix "Results" is used for all related information elements.

Beyond that, there are only a few changes in element names. The renaming actions include:

- traceRouteProbeHistoryProbeIndex to IndexPerHop,
- traceRouteProbeHistoryResponse to RoundTripTime,
- traceRouteProbeHistoryTime to ResultsEndDateAndTime,
- traceRouteProbeHistoryLastRC to ResultsHopRawOutputData.

8.2. Semantics

The semantics was changed for two information elements only.

For traceRouteProbeHistoryResponse in the DISMAN-TRACEROUTE-MIB, a value of 0 indicated, that it was not possible to transmit a probe. For the traceroute strorage format, a value of 0 for element RoundTripTime indicates that the measured time was less than one millisecond, while for the case that it was not possible to transmit a probe a string is used that indicates the problem.

For traceRouteCtlIfIndex in the DISMAN-TRACEROUTE-MIB, a value of 0 indicated, that it the option to set the index is not available. This was translated to the traceroute strorage format, such that a

value of 0 for this element indicates that the used interface is unknown.

The element traceRouteProbeHistoryLastRC in the DISMAN-TRACEROUTE-MIB was replaced by element ResultsHopRawOutputData. While traceRouteProbeHistoryLastRC just reports a reply code, ResultsHopRawOutputData reports the full raw output data produced by the traceroute instance that was used.

8.3. Additional Information Elements

Only a few information elements have been added to the model of the DISMAN-TRACEROUTE-MIB module.

- For providing geographical information about hops in the traceroute path, HopGeoLocation was added.
- For providing the top MPLS label stack entry of a probe in the traceroute path MPLSTopLabel was added.
- For providing additional timestamp beyond ResultsEndDateAndTime, ResultsStartDateAndTime and Time were added.

9. Security Considerations

Security considerations in this section discuss are grouped into considerations related to conducting traceroute measurements and considerations related to storing and transmitting results of measurements.

This memo does not specify an implementation of a traceroute measurements. Neither does it specify a certain procedure for storing traceroute measurement results. Still it is considered desirable to discuss related security issues below.

9.1. Conducting Traceroute Measurements

Conducting Internet measurements can raise both security and privacy concerns. Traceroute measurements, in which traffic is injected into the network, can be abused for denial-of-service attacks disguised as legitimate measurement activity.

Measurement parameters MUST be carefully selected so that the measurements inject trivial amounts of additional traffic into the networks they measure. If they inject "too much" traffic, they can skew the results of the measurement, and in extreme cases cause congestion and denial of service.

The measurements themselves could be harmed by routers giving

measurement traffic a different priority than "normal" traffic, or by an attacker injecting artificial measurement traffic. If routers can recognize measurement traffic and treat it separately, the measurements will not reflect actual user traffic. If an attacker injects artificial traffic that is accepted as legitimate, the loss rate will be artificially lowered. Therefore, the measurement methodologies SHOULD include appropriate techniques to reduce the probability measurement traffic can be distinguished from "normal" traffic.

Authentication techniques, such as digital signatures, may be used where appropriate to guard against injected traffic attacks.

9.2. Securing Traceroute Measurement Results

Traceroute results are not considered highly sensible. Still, they may contain sensible information on network paths, routing states, use IP addresses, and roundtrip times, that the operator a networks may want to detect for business or security reasons.

It is thus important to control access to Information acquired by conducting traceroute measurement, particularly when transmitting it over a networks but also when storing it. It is RECOMMENDED that transmission of traceroute measurement results over a network uses appropriate protection mechanisms for preserving privacy, integrity and authenticity. It is further RECOMMENDED that secure authentication and authorization are used for protecting stored traceroute results.

10. IANA Considerations

This document uses URNs to describe an XML namespace and an XML schema for traceroute measurements conforming to a registry mechanism described in [RFC3688]. Two URI assignments are requested.

1. Registration request for the IPPM traceroute measurements namespace
   - URI: urn:ietf:params:xml:ns:traceroute-1.0
   - Registrant Contact: TBD.
   - XML: None. Namespace URIs do not represent an XML
2. Registration request for the IPPM traceroute measurements schema
   - URI: urn:ietf:params:xml:schema:traceroute-1.0
   - Registrant Contact: TBD.
   - XML: See the section Section 7 of this document.

11. References

11.1. Normative References

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

11.2. Informative References

[I-D.ietf-disman-remops-mib-v2]
Quittek, J. and K. White, "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", draft-ietf-disman-remops-mib-v2-09 (work in progress), February 2006.

[I-D.ietf-ipfix-architecture]
Sadasivan, G., "Architecture for IP Flow Information Export", draft-ietf-ipfix-architecture-12 (work in progress), September 2006.

[I-D.ietf-ipfix-info]
Quittek, J., "Information Model for IP Flow Information Export", draft-ietf-ipfix-info-14 (work in progress), October 2006.

[I-D.ietf-ipfix-protocol]
Claise, B., "Specification of the IPFIX Protocol for the Exchange", draft-ietf-ipfix-protocol-24 (work in progress), November 2006.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC3417] Presuhn, R., "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, December 2002.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4560] Quittek, J. and K. White, "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", RFC 4560, June 2006.

[XML] Yergeau et al., F., "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation, February 2004.

Authors' Addresses

Saverio Niccolini
Network Laboratories, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 4342 118
Email: saverio.niccolini@netlab.nec.de
URI: http://www.netlab.nec.de

Sandra Tartarelli
Network Laboratories, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 4342 132
Email: sandra.tartarelli@netlab.nec.de
URI: http://www.netlab.nec.de

Juergen Quittek
Network Laboratories, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115

Germany

Phone: +49 (0) 6221 4342 115
Email: quittek@netlab.nec.de
URI: http://www.netlab.nec.de

Martin Swany
Dept. of Computer and Information Sciences, University of Delaware
Newark DE 19716
U.S.A.

Email: swany@UDel.Edu

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).