

Outstanding Issues with “Current” NM-WG Schemas

Introduction

During the last 12-18 months NM-WG have been working on a pair of XML schemas for querying for network measurement data, and publishing the results of those queries.

While the basic format of the schemas has been stable for some time there are still a number of outstanding issues, which have been hindering the efforts of early adopters. This very informal document summaries the issues, and suggests a course of action for each, mainly resulting from discussions held between Mark and Paul in December.

Issues are attributed to the following people:

Mark:	Mark Leese, m.j.leese@dl.ac.uk
Paul:	Paul Mealor, pdm@hep.ucl.ac.uk
Rik:	Rik Tyer, r.tyer@dl.ac.uk
Tanya:	Tanya Brethour, brethour@ncsa.uiuc.edu
Yee:	Yee-Ting Li, ytl@hep.ucl.ac.uk

Response Schema

Mark: nm_report.rnc declares result as:

```
element result {Result} *,
```

Can you have zero or more (*) results?

Response: Not sure. Paul thought that Dan might actually be planning on returning zero results to indicate that there was no data satisfying the request, rather than returning a fault to indicate the same thing.

Verdict: Ask Dan.

Rik: What is the default time format?

Response: This is a business logic issue, as I don't think we can enforce a default format in the schema itself.

While we recognise that in certain cases it would be useful to be able to specify times using NTP timestamps and the built-in XML time type (the requirements state that this should be possible) we think seconds since the epoch is the most sensible choice.

Verdict: Settled, unless anyone has objections. This will need adding to the business logic.

Yee: The time information doesn't have a stated format. It should have an attribute or another element that indicates the time format, e.g. "seconds since the epoch".

Response: Agreed, and Paul was going to add an attribute to absolute time elements for this purpose. However, it never happened.

Nothing need be done with the time tolerance elements. The requirements state that they should be in seconds, with resolution to at least micro seconds.

Verdict: If no one complains, Mark will add attributes for specifying the time format.

Rik: What's the purpose of "release" in nm_tool.rnc?

Response: Say you're using version 3.2-5 of a particular tool. Major = 3, minor = 2 and release = 5.

Yee: The only other thing for Paul and I (from a coding point of view) is that it would have been nice if we didn't have to put each statistical value within a pair of <statistic> tags. This represents an anomaly between representing raw values (singletons) and statistical data.

Consider the following snip from a real life example, which shows two statistical values (min and max) and raw data value being returned:

```
<rep:statistic>
  <rep:timeInterval>
    <rep:timestamp>1101906341</rep:timestamp>
  </rep:timeInterval>
  <rep:name>min</rep:name>
  <rep:value>
    4.4941601E7
    <rep:units>b/s</rep:units>
  </rep:value>
</rep:statistic>
<rep:statistic>
  <rep:timeInterval>
    <rep:timestamp>1101906341</rep:timestamp>
  </rep:timeInterval>
  <rep:name>max</rep:name>
  <rep:value>
    8.5019068E7
    <rep:units>b/s</rep:units>
  </rep:value>
</rep:statistic>
<rep:result>
  <rep:timeInterval>
    <rep:timestamp>1101328638</rep:timestamp>
  </rep:timeInterval>
  <rep:achievableBandwidth>
    <rep:achievableThroughput>
      8.4870385E7
      <rep:units>b/s</rep:units>
    </rep:achievableThroughput>
  </rep:achievableBandwidth>
</rep:result>
```

Each piece of statistical data is returned within <statistic> tags. The type of statistic must also be identified within the tags (e.g. mean, median, min, max). Raw data is contained with <result> tags. A further simplified example:

```
<!-- two values of raw data -->
<result>
</result>
<result>
</result>
<!-- two values of statistical (min) data -->
<statistic>
  <min>
```

```

    </min>
  </statistic>
</statistic>
  <min>
    </min>
  </statistic>

```

Not only is this awkward for the system that has to prepare the data to send back, but it is also hard for Yee to interrogate the data coming back with XQuery or XPath (I forget which). He ends up with IF THEN statements everywhere, instead of treating all data as the same, and just changing a search term from "raw" to "mean" or whatever.

As Yee suggests, the following arrangement would be easier to code, and less XML to be transmitted, stored, processed etc.

```

<raw>
  <measurement>
  </measurement>
  <measurement>
  </measurement>
</raw>
<min>
  <measurement>
  </measurement>
  <measurement>
  </measurement>
</min>

```

Response: People have already produced some implementations, and it would be difficult to modify them. So we are probably best leaving this as is. However, there's nothing to stop us changing it in the future, e.g. with the new schemas.

Verdict: Leave alone, unless there are objections. Flag this as something to consider with the new schemas.

Yee: When specifying the attributes of elements, the request schema is heavily dependant on the use of Relax-NG attributes, whereas the response schema avoids Relax-NG attributes altogether, and uses separate elements to specify "attributes" of objects. If we consider a specific specific type of result element in the response schema...

```

<rep:result>
  <!-- time info goes here -->
  <rep:achievableThroughput>
    8.315039E7
    <rep:units>mb/s<rep:units>
  </rep:achievableThroughput>
</rep:result>

```

...why are the units of achievableThroughput expressed as an element, and not simply as an attribute of the "result" or "achievableThroughput" elements? Are we going to change anything or are we better leaving things as they are.

Response: We think the response schema was defined in this way for a reason, but can't remember why. We think it may be because lots of attributes upset some web services/XML tooling, but considering we expect these schemas to be superseded by the new versions, we also think the current schemas can be left as is. If for no other reason, it will show us which is best:

- elements only
- element + attributes

Verdict: Ask Dan why, but in all likelihood, leave alone.

Tanya: How can you represent a hop list using the response schema?

Response: Err, good question. We couldn't find a way with the schema as it was. hopList (in results.rnc) was defined as:

```
HopList =  
  ## Number of hops  
  element hopCount { xsd:int },  
  element units { token }
```

Which looks like it only gives you the ability to report the number of hops.

We also thought of introducing a structure like:

```
<HopList>  
  <hop>  
    <source>A</source>  
    <destination>B</destination>  
  </hop>  
  <hop>  
    <source>A</source>  
    <destination>C</destination>  
  </hop>  
  <hop>  
    <source>A</source>  
    <destination>C</destination>  
  </hop>  
  <!-- router D returns no response, so we have no data -->  
  <destination></destination>  
  </hop>  
  <hop>  
    <source>A</source>  
    <destination>E</destination>  
  </hop>  
</HopList>
```

This would give you the ability to report the hops along a path, however, what we really wanted was to integrate a hop list into the results section, so that you can specify a hoplist and also give the data associated with each hop, like the RTT you would get to each node along a path when using traceroute.

A more complete example is given below. It's based on some real NM-WG output (from the EGEE JRA4 project) and has been tweaked to show how this could work. Assume we're performing a traceroute type test from rtvig.dl.ac.uk to gw-man.netnw.net.uk in the style, nodes A-B, A-C, A-D, A-E etc. Also assume the results look like this:

```
1    <10 ms    <10 ms    <10 ms    alan3.dl.ac.uk [148.79.1.1]  
2    15 ms     15 ms     15 ms     gw-fw.dl.ac.uk [193.63.74.131]  
3    *         *         *         Request timed out.  
4    25 ms     25 ms     25 ms     gw-man.netnw.net.uk [194.66.25.98]
```

```
<rep:networkMeasurementReport  
  xmlns:rep="http://www.ggf.org/namespaces/2004/01/gridNetworkMonitoring">  
  <rep:id>000</rep:id>  
  <rep:version>2004-01</rep:version>  
  <rep:networkMeasurement>  
    <rep:characteristic>path.hopList</rep:characteristic>  
    <rep:subject>  
      <rep:path>  
        <rep:source>  
          <rep:address>  
            <rep:host>rtvig.dl.ac.uk</rep:host>
```

```

        </rep:address>
    </rep:source>
    <rep:destination>
        <rep:address>
            <rep:host>gw-man.netnw.net.uk</rep:host>
        </rep:address>
    </rep:destination>
</rep:path>
</rep:subject>
<rep:methodology>
    <rep:tool>
        <rep:name>traceroute</rep:name>
    </rep:tool>
</rep:methodology>

<!-- Here's the results -->

<rep:results>
    <rep:timeInterval>
        <!-- we've asked for a result within a +/- 5 min period of 12:00 on Christmas Day. We
        actually get a result for 12:00 dead on. The time of all <section> results is the
        same, even though each test may be separated by a few seconds. -->
        <rep:timestamp>1103993700</rep:timestamp>
        <rep:timestamp>1103994300</rep:timestamp>
    </rep:timeInterval>
    <rep:result>
        <rep:routeResults>
            <rep:section>
                <rep:subject>
                    <rep:path>
                        <rep:source><rep:host>rtvig.dl.ac.uk</rep:host><rep:version>hostname</r
ep:version></rep:source>
                        <rep:destination><rep:host>alan3.dl.ac.uk</rep:host><rep:version>hostna
me</rep:version></rep:destination>
                    </rep:path>
                </rep:subject>
                <rep:resultSet>
                    <rep:timeInterval>
                        <rep:timestamp>1103994000</rep:timestamp>
                    </rep:timeInterval>
                    <rep:result>
                        <rep:delay>
                            <rep:value>10</rep:value>
                            <rep:units>mS</rep:units>
                        </rep:delay>
                    </rep:result>
                </rep:resultSet>
                <rep:order>1</rep:order>
            </section>
            <rep:section>
                <rep:subject>
                    <rep:path>
                        <rep:source><rep:host>rtvig.dl.ac.uk</rep:host><rep:version>hostname</r
ep:version></rep:source>
                        <rep:destination><rep:host>gw-fw.dl.ac.uk</rep:host><rep:version>hostn
ame</rep:version></rep:destination>
                    </rep:path>
                </rep:subject>
                <rep:resultSet>
                    <rep:timeInterval>
                        <rep:timestamp>1103994000</rep:timestamp>
                    </rep:timeInterval>
                    <rep:result>
                        <rep:delay>
                            <rep:value>15</rep:value>
                            <rep:units>mS</rep:units>
                        </rep:delay>
                    </rep:result>
                </rep:resultSet>
                <rep:order>2</rep:order>
            </section>
        </rep:routeResults>
    </rep:result>
</rep:results>

<!-- There's no entry for the third hop because we have no data for it -->
    <rep:section>
        <rep:subject>
            <rep:path>
                <rep:source><rep:host>rtvig.dl.ac.uk</rep:host><rep:version>hostname</r
ep:version></rep:source>
            </rep:subject>
        </rep:section>
    </rep:results>

```

```

        <rep:destination><rep:host>gwman.netnw.net.uk</rep:host><rep:version>h
ostname</rep:version></rep:source>
      </rep:path>
    </rep:subject>
  <rep:resultSet>
    <rep:timeInterval>
      <rep:timestamp>1103994000</rep:timestamp>
    </rep:timeInterval>
    <rep:result>
      <rep:delay>
        <rep:value>25</rep:value>
        <rep:units>mS</rep:units>
      </rep:delay>
    </rep:result>
  </rep:resultSet>
  <rep:order>4</rep:order>
</section>
</routeResults>
</rep:results>
</rep:networkMeasurement>
</rep:networkMeasurementReport>

```

Verdict: If people are happy with this, then Mark will look at implementing it in the schema.

Tanya: It looks like there's nothing in the request schema to store AvailableBandwidth.

Response: It's there, in nm_result.rnc (honest)....

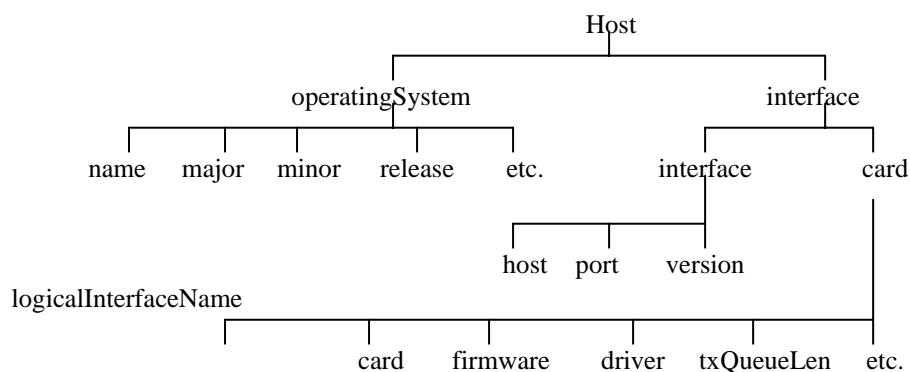
```

AchievableBandwidth =
## Indication of what is the bottleneck (network, CPU,
## NIC, memory, disk, etc.)
  element achievableBottleneck { string }?,
## The measurement
  element achievableThroughput { xsd:double, element units { token }
}

```

Tanya: "Host" is never used in the response schema.

Response: Yep, looks that way. "Host" (with a capital H) is shown below. It's never used, but it's possible Dan was going to reference it from "Node".



Verdict: Ask Dan what he planned to do.

Tanya: is anything going to happen about getting the schemas into CVS, and giving a wider group of people access?

Response: It can be done, but if we're looking to stabilise almost immediately, is there much point?

Verdict: Do nothing, unless there's feedback.

Minor Questions For Dan

1. The parameters defined in nm_param.rnc are separated by commas, meaning their order must be maintained. Should we separate them with ampersands (interleaving) so they can appear in any order?
2. The response schema includes nm_request.rnc. Should it?
3. Do we need to add a way of storing units for statistical data? It would make sense in some cases, e.g. the mean of RTT will still be in seconds.
4. Is the way of representing results in results.rnc sometimes inconsistent? For example, delay is defined as...

```
element delay { Delay }
```

...where Delay is defined as...

```
Delay =  
  element value { xsd:double },  
  element units { token }
```

This would give us real XML like...

```
<delay>  
  <value>10</value>  
  <units>ms</ units >  
</delay>
```

i.e. the actual data is in "value" tags. However, the definition of pathLoss is...

```
element pathLoss { PathLoss }
```

...where PathLoss is defined as...

```
PathLoss =  
  ## Number of packets since the previous loss (See RFC3357)  
  element lossDistance { xsd:int, element units { token } },  
  ## Number of groups of lost packets (See RFC3357)  
  element lossPeriod { xsd:int, element units { token } },  
  ## Percent of packets lost where if the distance between  
  ## the lost packet and the previously lost packet is no  
  ## greater than the "loss constraint" (See RFC3357)  
  element noticeableRate { xsd:int, element units { token } },  
<!-- ETC. ETC. -->
```

Which would give us real XML like...

```
<pathLoss>  
  <lossDistance>  
    100  
    <units>packets</units>  
  </lossDistance>  
  <lossPeriod>  
    5  
    <units>%</units>  
  </lossPeriod>
```

```
<!-- ETC. ETC. -->
  </pathLoss>
```

i.e. the actual data is not in “value” tags.

Request Schema

Yee: Using a token (essentially a string containing no white space) for “characteristic” is too open ended. Requestors could put anything in there, which the receiver then has to validate.

Response: This was done to allow people to use their own characteristics if they wanted. We didn’t restrict things with an enumeration type so that people were free to use their own characteristics if they wanted. You would have to match an incoming request to the metrics you support at some stage (e.g. using a case statement) so you could do the validation then.

As a slight aside, Paul also thought that based on discussions during one call, we’d specifically avoided using enumeration types because there was some software (web services tooling) that they cause to fail. Looking in the response schema, it comments that enumerations “break” wsdl2py, a Python tool for (amongst other things) generating client interface code from a wsdl definition.

Verdict: No change.

Rik: Are there problems associated with not being able to completely specify a request using just the characteristic? For example, as well as giving path.bandwidth.achievable as the characteristic, you would need to specify the protocol (TCP, UDP...) as <packetType> in the <methodology> section.

Response: We had lengthy discussions on this as a group, and the conclusion was characteristic should be protocol independent, e.g. <characteristic>path.bandwidth.achievable.TCP</characteristic> would be illegal. The reasoning was that while putting the protocol with the characteristic name would be useful in some cases, in general it would make the characteristic name too complicated:

- It was going to set a precedent which would have to be followed with other characteristics names, e.g. path.delay.oneWay.ICMP for ping tests.
- It conflicted with the response schema as it was then, which had a separate <packetTypeParam> element.
- It conflicted with what seemed the logical way to structure the request schema. That is, specify a characteristic, and separately define any test specific parameters, inc. protocol, buffer size, number of streams etc.

The only real problem is that it conflicts with Eric’s idea of being able to specify a request that contains the bare minimum information, but Paul and I thought this the lesser of the evils.

Verdict: No change.

Rik: Why is TimeInformation in nm_requestbody.rnc not mandatory? It’s declared as “?” meaning 0 or 1 instances.

Response: This relates to Eric’s idea of being able to specify a request containing minimal information. Paul’s idea was that a request containing no time information would be interpreted as “give me a recent result” where “recent” is whatever the system satisfying the request defines it to be.

Verdict: Okay, but this needs to be clearly defined in the accompanying business logic.

Mark: An outstanding issue from the requirements document is that the pre-defined set of tags for specifying test (tool) parameters, e.g. <tcpBufferSize> still need to be defined.

Response: The request schema now all of the parameters available in the response schema (by sharing file nm_params.rnc). Appropriate text can be added to the documentation. The list is:

- duration – length of test
- packetType – e.g. TCP, UDP or ICMP
- packetSize
- numPackets
- packetSpacing – the algorithm is used to space packets, e.g. poisson or periodic
- packetGap – for periodic tests, the time between test packets
- protocolID – IP version
- tos – Type Of Service (IP precedence bits)
- IP precedence bits
- dscp – Differentiated Services Code Point
- flowLabel – the IPv6 option for QoS
- lossThreshold – the threshold used to distinguish between a large finite delay and a loss
- element numBytes – specify the amount of test traffic
- includesDisk – should data transfers be memory-to-memory or disk-to-disk?
- tcpBufferSize
- tcpType – e.g. Reno, Vegas, HSTCP
- numStreams – the number of parallel streams to use
- port

Verdict: Settled, unless anyone has objections.

Mark: Another outstanding issue from the requirements document is whether it should be possible, for a given test path, to specify one end's parameters using parameter specific tags and the other end's parameters using a list of parameters (e.g. “-p 5001 -w 1048k”).

Response: To avoid things getting messy Paul and I decided that both source and destination parameters should be specified in the same way. However, this is largely theoretical for the implementation point of view, since Paul didn't implement the parameter list elements.

Verdict: Settled from the requirements doc angle, unless anyone has objections. Mark will look at implementing the parameters lists, and making them mutually exclusive to parameter specific tags. If they can't be made mutually exclusive, then it will have to be a business logic rule.