xWAMP usage nits

Access

Yaakov (J) Stein

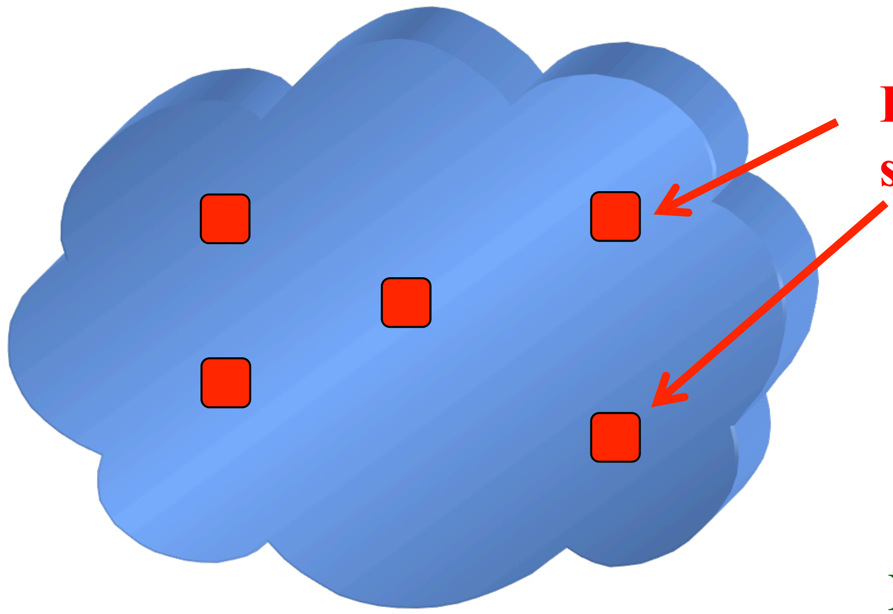
RAD Data Communications, Ltd.

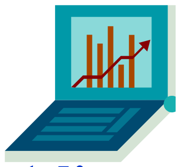
RAD data communications

SP usage case

xWAMP can be used as a protocol
for L3 **P**erformance **M**anagement (PM)
in **S**ervice **P**rovider (SP) networks

It thus complements tools for L2 PM
but needs to fit into the management scheme


PM probes (senders/receivers/reflectors)
scattered throughout the network



Note : test sessions may be expected to run continuously

General model

RFCs 4656 and 5357 describe a very general model
with client / server / sender / receiver / fetch client
in general locations (so xWAMP should be able to fit)

```
+----------------+               +------------------+
| Session-Sender |--OWAMP-Test-->| Session-Receiver |
+----------------+               +------------------+
  ^                                     ^
  |                                     |
  |                                     |
  |                                     |
  |  +----------------+<----------------+
  |  |     Server     |<-------+
  |  +----------------+        |
  |    ^                       |
  |    |                       |
  | OWAMP-Control         OWAMP-Control
  |    |                       |
  v    v                       v
+----------------+     +-----------------+
| Control-Client |     |   Fetch-Client  |
+----------------+     +-----------------+

(Unlabeled links in the figure are unspecified by this document and
 may be proprietary protocols.)
```

Specific scenario

But the specifications and protocols assume a specific scenario

Different logical roles can be played by the same host. **For example**, in the figure above, there could actually be only two hosts: one playing the roles of Control-Client, Fetch-Client, and Session- Sender, and the other playing the roles of Server and Session- Receiver. This is shown below.

```
+-----------------+                     +-------------------+
| Control-Client  |<--OWAMP-Control-->| Server              |
| Fetch-Client    |                     |                   |
| Session-Sender  |---OWAMP-Test----->| Session-Receiver  |
+-----------------+                     +-------------------+
```

This *example* scenario is tailored to two hosts
but does not match the SP network scenario

- the control and fetch clients will usually be remote from the senders and receivers
- the server may be at senders that are not also receivers

Example ?

Actually, only this *example* is fully supported by the RFCs

The required protocols for other scenarios are undefined

```
(Unlabeled links in the figure are unspecified by this
  document and may be proprietary protocols.)
```

For example

- How does a client (or server) configure a remote sender ?
- How does the server collect information
  from a remote receiver which is not co-located with a server ?

Fetch client issues

If the fetch client is NOT co-located with the sender

- How does the fetch client know when stop-sessions has been sent ?
- How does it know the SIDs ?

  it needs SIDs to identify the sessions

  but these are chosen by the receiver with a random component

Both of these can be solved by new fetch messages :
list stopped SID request, list stopped SID response

- How does the fetch client collect from test sessions in the server-client direction ?
- Why does TWAMP not define a fetch client ?

More fetch client issues

Why is port 861 re-used for the fetches ?

In embedded server implementations this requires spawning a task before knowing if it is for a new test session or simply a fetch

Alternative

In both the RFC scenario and the SP scenario
the control client and fetch client are co-located

So why can't we combine control and fetch client functionality ?

After the stop sessions command
the control client could simply request the data
using the same TCP session !

Yet more fetch client issues

RFC 4656 says

```
Begin Seq is the sequence number of the first requested packet.  End
Seq is the sequence number of the last requested packet.  If Begin
Seq is all zeros and End Seq is all ones, complete session is said to
be requested.

If a complete session is requested and the session is still in
progress or has terminated in any way other than normally, the
request to fetch session results MUST be denied.  If an incomplete
session is requested, all packets received so far that fall into the
requested range SHOULD be returned.  Note that, since no commands can
be issued between Start-Sessions and Stop-Sessions, incomplete
requests can only happen on a different OWAMP-Control connection
(from the same or different host as Control-Client).
```

So, the fetch client needn't request all collected information at once

However, the wording on fetch requests is confusing (to say the least)

Yet more fetch client issues (cont.)

1. It is strongly implied that incomplete fetch requests can only be made before the session has terminated

   Why ?

   It is useful to retrieve information on critical intervals first

2. Can incomplete requests overlap ?
   for example
   after requesting 1-20, can we request 10-30 ?

3. After stop-sessions, how does the server know when the collected information can be deleted ?
   (for embedded servers this can be a LOT of data!)

- only timeout ?
- noting that ALL data has been retrieved ?
  (see below - must fetch clients retrieve data in order ?)

Information retrieval issues

OWAMP replies contain 46 B per packet
For a 256 packet session this is almost 12K of payload
and can can not usually be sent in a single IP packet

Workarounds and solutions
- fetch client retrieves N packets at a time (in order)
- server places information into a file and client uses TFTP
- server stores information in a database and client queries
- server stores information in a MIB (similar to L2 PM)

Use of UDP port numbers

The test session UDP port numbers
SHOULD be chosen from the dynamic port range (49152-65535)
and MAY be chosen randomly per RFC 6056

Can the same *source* port number be used
for multiple test sessions
differentiated by the destination port number only ?

Stop-sessions

Is the stop-sessions command crucial ?
In the SP scenario
  some sessions are expected to be continuously running
Workaround
  schedule sessions one after the other

Can the stop-sessions command stop a subset of sessions ?

```
Number of Sessions MUST contain the number of send sessions started
 by the local side of the control connection that have not been
 previously terminated by a Stop-Sessions command (i.e., the Control-
 Client MUST account for each accepted Request-Session where Conf-
 Receiver was set; the Control-Server MUST account for each accepted
 Request-Session where Conf-Sender was set).  If the Stop-Sessions
 message does not account for exactly the send sessions controlled by
 that side, then it is to be considered invalid and the connection
 SHOULD be closed and any results obtained considered invalid.
```

Does `exactly` mean `all` ?
If so, then why discuss sessions not previously terminated ?

Wireshark dissector

Announcement

RAD has developed a Wireshark dissector for OWAMP

We are in the process of extending it to TWAMP as well

Features:

- handles both IPv4 and IPv6
- dissects both control and test protocols
- interprets all control protocol fields
- automatically configures UDP test ports
  based on control protocol messages (needn't use `Decode as` …)

We intend releasing to the community once fully validated