IEEE P802.1CS Link-local Registration Protocol Draft 1.6 introduction

Norman Finn
Huawei Technologies Co. Ltd
cs-finn-D1-6-introduction-0918-v02

Changes to Draft 1.6 from Draft 1.5

- The terms, “Native system”, “Proxy system”, “Slave system”, and “target port” were introduced and applied consistently throughout the document.

- Optional conformance requirements for proxy systems and slave systems were added to clause 5. Some items common to Native end and relay systems were grouped together into a section 5.2.

- Clause 6 was modified to use the Native, Proxy, and Slave systems terms and to clarify the use of LLDP.

Changes to Draft 1.6 from Draft 1.5

- In Clause 7, the “Portal Neighbors List” is now the “Potential Port List”, which makes its function much easier to understand. “ProcessLrpLldpTlvChanges” was split into two pieces, with the second called “ProcessPotentialPortChanges.” This latter half has been revised considerably, as the old method for connecting Portals to LRP-DT Instances did not work. In particular, IP addresses are handled differently.

- In Clause 7, the LRP-DT instance descriptions have been modified to differentiate better between active and passive TCP OPEN, but to take much less notice of link-local vs. non-link-local IP addresses.

- In Clause 8, pamReceiveHello has been modified, and pamResolveInstanceRepeat added, in order to create Portals on the fly when Hello LRPDUs are received on passive TCP connections.

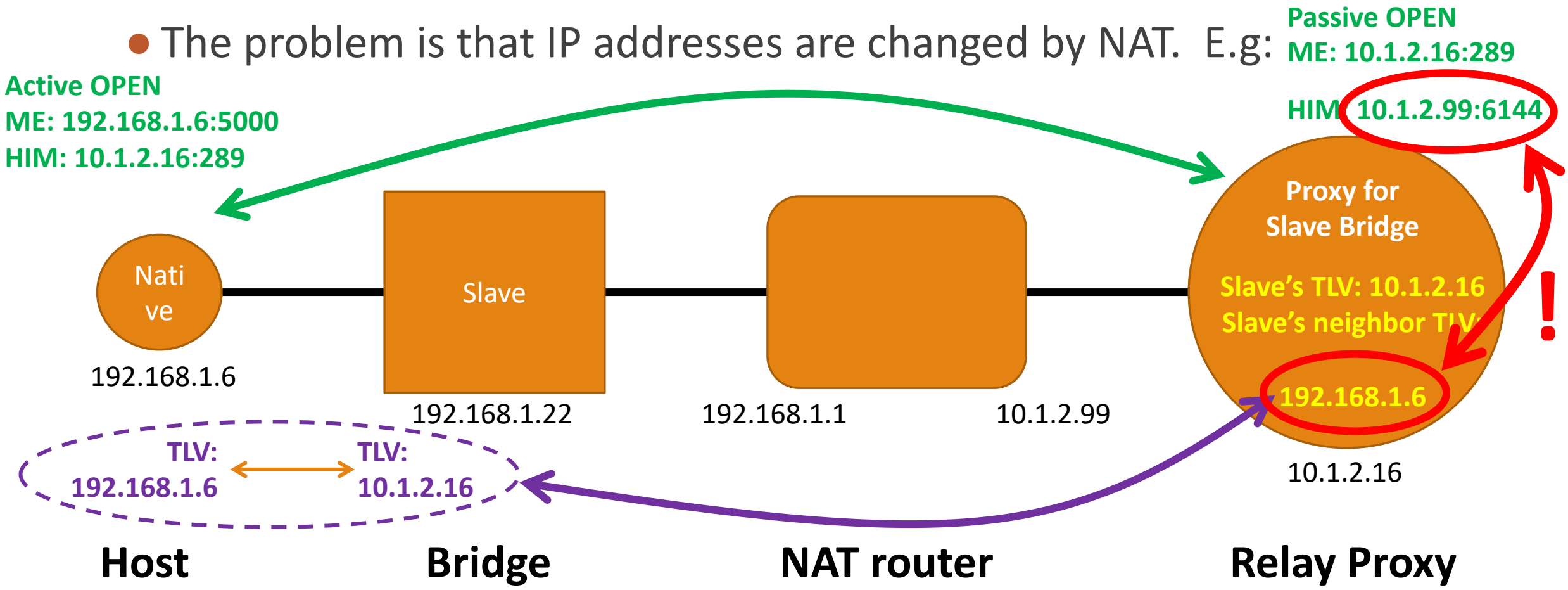
- In Clause 10, the Applicant and Registrar interfaces have been subsumed under the Portal interface.

Changes to Draft 1.6 from Draft 1.5

- The UML descriptions in Clause 11 have been improved.
- The names of many variables and managed objects have been changed for consistency and to make them more meaningful.
- The formerly empty sections 13.1 through 13.4 have been written. A PICS was added to clause A.
- See Annex Z for other issues.

D1.5 didn't work for assigning Portal state machines to TCP connections

- The problem is that IP addresses are changed by NAT. E.g:



Resolved:
Who does active / passive TCP OPEN?

- In the LLDP TLVs (or equivalent configured information), there is a field for “prefer **active** TCP OPEN” and “prefer **passive** TCP OPEN” and “**no preference**.”
- If the result is obvious, one makes active OPEN, and the other passive OPEN.
- If the result is not obvious, then **both** make **passive** OPEN and both make **active** OPEN.

Resolved (I think): Who does active / passive TCP OPEN?

- If both TCP connections succeed, Hellos are compared, and my/neighbor chassis/port IDs determine a “winning end”. Winner selects connection made actively, loser selects connection made passively.
- **ASSUMPTION MADE:** Although IP addresses can be altered between LLDP TLVs and making the TCP connection, the two systems do agree as to which system made the active OPEN and which made the PASSIVE open.
- **QUESTION**: Is this a valid assumption? If **not**, then we can:
  - Allow multiple connections, with (sometimes) one used in one direction, and one in the other direction.
  - Require the system administrator to not invalidate the assumption.

Why use preferences? Why not just the chassis/port tiebreaker?

- Because of NAT, it may be impossible for one system or the other to make an active TCP OPEN. It takes extra NAT configuration to allow a device on the “global” side of the NAT box to make a connection to a specific device on the “protected” side.
- The network administrator knows these things, and can set up the preferences so that the connections are made in the right direction. The chassis/port IDs can make the wrong choice.
- If preferences are specified, then only one connection is attempted. If not specified, then **both** try the connection, and one or the other can fail. If both succeed, we prune back.

Resolved
Which Hellos are sent on which connection?

- System making **active** TCP OPEN knows exactly what Portal pairs are associated with that connection, and can send all of those Hellos.
- System making **passive** TCP OPEN does not know for which Portals Hellos can be received on that connection. This system must wait to receive a Hello, match it to the potential portals list, then create that Portal's Hello state machines and pass the Hello to them.

To summarize closed (I think) TCP issues

- Configuration allows administrator to (for example) say that end stations prefer active, relay nodes prefer passive.
  - This can reduce the amount of TCP OPEN activity.
  - Mis-configuration of preferences cannot prevent TCP connections.
- Both ends **must agree** on who is active party and who is passive.
  - That makes it possible to eliminate one connection when two are made.
  - That makes sure that at least one Portal will send a Hello.

Are we near Working Group ballot?

- Enough of UML has been added to make decisions required to proceed with YANG model.
- The biggest unknowns are whether RAP can actually make use of the primitives, and whether some changes would be helpful to RAP.

Thank you