RAP/LRP Interaction using LRP Service Primitives

Feng Chen, Juergen Schmitt, Franz-Josef Goetz

IEEE 802.1 Interim Meeting
Sept. 2018, Oslo, Norway

Introduction

This presentation examines some of the LRP functions currently specified in P802.1CS-D1.5, in particular the LRP service primitives, and provides some thoughts on how to use them to specify the Resource Allocation Protocol (RAP) in the upcoming P802.1Qdd project.

1) Use of the LRP association primitives in RAP
2) Handling of Portal disconnection
3) Registrar database overflow
4) RAP proxy/slave systems

1) LRP Association Primitives for RAP

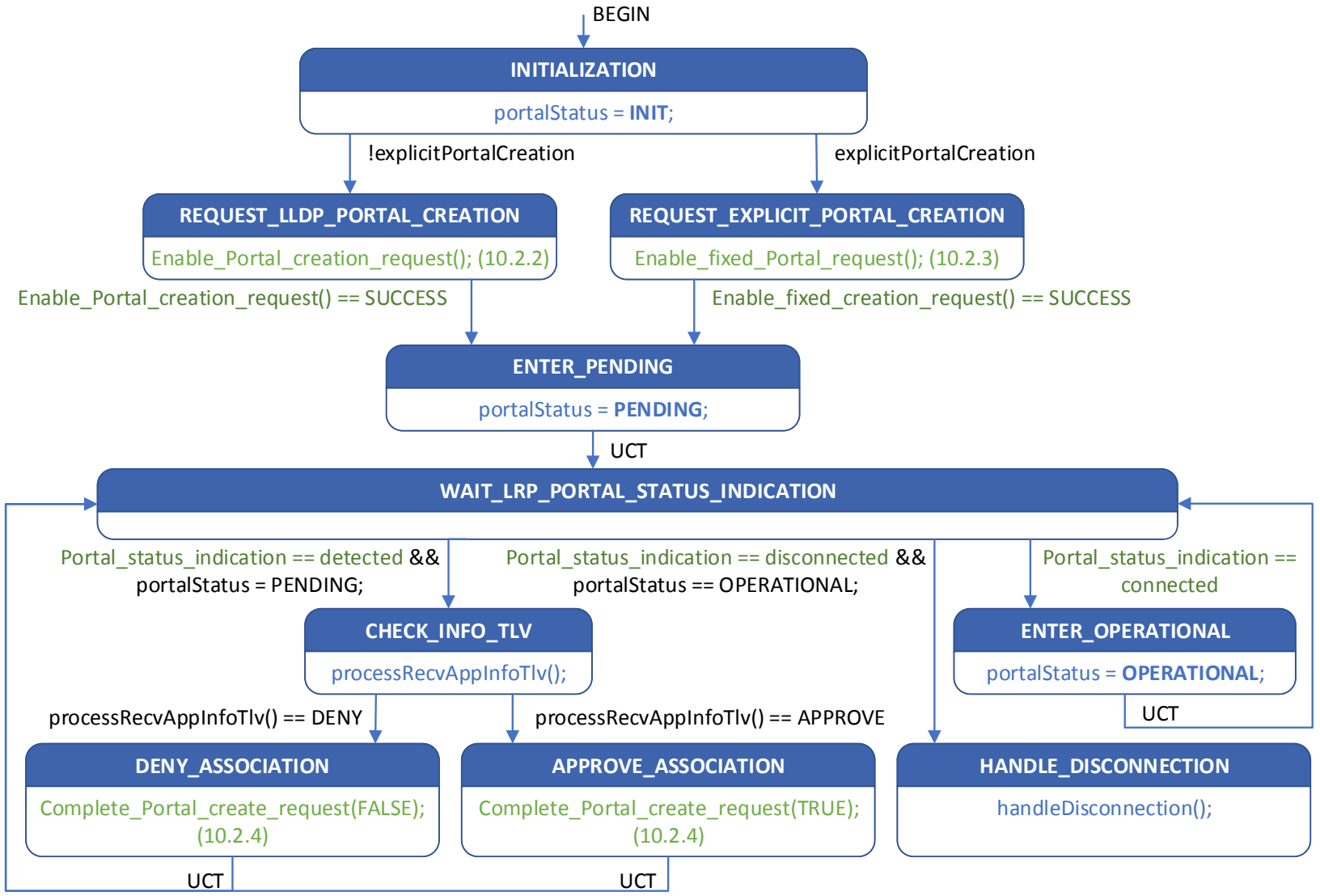
LRP D1.5 provides the following Association primitives for its applications to

- initiate the creation of LRP-DT instances and Portals in either of the following two ways:
  - Enable Portal creation request (10.2.2) – when using LLDP for discovery
  - Enable fixed Portal request (10.2.3) – when using static configuration
- approve or deny the portal handshake: Complete Portal create request (10.2.4)
- receive the portal association status from LRP: Portal status indication (10.2.5)

RAP will specify state machines (see the next slide) that interacts with LRP to handle portal association, which

- support choosing either of two portal creation methods via a managed object
- trigger routines when receiving portal status indication from LRP
  - process the Application Information TLV (9.2.2.10) received from the neighbor, when "detected"
  - enable use of applicant/registrar primitives to operate database, when "connected"
  - perform actions, when "disconnected"

Use of LRP Association Primitives in RAP State Machine



The state machine keeps track of the following three states:

- **INIT**: initial state
- **PENDING**: portal creation in progress, use of LRP database primitives disabled.
- **OPERATIONAL**: portal connected and operational, use of LRP database primitive enabled

The state machine calls the following routines in response to LRP portal status indication:

- **processRecvAppInfoTlv**: in the event of detected, check received App info TLV to decide whether approve the handshake
- **handleDisconnection**: perform actions in the event of disconnection (hello time-out), e.g. reset database or shut down the portal completely

portalId Assignment Issue

portalId on the LRP/Application service interface

- used as an identifier that associates the application with a specific Portal
- present in all service primitives, except the *Enable Portal creation request* and *Enable fixed Portal request* primitives, because portalId is unknown to the application when these primitives are issued by the application to initiate the portal creation.

Issues:

- The current LRP draft D1.5 does not specify how and when portalId will be allocated by LRP and notified to the application.
- On the port connected to shared medium, a single call of either of the enable Portal creation primitives can possibly result in creation of more than one portals for mutliple point-to-point associations. A more sophisticated scheme for assigning multiple portalIds and notifying them to the application is needed.

2) Primitives for Handling of Portal Disconnection

- LRP makes use of periodic exchange of Hello LRPDUs to ensure that the Portals are connected and operational. Once Hello exchange has timed out or otherwise failed, a Portal status indication with the code „disconnected“ is issued by LRP to the application.

- RAP conducts soft-state reservation and relies on the LRP Hello mechanism to keep alive. When the “disconnected” status on a portal is reported by LRP, RAP may take one of the following actions:
  - **Reset database** by using the *Write record request* (10.3.1) primitive (with zero data size) to delete records on the applicant side and using the *Delete record request* (10.4.1) primitive to delete records on the registrar side, and meanwhile leaving LRP to continue running the hello state machines, which will keep trying to reconnect to the neighbor.

  - **Shut down** the Portal completely including deletion of all state machines and data associated with that Portal.
    - **Proposal**: define a dedicated primitive for use by the application to request LRP to shut down an existing Portal indexed by the portalId

3) Registrar Database Overflow

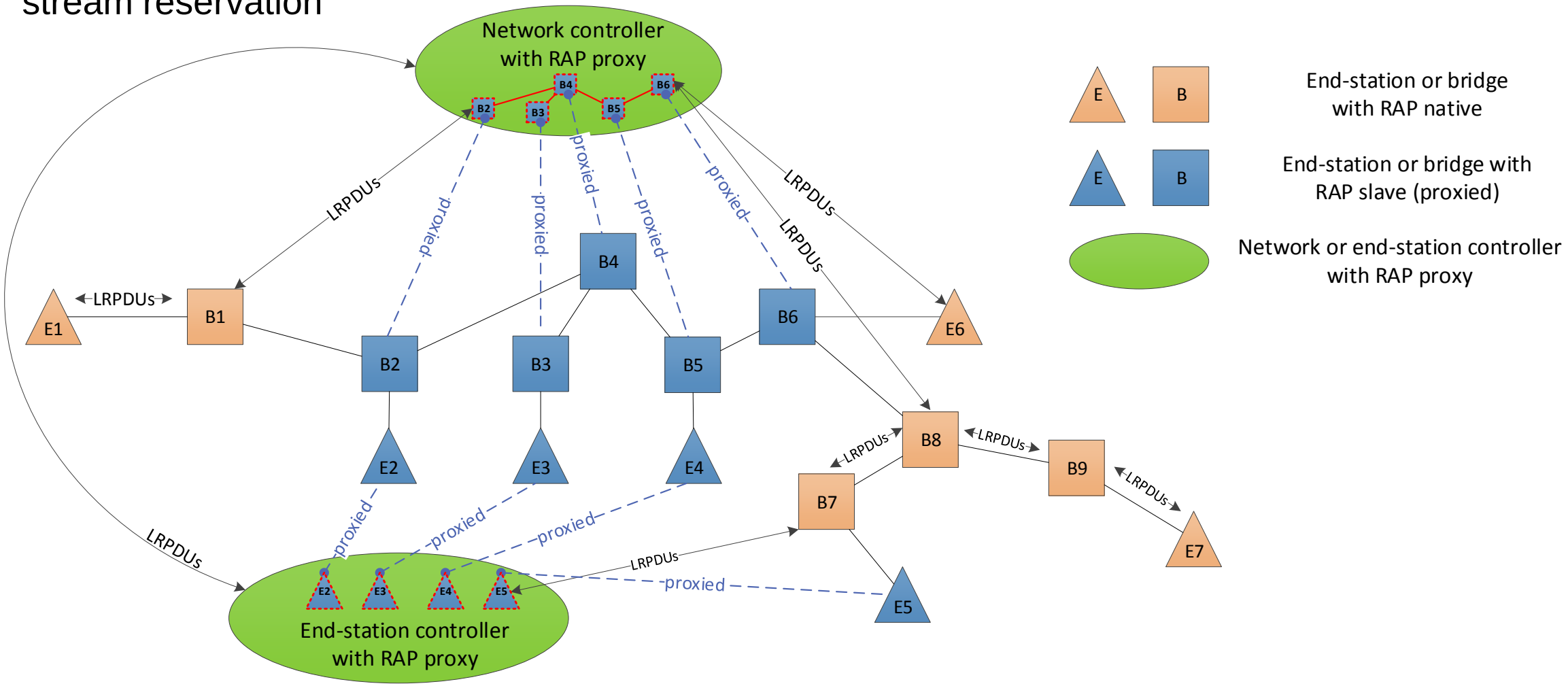
- The *Database overflow request* primitive (10.4.2) in the current LRP draft D1.5 provides the ability for the application above the registrar to report an overflow status in the registrar database, which triggers LRP to set the database overflow bit in the Hello LRPDUs that are transmitted to the applicant side.

- It would be useful for the application if LRP adds support for a more precise overflow reporting and handling mechanism on a per-record granularity, which means
  - The registar knows which record causes overflow and sends this information back to the applicant
  - The applicant reports the overflowed records to the application above it, with which the application can take measures accordingly, e.g. instruct LRP to delete the records that cause overflow.

- **Proposals**:
  - add an overflow flag as a return value of the existing *Record written indication* primitive (10.4.3)
  - define an extra overflow bit in the record header, which will be carried in partial and complete list.
  - define a new per-record **database overflow indication** primitive for use by the applicant

4) LRP Proxy/Slave Systems for RAP

- RAP could leavage the LRP proxy/slave systems to realize network function virtualization for stream reservation



Thank You!

Discussion