Reconciling requirements for the Resource Allocation Protocol (RAP), the Link-local Registration Protocol (LRP), the Multiple Resource Registration Protocol (MSRP), and Enhancements to MSRP (IEEE Std 802.1Qcc), the Centralized User Configuration (CUC), and the Centralized Network Configuration (CNC)

Norman Finn
Huawei Technologies Co. Ltd
dd-finn-RAP-LRP-MSRP-Qcc-0918-v01

Preface

We had some contention in Oslo over what is needed from RAP/LRP and from a CNC. This led to arguments of the form:

- You're trying to expand the scope of RAP/LRP unreasonably!
- You're trying to cripple the CNC!

I believe that these differences are not due to bad intentions, but to different assumptions made by the contending parties. There has also been some confusion about requirements, for which this author bears some responsibility.

Preface

The following presentation is an attempt to reconcile our differing assumptions, so that we can proceed with a clear plan for RAP and at least one other project. Outline:

- Evolution from MSRP to RAP/LRP
- How many controllers are there?
- How many UNIs are there?
- Summary

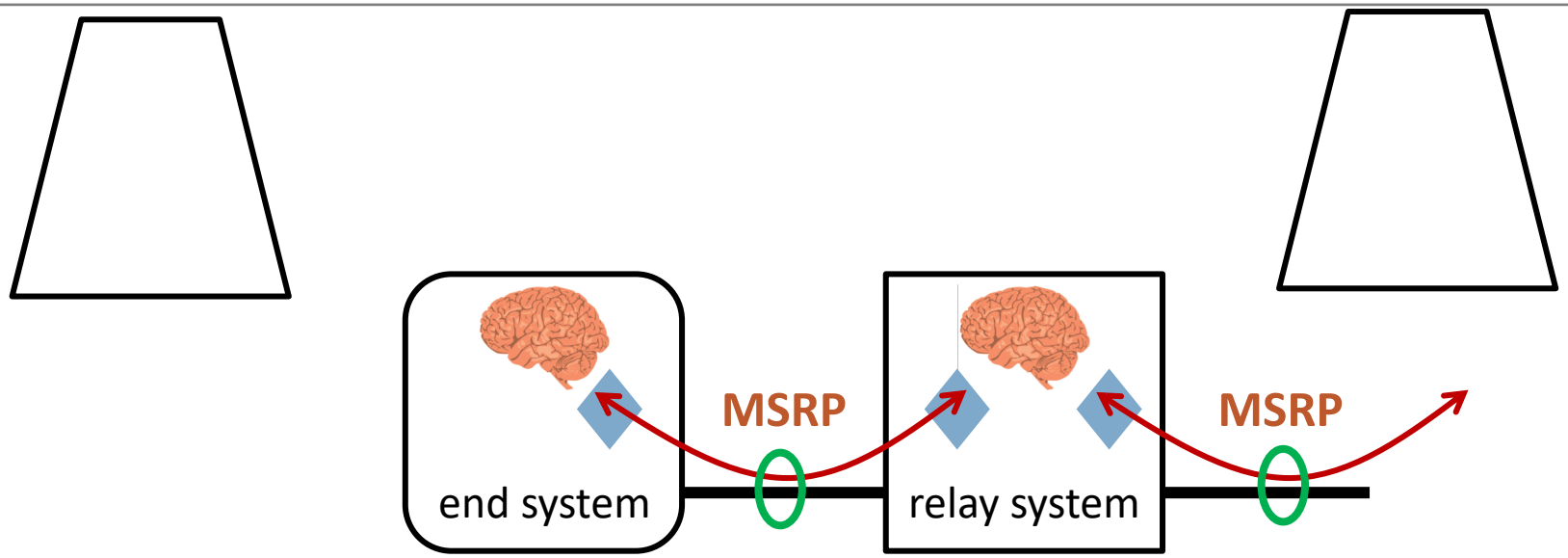
Evolution from MSRP to RAP/LRP

How many controllers are there?

How many UNIs are there?

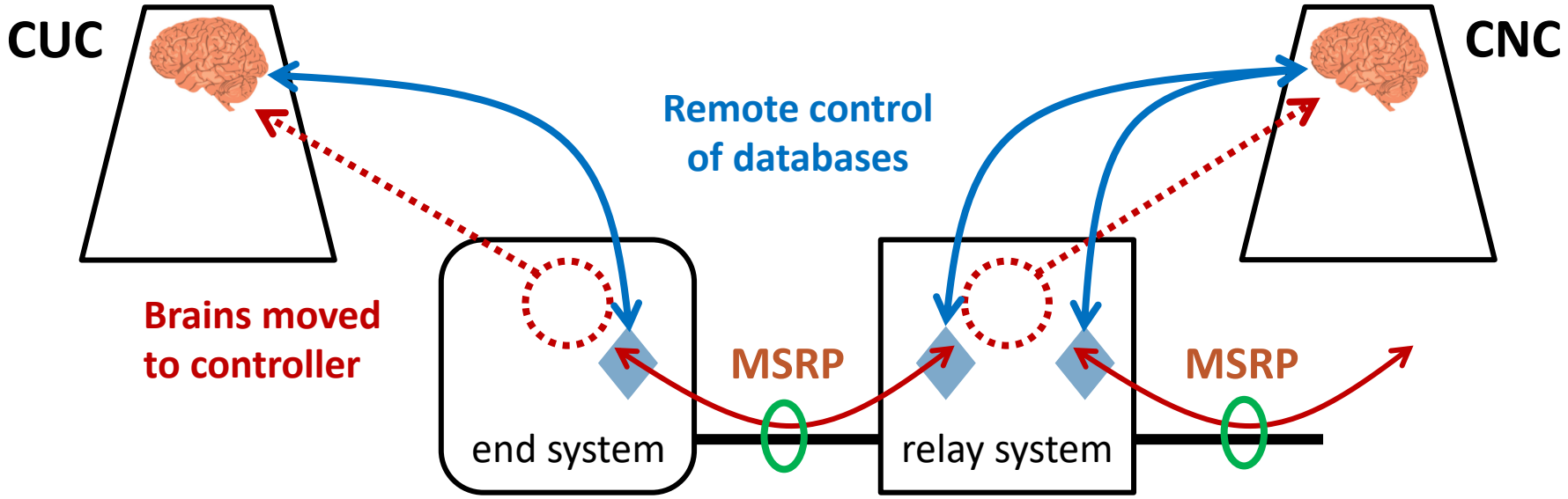
Summary

Step 1: MSRP



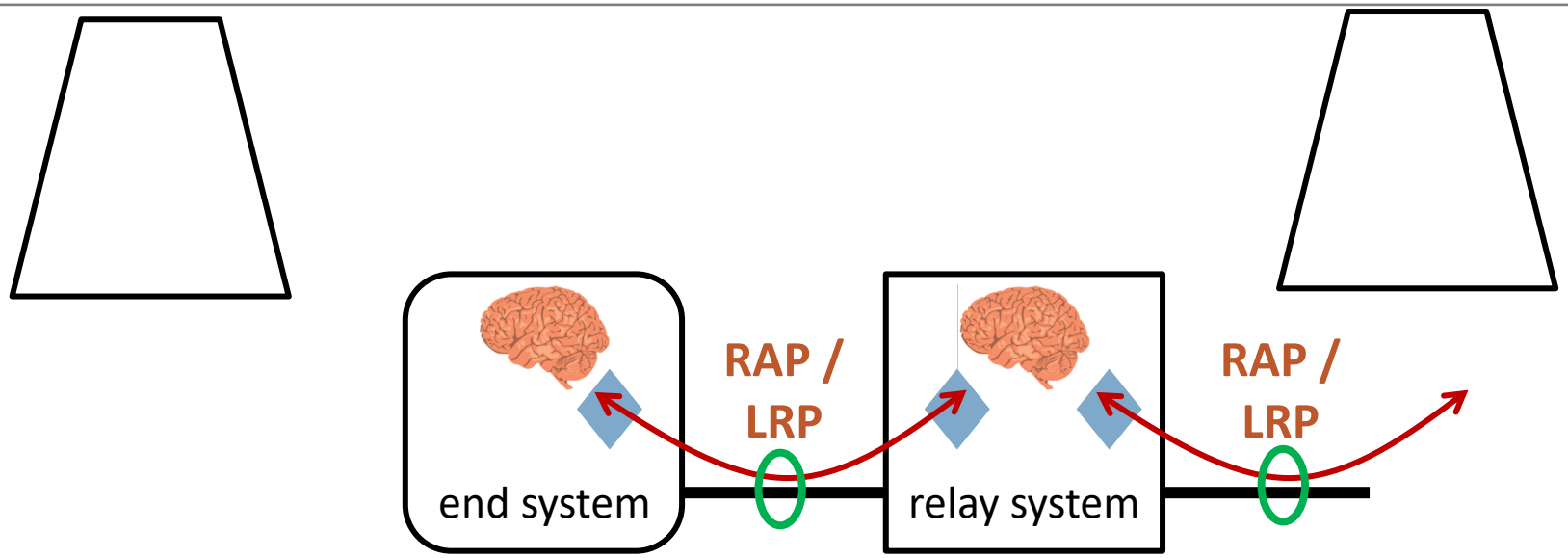
- MSRP information follows the data path.
- Every MSRP attribute is tied to one particular target link.

Step 2: MSRP + .1Qcc



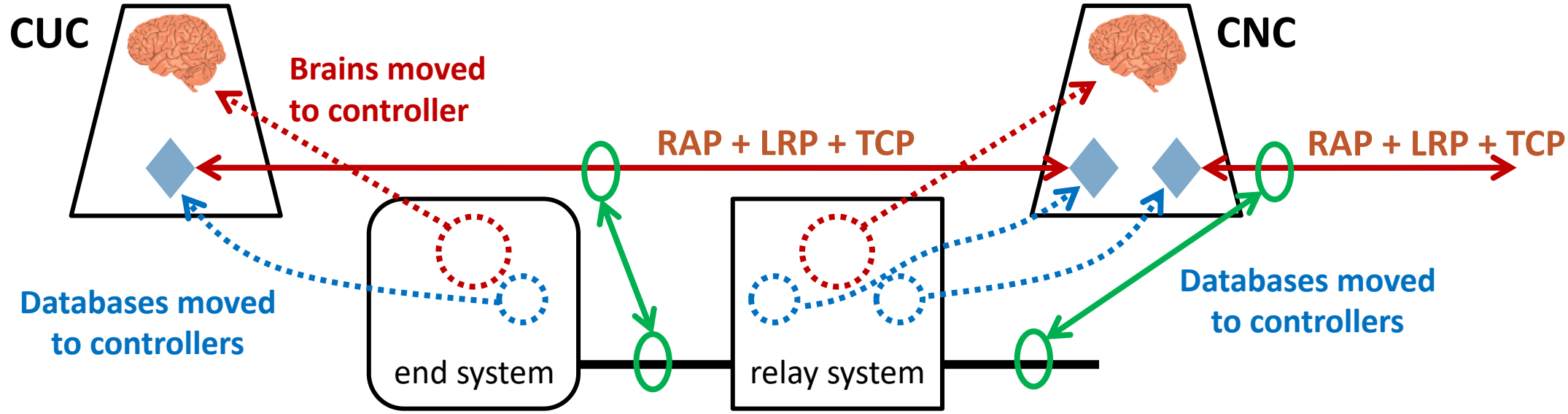
- MSRP **STILL** information follows the data path.
- Every MSRP attribute is **STILL** tied to one particular target link.

Step 3: RAP + LRP Native



- RAP information **STILL** follows the data path.
- Every RAP attribute is **STILL** tied to one particular target link.

Step 4: RAP + LRP + Proxy/Slave



- RAP information **no longer** follows the data path.
- But, every RAP attribute is **STILL** tied to one particular target link.

Constants from MSRP to Proxy RAP/LRP

Every attribute is in an applicant and/or registrar database.

Each database is locked to a target port.

MSRP locking: MSRP passes through the target ports.

RAP/LRP locking: LLDP chassis/port ID and My Portal Number are in the Hello LRPDU, then My Portal Number is in every LRPDU.

Evolution from MSRP to RAP/LRP

How many controllers are there?

How many UNIs are there?

Summary

Talker requests vs. Third-party requests

Talker request: I am “A”. I want to send to destination address “B”.

- By definition, a Talker request is from a TSN participant.
- It can come from a CUC, but from the CUC-as-Talker-Proxy.
- A Talker request is tied to a target port. It is the first hop of a (potentially) peer-to-peer protocol.

Third-party request: Source “A” wants to send to destination “B”.

- A Third-party request is, by definition, from a CUC.
- It may control only a small part of the network, but it is a CUC.
- A third-party request is not tied to a target port.

MSRP and third-party requests

Imagine giving peer-to-peer MSRP a third-party request.

- MSRP does not accept requests except from AVB/TSN-capable devices. A CUC need note be an AVB/TSN-capable device.
- How would a bridge receiving the request know where to find the Talker, the first target port, and the edge bridge serving that Talker? (I'm not saying it's impossible – but it's far beyond the scope of the current MSRP.)
- When the reservation is complete, how would the approval get to the original requester?
- Would the CUC have to have L2 connectivity? Why?

Not caring

Two of the goals of LRP/RAP:

- The Talker does not know or care whether it is making a request to a Bridge or a CNC/Proxy.
- The Bridge does not know or care whether it is receiving a request from a Talker or a CUC/Proxy.

But, **this only works for Talker requests**, not third-party requests.

CUCs make third-party requests. A CUC knows it's a CUC. A non-CNC Bridge can't handle a third-party request. A system that can handle a third-party request knows it is a full-service CNC.

Two kinds of CNC, two kinds of CUC

A CNC can be a Proxy Bridge, and handle only Talker requests

A CNC can be a full-service CNC, and handle third-party requests

A CUC can be a Proxy Talker, and make only Talker requests.

A CUC can be a full-service CUC, and make third-party requests.

If one issues third-party requests, then one is a full-service CUC, and one knows it is talking to a full-service CNC.

Evolution from MSRP to RAP/LRP

How many controllers are there?

How many UNIs are there?

Summary

How many kinds of UNIs?

From the above arguments, there are clearly **two UNIs**:

- A **Talker UNI** is used for Talker requests.
  - At one end of the Talker UNI is a Native Talker or a Proxy Talker CUC.
  - At the other end of the Talker UNI is a Native Bridge or a Proxy Bridge CNC.
  - No request is defined for the Talker UNI that cannot be handled by a peer-to-peer implementation using the ruled defined in MSRP/RAP/LRP. (If this were not true, then the requestor **does** care what it's talking to.)
- A **Third-party UNI** is used for Third-party requests.
  - At one end of the Third-party UNI is a full-service CUC.
  - At the other end of the Third-party UNI is a full-service CNC.
  - Any request we can think of in the future could be defined for third Third-party UNI.

Evolution from MSRP to RAP/LRP
How many controllers are there?
How many UNIs are there?
Summary

Two kinds of CNC, two kinds of CUC

A CUC or CNC can implement one UNI + function or both.

The fact that most of the information elements (TLVs) are common between the two UNIs confused most of us (certainly me) into thinking that we were talking about only one UNI.

Summary

- We limit RAP/LRP capabilities to things that can be done with a peer-to-peer implementation.
- We start a new project for the Full-Service CNC + Third-party UNI.

With the suggested distinction between Proxy and Full-service CUC and CNC, the implementors, operators, and system designers all have a common set of expectations about cost vs. capability.

Thank you