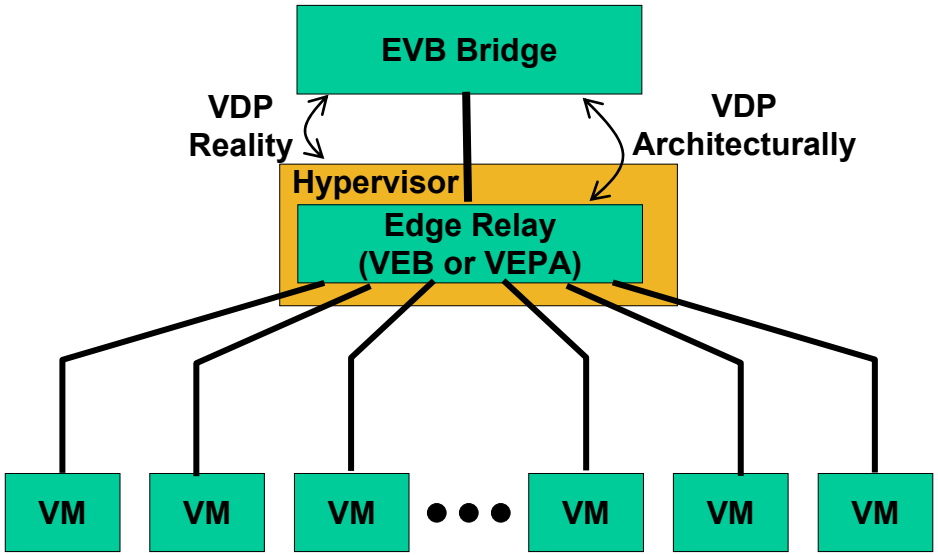
Aggregating VDP associations for scalability

Joe Pelissier

br-pelissier-VDPscalability-0911.pdf

VDP Termination in an EVB environment



- ***Architecturally***, **VDP is a protocol that operates between the EVB Bridge and the Edge Relay (aka VEPA or Bridge)**
- ***In reality***, **the station side of VDP is run by the hypervisor**

  802.1 does not specify hypervisors, so we put it in the block we do specify, the ER in this case (which may or may not be part of the hypervisor)

  Either way, it's the hypervisor that has the knowledge of when a VM is going to migrate, etc.

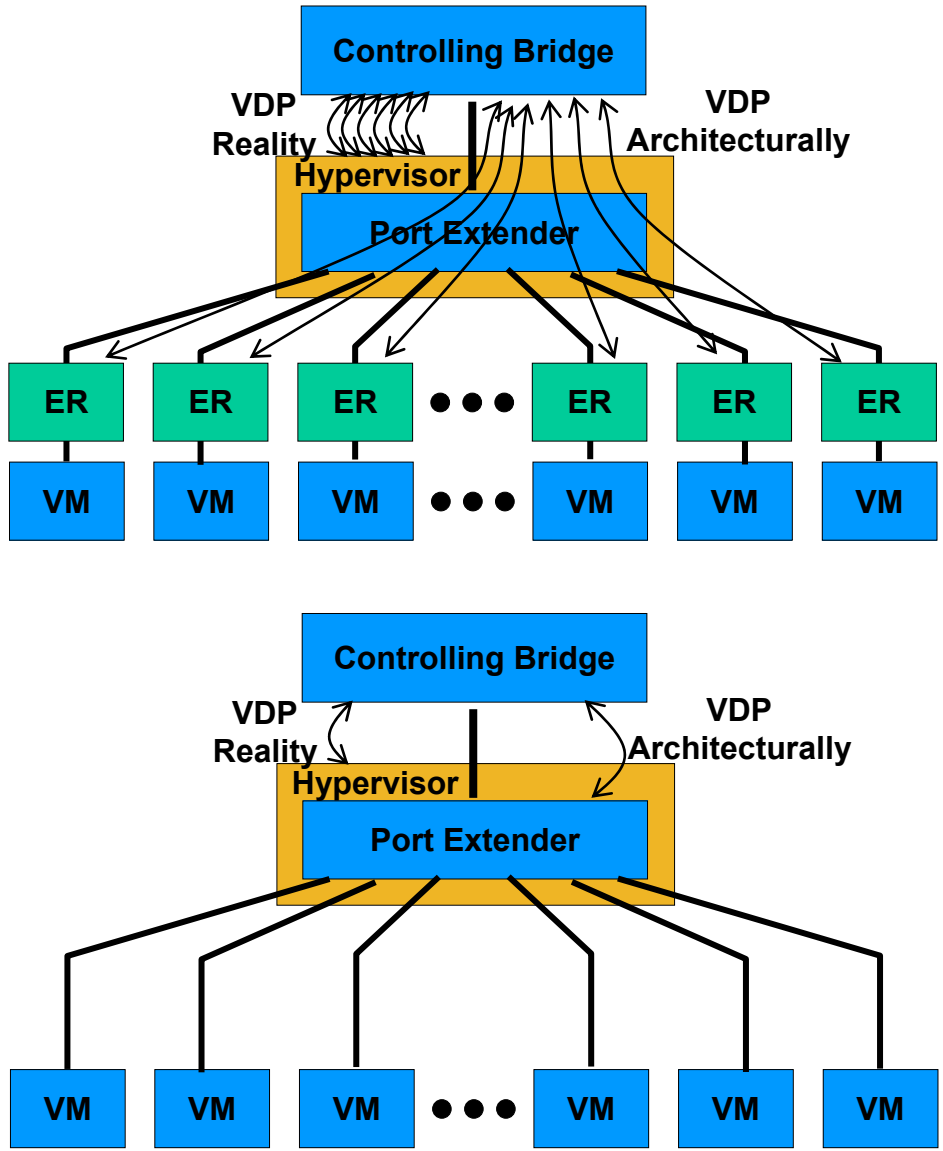
  One instance of ECP per ER

  One receive buffer per ER

  All VMs may be aggregated

VDP in an PE environment

- **Current Model**
  - ***Architecturally,* VDP is a protocol that executes between the Controlling Bridge and *each* ER**
  - **One ER per VM**
    - One instance of ECP per VM
    - One receive buffer per VM
    - No aggregation
  - ***Reality*, VDP runs between the hypervisor and the Controlling Bridge (same as EVB). Hypervisor performs backflips to emulate 1000's of ERs, ECP sessions, etc.**
- **Proposed model**
  - ***Architecturally*, VDP is a protocol that operates between the Controlling Bridge and the Port Extender**
  - ***In reality*, the station side of VDP is run by the hypervisor**
    - One instance of ECP per PE
    - One receive buffer per PE
    - All VMs may be aggregated



Observations

- **VDP is a control protocol between the hypervisor and the EVB-B / Controlling Bridge**
  - **Enables a bridge to configure itself for each VM**
  - **Allows the bridge to identify the traffic to/from a VM**
- **This is the purpose in both an EVB and PE environment**
- **There is no need for the protocol to operate over the data channels**
  - **In many cases, it does not even make sense**
  - **For example, at a pre-associate stage there is no VM, no VSI, and no need for a channel**

Some History

- **VDP was intended from the beginning to be used in both PE and EVB environments**
- **From the earliest designs, the need to aggregate multiple associations in a single ULPDU was recognized:**
  - **Many early discussions concerned the size of the association TLV, how many could be packed into a ULPDU, and how this compares to the number of VMs that might be supported**
  - **Care was taken to keep the association TLV reasonably small**
  - **When new filter info formats were suggested, concern was raised regarding their size and the reduction of the number that may be aggregated**
    - Ultimately, we decided that some new filter info formats were necessary
  - **A proposal was made to expand a few fields to GUID size**
    - Parts of this proposal were rejected for several reasons, including concern over increasing the size of the association TLV and thus the number of associations that may be aggregated in a single TLV
  - **A proposal was made to add a manager ID to the association TLV**
    - We decided to make this a separate TLV so that the information would not need to be repeated in each association TLV increasing the association TLV size
  - **The desire to efficiently aggregate these associations has never been questioned by anyone in these discussions**
    - Until now

Reality backs theory (at least in this case…)

- **Real data backs up the assumptions we made from the beginning**
- **Many VDP operations execute quite fast:**

  Associate and pre-associate with reservation if the port profile is cached and trivial

  Pre-associate without reservation if the port profile is cached

  De-associate

  Keep-alive

- **Real implementations show that the cycles required to process basic frame reception and transmission are comparable to the cycles required to perform the above operations**

  Without aggregation, frame reception and transmission consume approximately half of the processing cycles of the above operations

  Or, from a scaling perspective, aggregation provides an approximate 2x increase in the number of VMs that may be supported

  Your mileage may vary…

Buffering is another big issue…

- **ECP provides basic flow control**
  - **You cannot send a new frame until the last one has been acknowledged**
  - **Therefore, ECP requires a single receive buffer of about 1.5kB**
- **Using VEPA as an example:**
  - **A VEPA uses a single instance of ECP for all subordinate VMs**
    - Thus a VEPA requires about 1.5kB of receive buffer
    - The EVB-B requires about 1.5kB of receive buffer per VEPA
- **As proposed, PE is denied the ability to aggregate VDP from multiple VMs into a single ECP session**
- **Consider a base PE with 4k VM capacity**
  - **A separate instance of ECP is forced for each VM**
    - Thus the base PE requires approximately 6MB of buffering
    - The Controlling Bridge requires 10's or 100's of MBs of buffering

Summary

- The ability to aggregate multiple VDP associations in a single ULPDU was fundamental in its design from the beginning
  - Throughout its design, we took steps to preserve this capability
- This capability provides real performance enhancements and memory savings
- It was assumed, at least by some, that this was being done for the benefit of both EVB and PE
- br-pelissier-PEVDP-thoughts provides a simple approach to provide these benefits equally to both EVB & PE environments

Thank You!