Getting 200+ Mpps from a COTS server

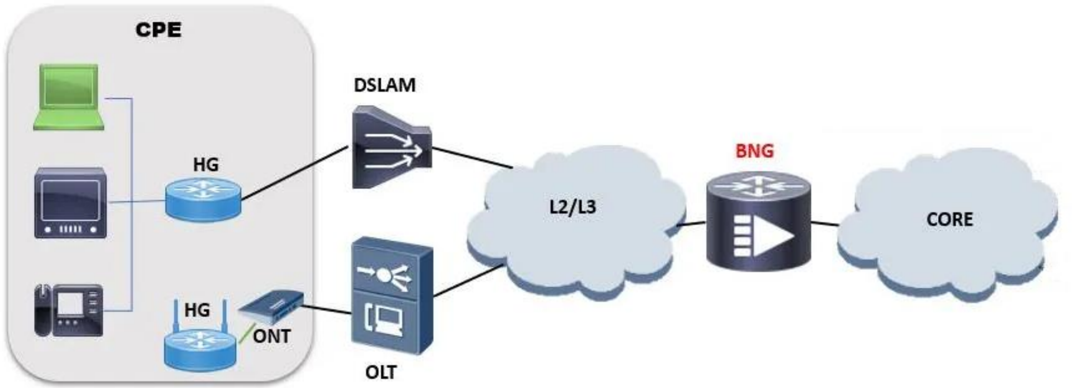
Hrvoje Habjanić
Architect & Co-Founder

About Author

- Graduated from ETF in 1996 (now known as FER)
- Worked 17 years in Hrvatski Telekom
- Last 7 years in 5x9 Networks
- 20+ years developer experience
- 10+ years network experience
- 10+ years of database administration
- Project lead for numerous projects

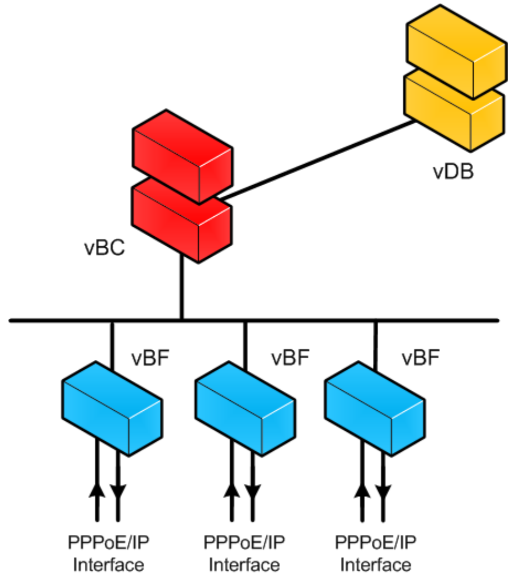
What is BNG

- Broadband Network Gateway
- Terminates PPPoE/IPoE user sessions
- Provides L3 connectivity to the Internet
- Enforces per user specific QoS profile and ACLs
- Responsible for AAA, LI, etc.



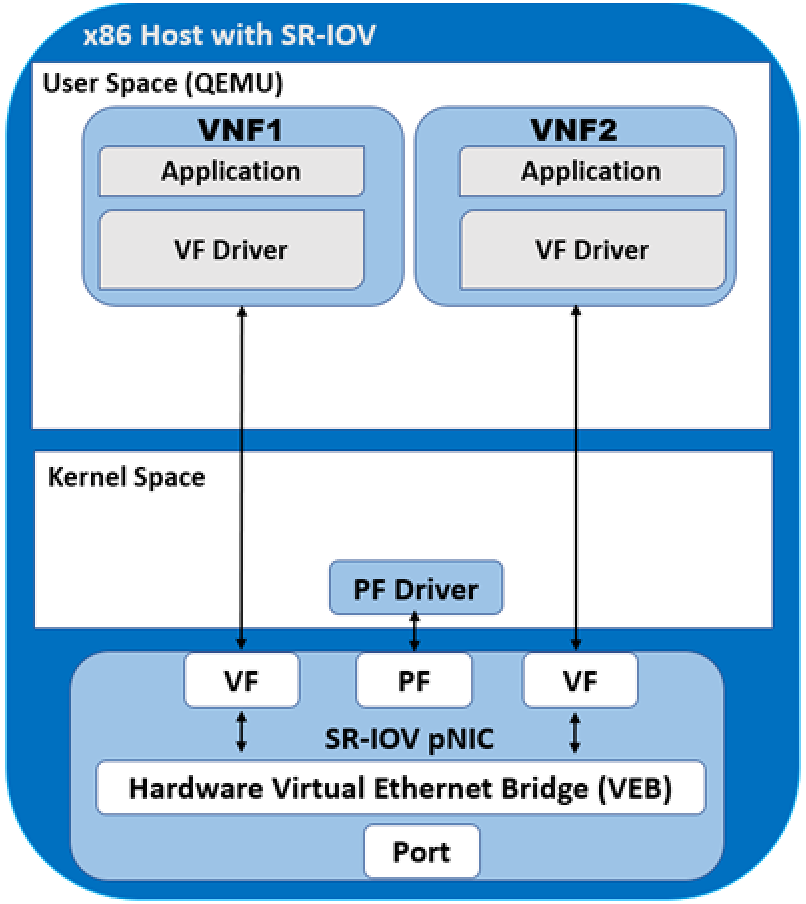
5x9 vBNG

- Virtualized CUPS system
- But, with full automation built-in
- Three main components:
  - Dashboard – configuration, automation, GUI, API
  - Controller – Radius, IP allocation, routing, …
  - Forwarder – PPPoE/IPoE termination, QoS, ACL, etc.
- Build for scalability and flexibility
- Relies on SR-IOV and Intel DPDK accelerations



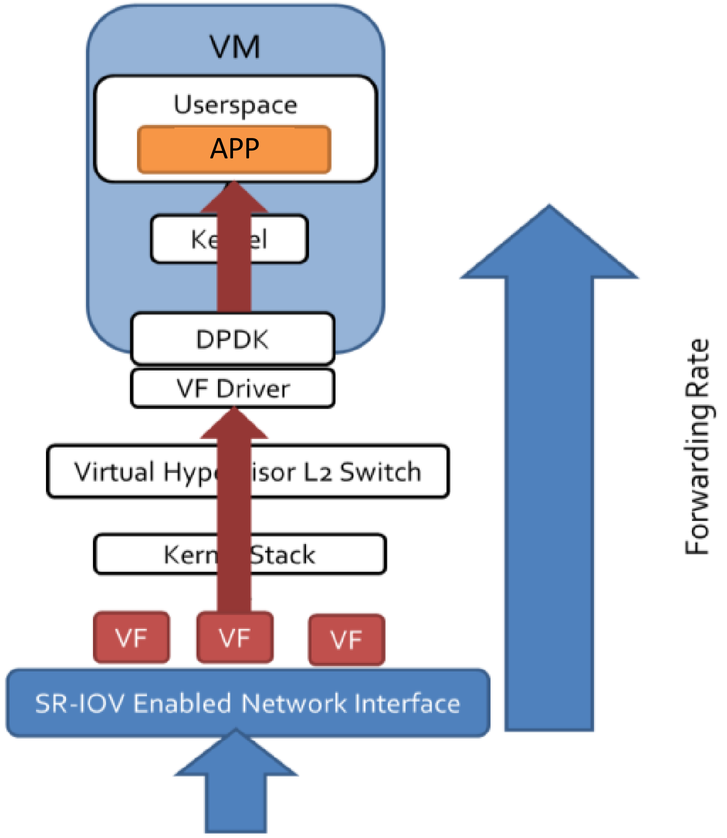
SR-IOV

- Single Root Input Output Virtualization
- Mechanism which present virtualized physical device directly to user-space application
- Mainly used to go around host kernel and its drivers
- Not supported by all hardware



Intel DPDK

- Data Plane Development Kit
- A set of libraries, used to directly access network device skipping guest kernel and its drivers
- Works in “pull” mode
  - CPU constantly checks network device
  - Causes 100% load on CPU
- Network card must be supported by DPDK



The “Small VM” Concept

- Initial concept was to make forwarder instance as small as possible
  - 4 vCPUs and 4 GB of memory per instance
  - Automated horizontal scaling approach used
- Pros
  - Optimal resource utilization
  - Each instance can have unique configuration
  - Easy to find virtual resources to start instance
- Cons
  - Large number of instances

Initial Performance

- 2x Intel Xeon Gold Gen2 CPU with 18 cores
- 16 Forwarders
- 40 Mpps no hQoS
  - 160Gbps for 500B packet size
- 26 Mpps using hQoS
  - 100Gbps for 500B packet size

- Intel reference document claims 100 Mpps per CPU
  - Newer hardware, simple in-out packet forwarding applications
  - But, still significant performance difference

Quick Performance Wins

- Move to newer hardware
  - Optimize application for new HW
  - Initial 30% performance gain
- In-depth Intel DPDK tuning
  - Significant time spent to get deep understanding of underlaying hardware
    - CPU options, PCIe communication, BIOS options, network cards tweaks
  - Additional 30% performance gain
- Required significant time to the change-testing cycle
  - Each change is tested to verify performance impact

Painful Deep Dive Into the Code

- Intel vTune application
  - Used for runtime in-depth code analysis using CPU internals
  - Provides detailed performance insights for the code
- Executed many code analysis using Intel vTune application
  - Unfortunately, monitoring performance influences the performance itself
  - Identified “hot spots” in code (significant CPU wait)
  - Rewrite code to avoid waits (reorder instructions)
  - Rewrite loops to use multiple execution units
  - Reduce code – force compiler optimizations
- Additional 20% performance gain

Moment of Revelation

- In the end, main cause was CPU cache misses and waits
- High count of memory object used a lot of CPU cache memory
  - CPU ended constantly pulling and pushing data from main memory
- Memory data changes caused cache stalls
  - Every change needs to be flushed back to main memory
  - Memory page (block) is 4k bytes!
- Cache alignment was not optimal
  - CPU reads a “cache line” (in most cases 64 bytes)
  - Single data object must be fitted in cache line

Post Revelation Actions

- Major code rewrite
  - Separate read and write data sections
  - Separate frequently and occasionally used data structures
- Identification of additional PCIe bus bottleneck
  - Addressed by grouping and carefully scheduling PCIe transactions (DMA)
- Introduction of more advanced hashing algorithms
  - Significantly reduced lookup cycles (to just 1)
  - Significantly reduced in-memory routing table footprint (90% reduction)
- Recognition of “Small VM” concept as inefficient
  - The same information is stored multiple times in the cache
    - Routing tables, interface tables, etc

The “Big VM” Concept

- Combine multiple small VMs to avoid in cache data multiplication
  - Single VM utilizes the entire NUMA resources and all available CPU
  - QoS mechanism and route lookup improvements
  - At last targeted 200 Mpps!

- Now it became clear why other VNF vendors are using similar design 😉

- Rely on built-in full automation for deep network distribution
  - Appliance (Central Office)
  - One VM (Edge Cloud)
  - OLT integrated (Access Node)

Lessons Learned

- Always use latest hardware
  - Intel Xeon Gold Gen4 CPU with 32 cores each
  - Intel and Mellanox 100GE network cards
- Embrace the change, (re)use what is good
  - Switch to “Big VM” concept
  - Use “Small VM” concept where it fits best
    - Perfect for Edge Cloud deployment (One VM flavor)
- Continuous optimization is a must
  - What gets measured gets improved
  - Curiosity, learning, mindset and attitude are the key

Thanks for attention

Q&A