Migrating NFV Applications to KVM Guest

Mario Smarduch
Senior Virtualization Architect
Open Source Group
Samsung Research America (Silicon Valley)
m.smarduch@samsung.com

Background

- Bell Labs – 5ESS
  - RT- Kernel patching – 0 downtime
    - Patch, reclaim, atomic transfer vector switch
  - Enhanced CPU accounting – provide more reasons

- Motorola iDEN – push to talk wireless system
  - 1.3s for end-to-end group call – LMR – cross country
  - RT scheduling end-to-end latency tuning
  - Implemented Precise Process Accounting
    - Used by Sprint, Clearwire, …, - root cause deployment issues

- NFV Hypervisor run-time hardening (not Security Hardening)

- Spent time at customer sites -
  - On site in Seattle Public Safety outage, LA, SF deployment
  - Seen deployments blocked cost - millions $$$

What is NFV

- ETSI Standard
  - Virtualization of Network Functions previously deployed on hardware

- NFV Enables
  - COTS, Hardware Flexibility
    - HW is abstracted (for example QEMU - machine model)
  - Rapid Network Function innovation/implementation/deployment
    - Aka – Virtual Network Function
    - Innovation/implementation – VM a sandbox (image, qemu/kvmtool)
    - Deployment – cloud image server
  - Lower Operational cost and power usage
    - Cloud infrastructure and VM operations – i.e. migration, VM power off
  - Dynamic Network Function Chaining
    - Cloud infrastructure – orchestration, scaling
  - Standard VNF to HV and Cloud interface
    - Standardized VM Image & HV cloud mgmt interface

Advantages of full Virtualization for NFV

- Run mixed OS's
- Run mixed distros
  - no kernel configs & system tunable conflicts
- Own whole vertical stack – kernel & modules, user space
  - TEMs need some custom features in kernel –
    - For example TIPC, SAF HPI to emulated PV-IPMI
- Live migration, snapshot – get whole kernel state
- Debug – you own whole stack
- Backward compatibility – Old OS, New OS on older HW emulated
- No SPOF, quick restart on panic or HA
- Coarse grained resource isolation/security isolation
- No /proc conflicts
- Deliver whole VM – i.e. no worries about library versioning

Sometime back - CG-Linux

- Early CG-Linux not quite same but similar – new challenges
  - TEMs adapted quickly – Freedom from OS lock in

- CG – Linux deployment
  - Moving from RT proprietary OS's to Linux
  - Gaps – **expectation vs. implementation – NFV new challenges**
    - Timers non granular - coarse
    - Pre-emption – long periods non-preemptible
    - Poll/select – poor scaling with large fd set
    - POSIX Extensions – not compliant – no robustness, priority inversion,…
    - Memory Overcommit policies confusing
    - CPU – accounting – huge issue - sampled – unreliable results
      - If you can't measure you can't tune
    - IP pkts out of order

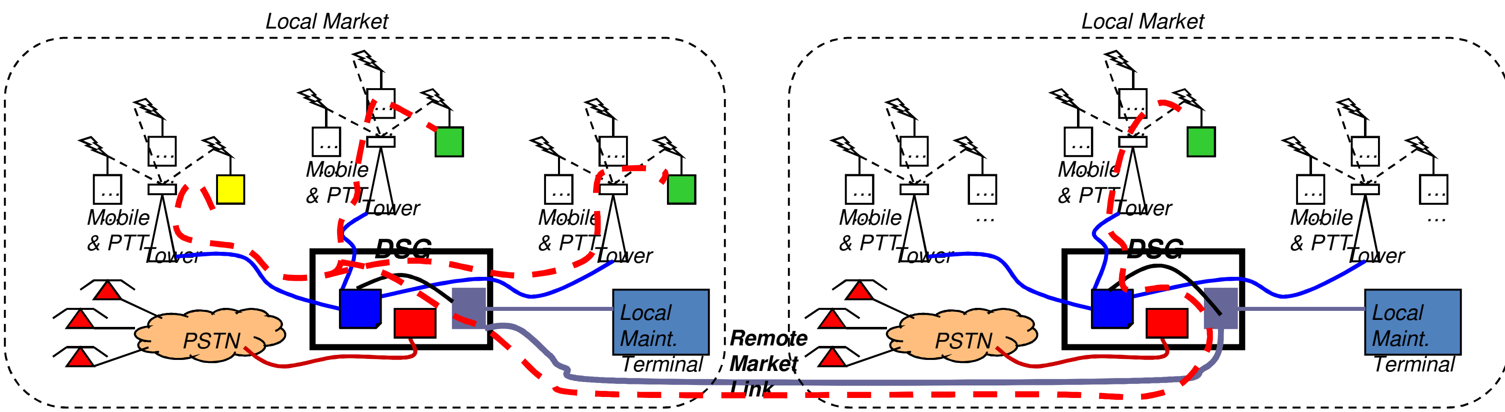
- Eventually Gaps resolved
  - Application adaption
  - Community – OSDL CGL played a big role
  - HW vendors

Wireless Networks and CG Environment

- Demanding Env – primarily measure - system up-time
  - Widely used metric 5 9's (.99999) availability
    - outages <= 30s don't count
    - <= ~5min 26sec downtime/year
  - But … real metric customer
    - 95% of user per cell satisifed
    - Data Plane - satisfied means 98%+ VoIP pkts arrive within 50mS
    - Latency perception – **key issue**
    - Call Processing – huge impact on capacity and QoS
      - ❑ CP – SAU/cell - ~400/5MHz – RRC_IDLE to CONNECTED ~100mS
        - RRC_CONNECTED – primarily Ue can issue UL sched req., get grants
  - Extreme emphasis on immediate **root cause**
    - IT Support - We're working on it or maybe – not acceptable
    - Carrier Support – immediate root cause …
      - ❑ Deployment moved back, huge financial penalites
    - Nothing invasive (PMU, Debug, …), no direct access allowed

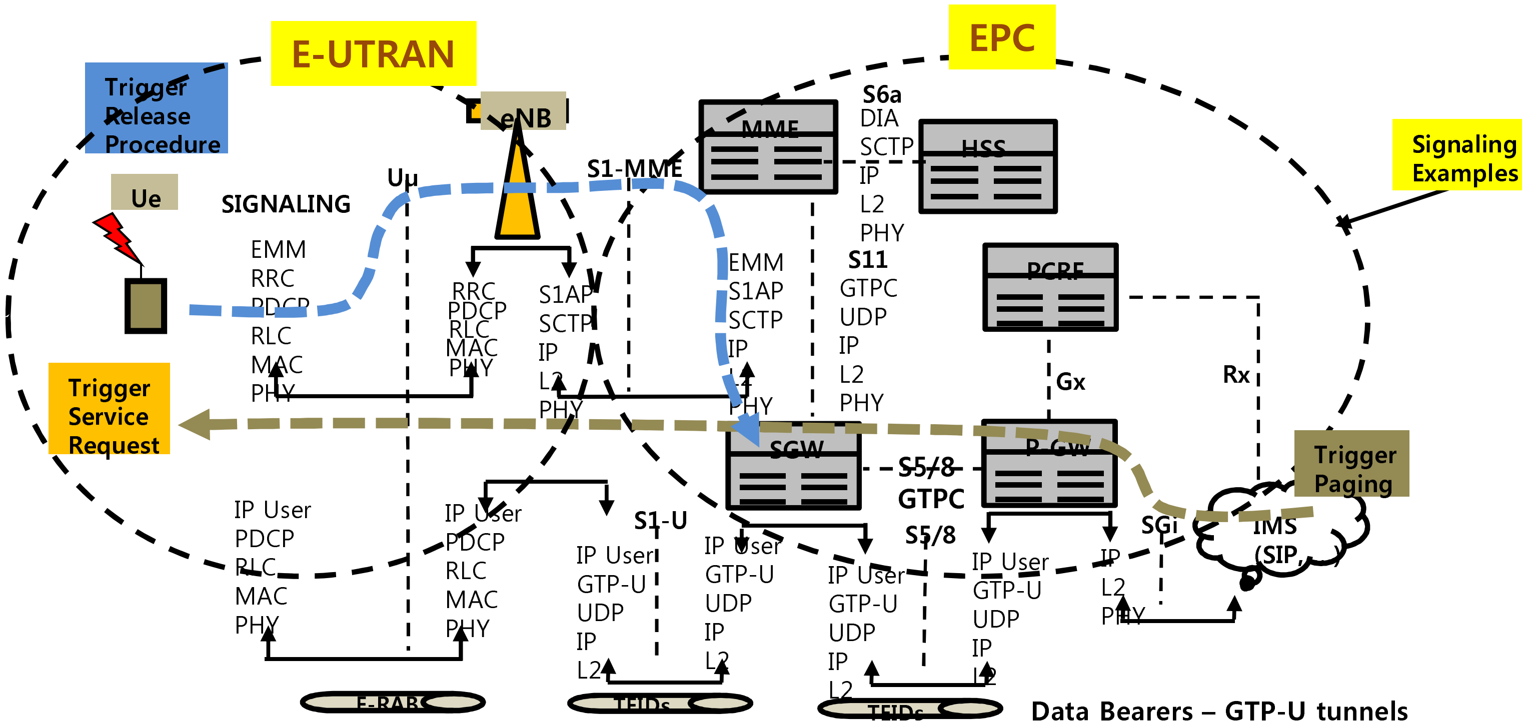
Wireless Environment

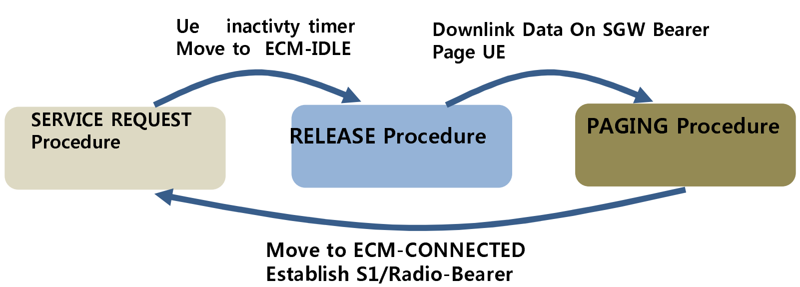
- Regardless 2G-4G – Control Plane, Data Plane, O&M
- Control Plane – state machine
  - Driven by – Event or Timer
  - Range - extreme LMRN - PLMN
  - LTE huge improvements Radio Access – increased capacity
    - OFDMA, SC-FDMA, MIMO – spatial multiplexing, rcv/tx diverslity, …



- Signaling Example
  - Public Safety – PTT Group call - yellow dispatches green – 1.3s to 'chirp'
  - Message – lookup HLR, page all mobiles, allocate bw, get confirm, …
  - real-time – but what does it mean here?
    - Deterministic execution – each NE bound latency/capacity
    - For CP – signaling deadline constraints

LTE High Level Architecture



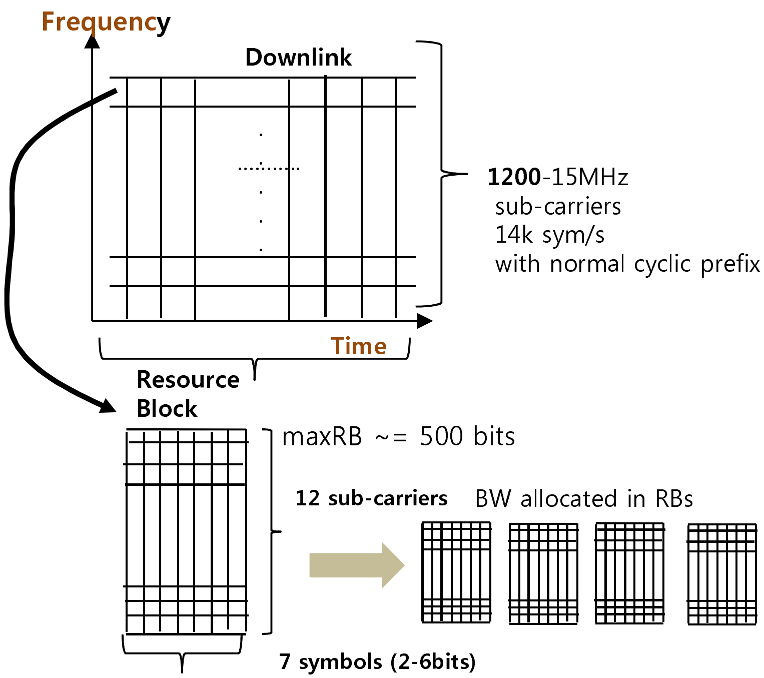


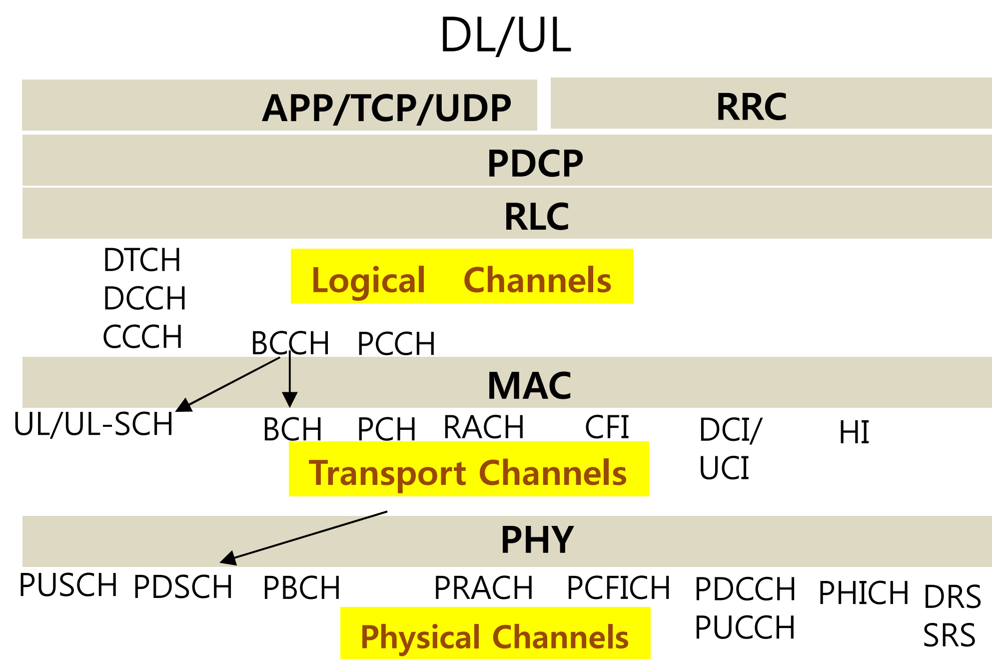
- Control Plane – limits capacity
- Attach, paging, MM – procedure determine
  - Number of UEs admitted, # of Bearers
- Signaling – state machine – again – determinism, timing

Basics of LTE Radio Access Side

➢ LTE DL Resource Grid

- eNB only or EPS procedures





➢ Some traffic eNB only

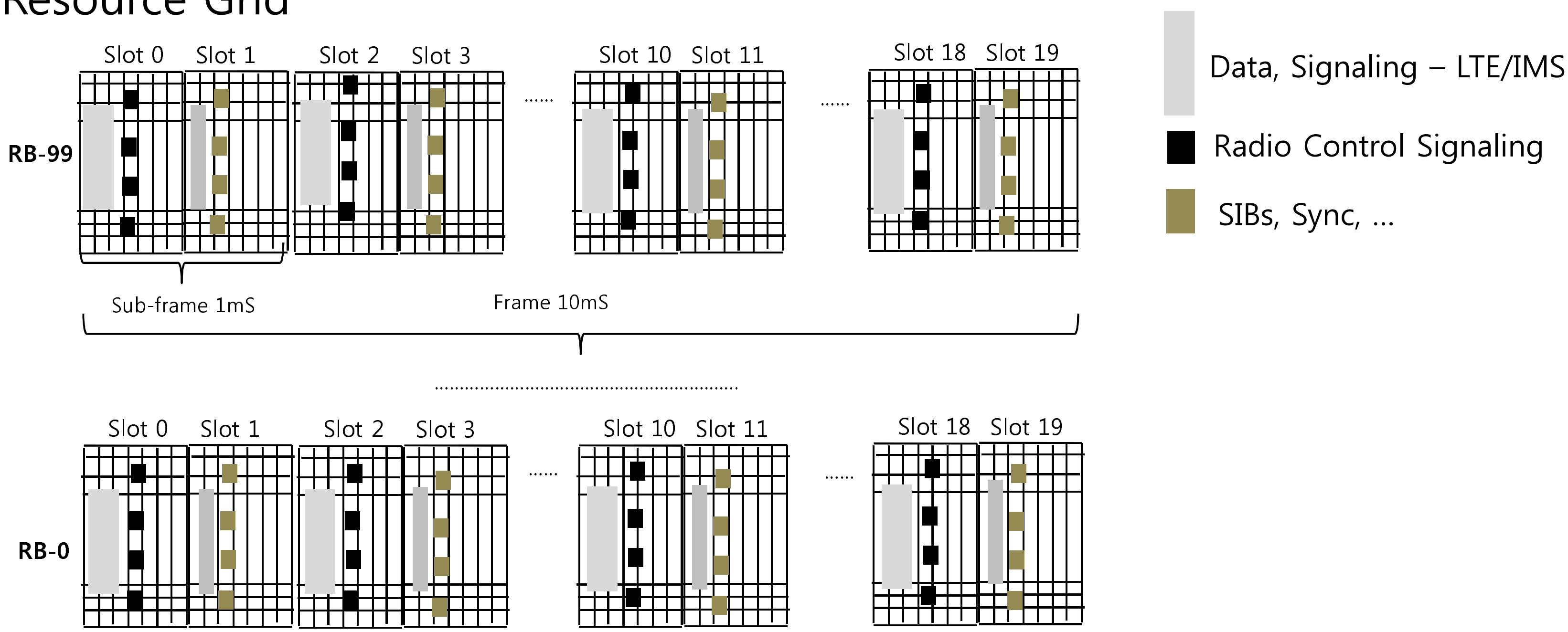
- DCI – DL sched cmd, UL sched grant, pwr/diversity ctrl
- UCI – ARQ ack, sched rqst; CFI – pfi – organization of data & ctrl info
- PRACH – random access – i.e.– attach, sr or tracking update procedures

➢ eNB radio & EPC processing

- Scheduling most complex – Fairness vs Throughput, power ctrl, fading, spreading
- Dimensioning – TA List, page rqsts, handovers

Basics of LTE Radio Access Side

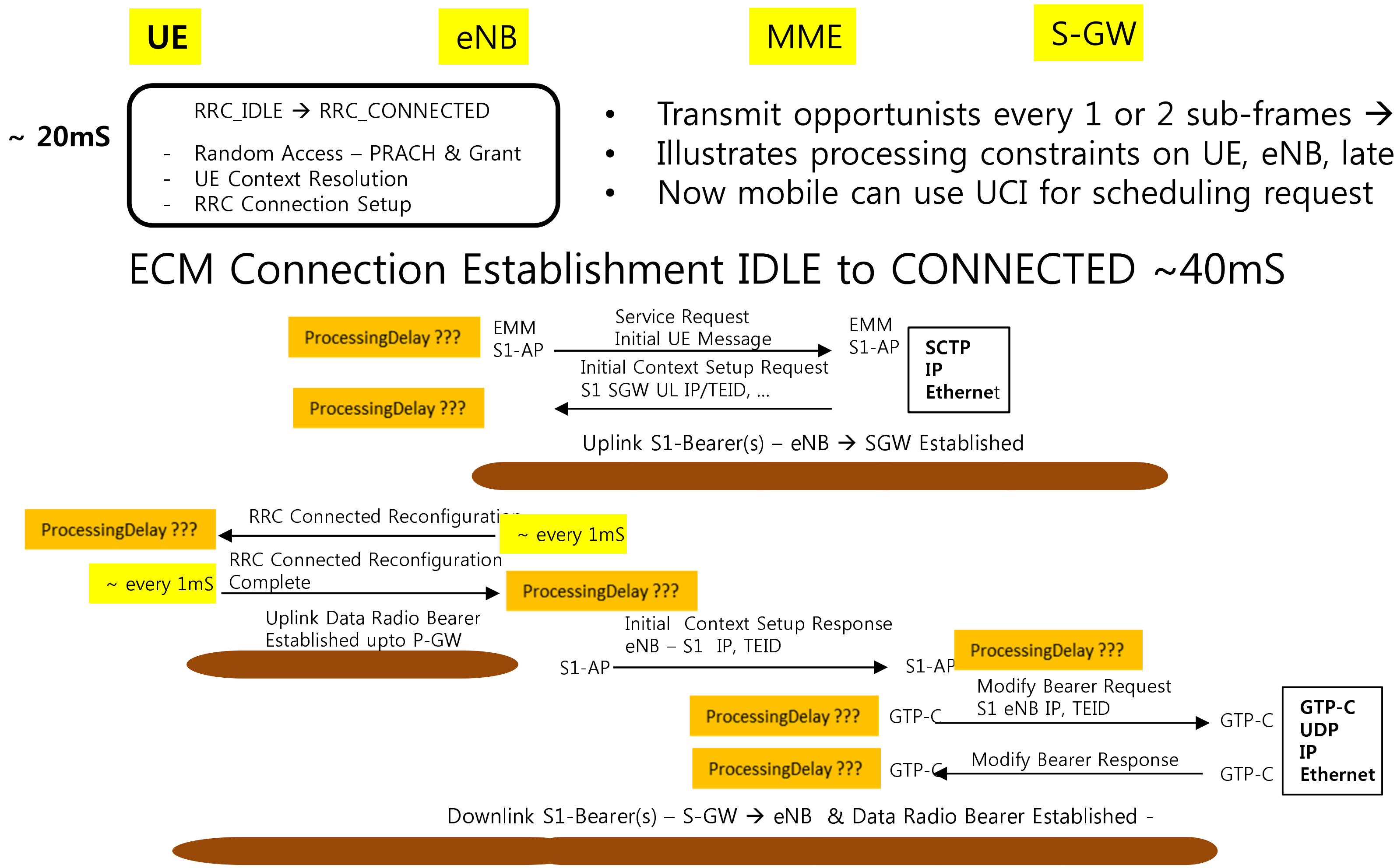
- Resource Grid



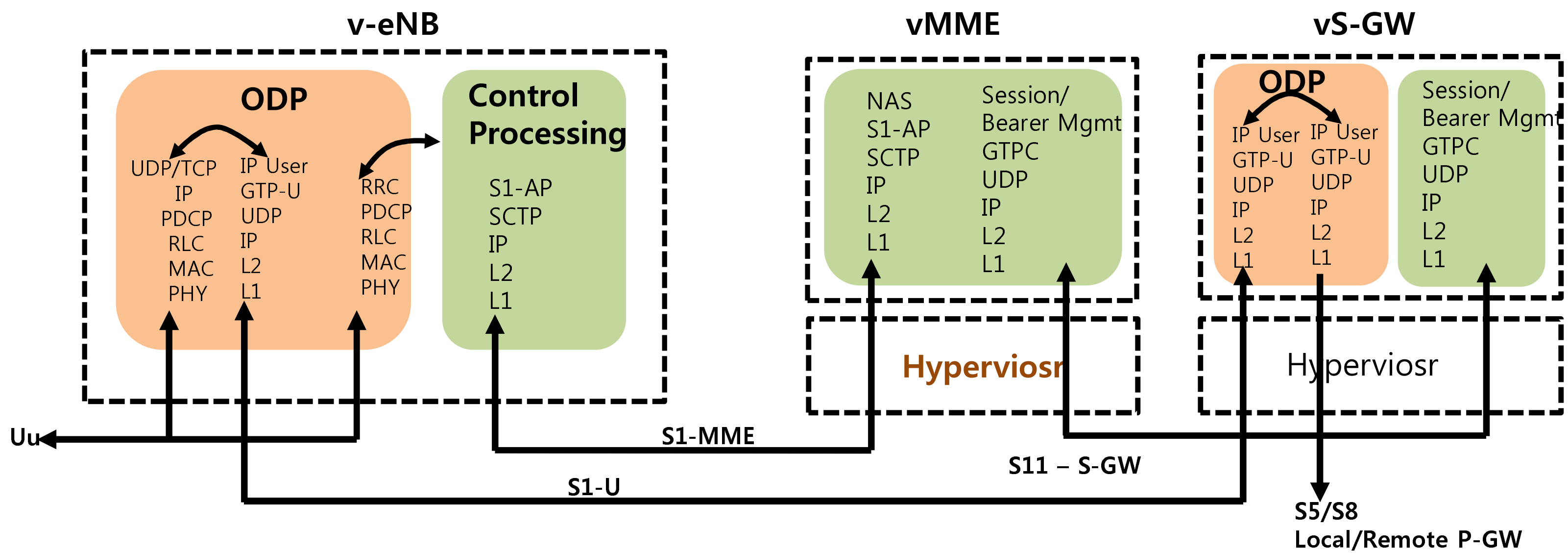
- RBs allocated to Control – like B-CH, PDCCH, PUCCH, PRACH, PHICH
  - Mostly Base Station (eNB) processing – RT, demanding
  - Signaling relevant only to eNB
- RBs for data – PDSCH/PUSCH
  - Not really though – LTE procedures - paging, attach, idle → connected
  - SIP-UA and IMS – transparent to LTE except QoS
- Allocation carrier & area specific – dimensioning engineers

Network Call Processing Example

- Service Request – mobile UL request – idle to connected
- RRC 100ms – in practice < 100ms for EPS Bearer setup



Example NFV Implementation



- Mixed Run-time environment –
  - Data Plane -
    - CPU dedicated, isolated from OS – i.e. no scheduler, interruts, timers
    - Hard real-time – tight loop – latency in few hundred uS's
    - Device passthrough
  - Or – control plane
    - RT deterministic – kernel scheduler/timers
    - Virtio
  - Management – VM – not RT intensive
  - eNB –
    - RRU – backhauled – multiple-technologies – GSM/UMTS/LTE
    - Vision – C-RAN
  - MME, S-GW – virtualized NEs

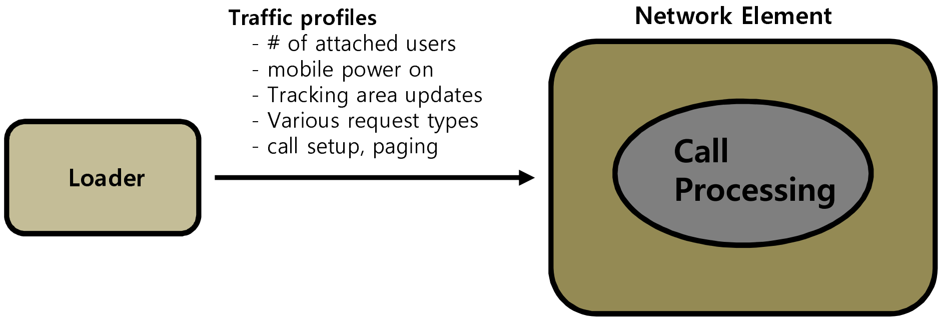
Take Away

- New Gaps – more challenging at System Level
  - Deterministic Execution
    - SR higher pri then UE Attach, …
    - DP co-exist with CP i.e. ODP w/ RT, TS apps
      - May decompose
  - Timers
    - LTE Ue timer appear friendly – Service Request 5s
    - But for MME pool - 100,000s, or millions of attached user
    - Rush hour or event – 10000s of signaling messages
  - Accounting – CPU – all starts here – time accrued to something
    - Need precise measurement – non-intrusive
    - Load shedding – relies on it
    - O&M, Root Cause analysis
  - Other Gaps – some highlighted later
  - Challenges – latency, performance, capacity
    - W/reasonable overhead

Lmbench and rt-tests test environment

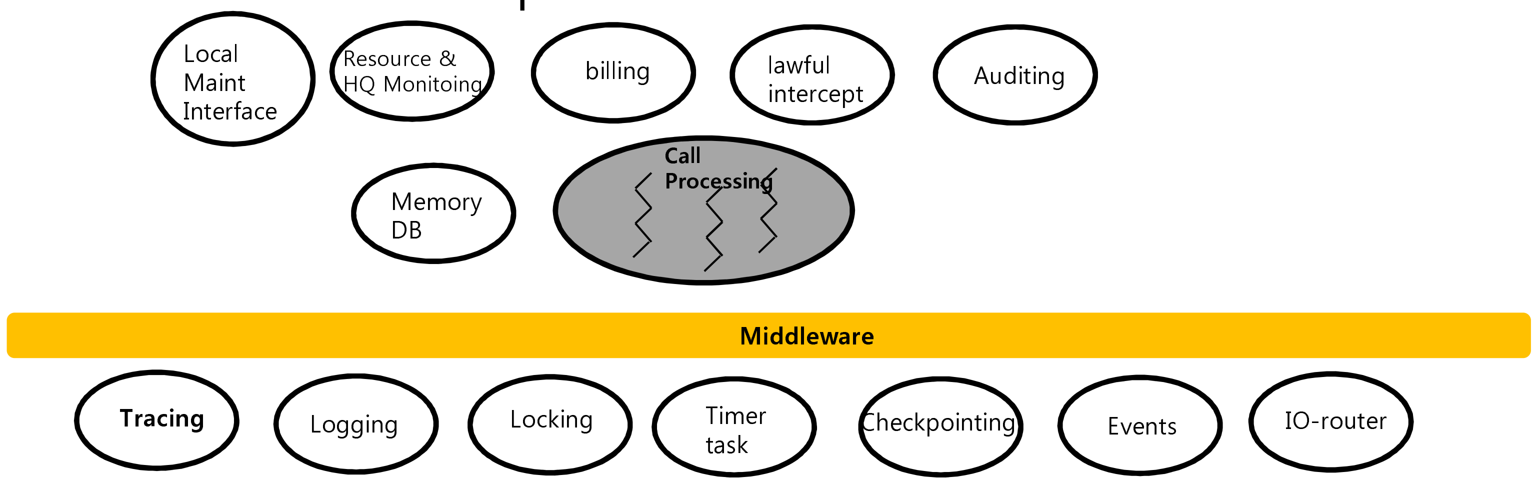
- Intel Xeon 2.3GHz, l1-cache 32k, l2 256k, l3 15MB – 12 CPUs
  - NFV – COTS
- kernel 4.1, QEMU 2.0.0
- Focus on Generic gaps
- Host/Guest PREEMPT –
  - Host – CONFIG_HZ_1000, CONFIG_NO_HZ
  - Guest – CONFIG_NO_HZ, CONFIG_HZ_500
  - Hosts/Guest(s) vCPUs pinned – 4 CPUs
- LMBench/rt-tests – both heavily used in wireless
  - LMBench – basic cost of operations
  - rt-tests – sched latency, migration delay
- Key NFV attributes –
  - COTS – VNF support
  - VNF Decomposition
  - Improved operational efficiency, scalability

Building a Network Element



1. Vary traffic
2. Collect Results
   - Determine Capacity, Latency
3. Back to one increase Load

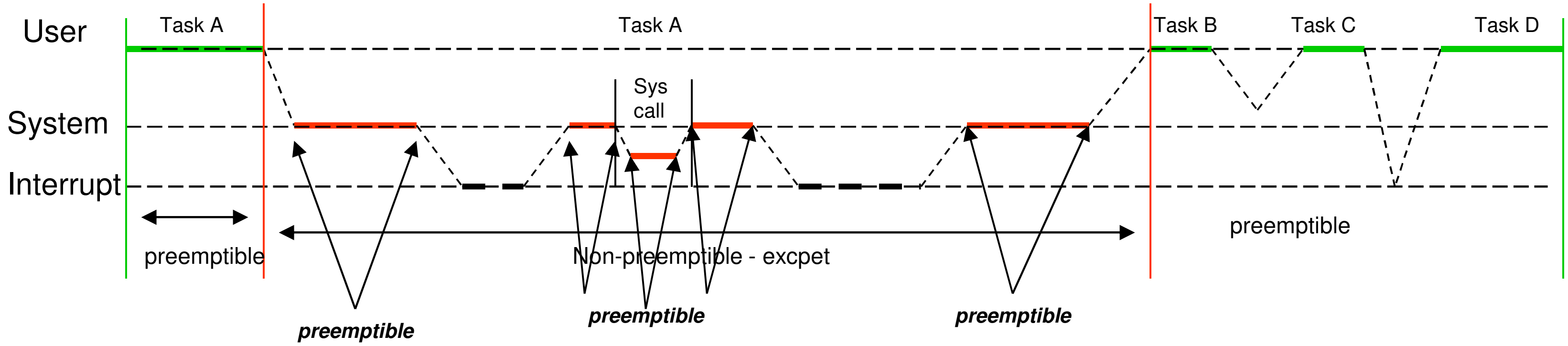
➢ Then come real requirements



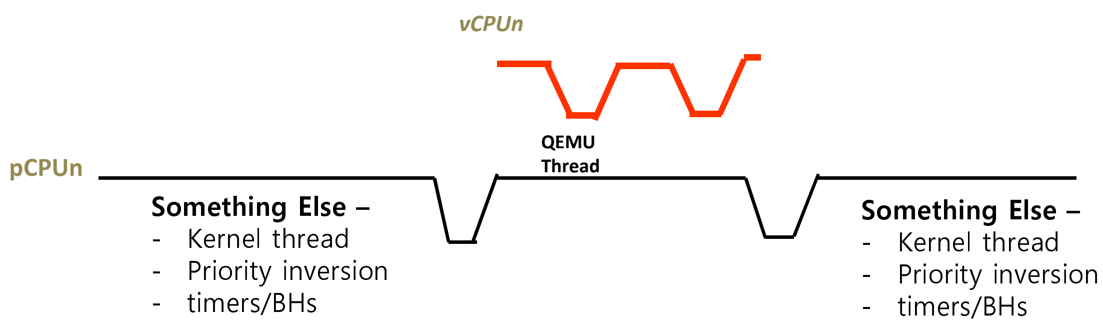
➢ Long iterative process – in short conflicting workloads
- Prioritize, vary load, determine capacity
- Defect arrival rate < X field trials

Two Level Scheduling & Determinism

- Control Plane need this – PREEMPT



- But winding up with this



Latency Testing

- Cyclictest – Host/Guest – idle system
- **Host**

  # taskset –c0,3 ./cyclictest -q –t20 –p 99 –n –i 500/5000 –l 10000 – **1-2% - CPU**

  Min 2uS **Max** ~16uS/390uS **Avg** ~2uS

- **Guest – vCPUs bound to cpu 0-3 – io thread to other, w/-realtime**
  - vCPU threads – SCHED_OTHER, 1 Guest - intervals 500uS & 5000uS

    Min 19uS **Max** 1000uS **Avg** 60uS - **40% CPU**

    Min 23uS **Max** 1200uS **Avg** 90uS - **20% CPU**
  - vCPU threads – fifo or -rr – priority 99

    Min 17uS **Max** 300us **Avg** 60uS - **40% CPU**

    Min 16uS **Max** 433uS **Avg** ~90uS - **20% CPU**
  - Two Guests - fifo/rr – priority 99

    Min 20uS **Max** 495uS **Avg** 65uS - **2 x 40% CPU**

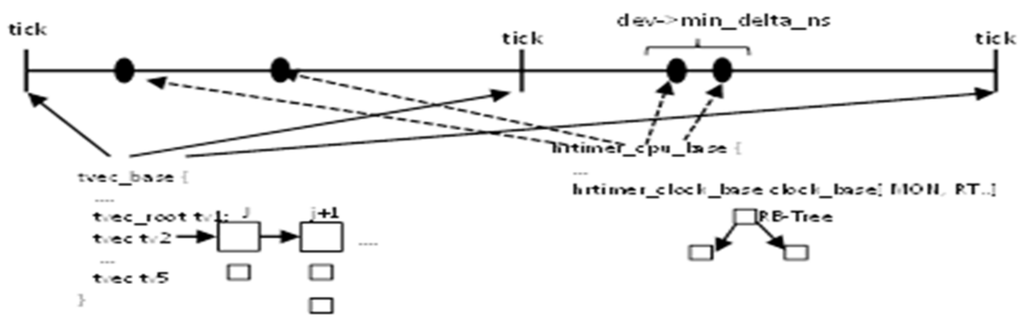
    Min 21uS **Max** 540uS **Avg** 100uS - **2 x 20% CPU**

- Conclusions
  - Guest Latencies reasonable
  - CPU high – kills COTS, managebility
  - Setting vCPUs to RR/FIFO helps lower MAX
  - Todo:
    - Host PREEMPT_RT – NO_HZ_FULL, nohz_full, rcu_nocbs – for Data Plane
    - High tick rate for Control Plane – tune kernel threads, ftrace, perf, ….,
    - CP/DP – decompose several VNFs?

Timers LMBench Test

- lat_usleep usleep | nanosleep ...
  - **Guest** – 50% slower (100uS to 54uS) – CPU usage up 20% higher

- To mitigate
  - Dedicated timer task – coalesces requests – x requests/interval
  - Overhead negligible – for 2mS coalescing



- Conclusions
  - Coalescing helps – not total solution (i.e. MME 20,000 SRs/5s – 250uS)
    - More vCPUs/pCPUs
  - How to deliver high timer rate to guest with low overhead?

MMU

- lat_mem_rd – 128MB strides 32/64/128/256 bytes
  - Latency to red 32/64/128/256 bytes over 128MB region
    - Guest Host CPU usage – constant 100% Host – 13-41% -
    - memory access latency doubled or 40% slower (nS ranges)

- bw_mem – 200MB rd/wr/rd – looks reasonable
  - Guest Host CPU – 104% Host 96%
  - Guest Host CPU - 102% host 97%
  - Guest Host CPU – 102% host 100%

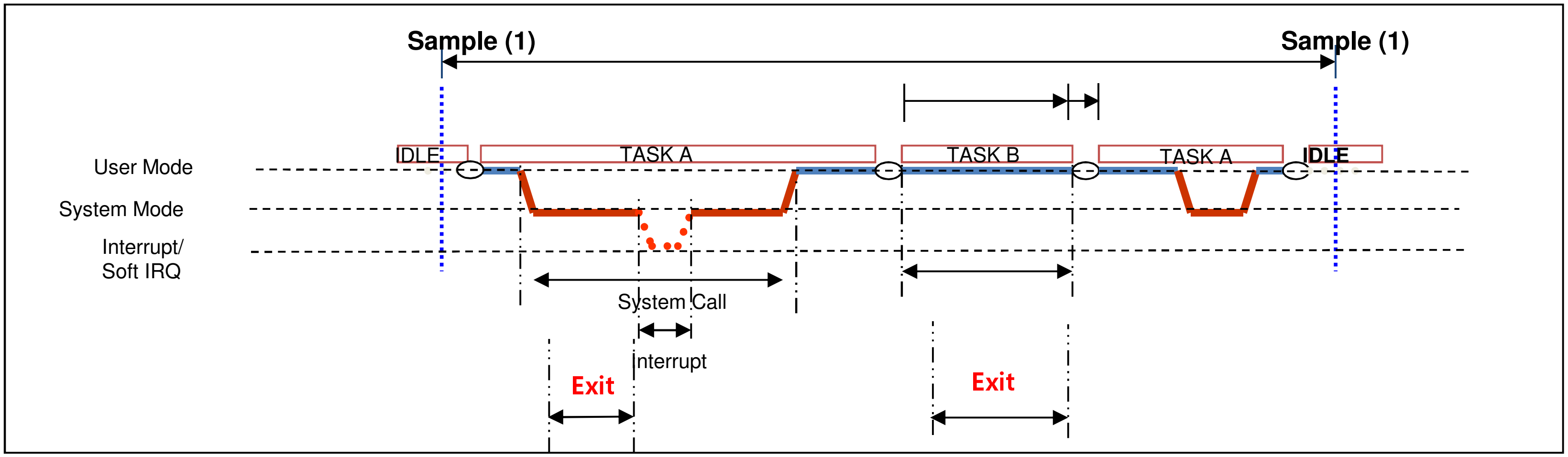
- lat_mmap/lat_ctx – some issues here
  - Host – 24uS CPU Guest 64uS – **mmap** – (not sure why?, could live with it)
  - CPU usage 81% host 41% - **ctxt** - with 8MB noise – nested walk?

- To mitigate – stripe memory across vCPUs – 128MB/4 – usage 45-62%
  - Thread/vCPU

- Conclusions?
  - Nested Page Table Walk, Guest friendly flushing
  - IPTW Cache – size/associativity – performance monitoring?
    - Need proper hw selection, benchmarking – close gap

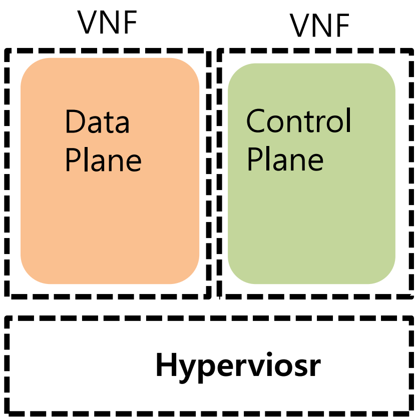
CPU Accounting now with Exits

- Now we have this



- 'spin' on Guest/Host – both show 100%
  - Cycle based accounting – per-cpu – w/more info
  - Load capacity mgmt confused –
    - SNMP trap – Guest & Exit time
      - UCD-SNMP-MIB – i.e. snmptable, …; snmpwalk <IP> UCD-SNMP-MIB::systemStats
      - Augmented by VM exit stats
    - Confusing to O&M – two indicators go red
    - Guest - associate exits with mode, thread, vCPU
    - O&M view VNF as NE – intelligent load scheding
- During development – rely on tools only available in field
  - That's all you get!

Inter VM IPC

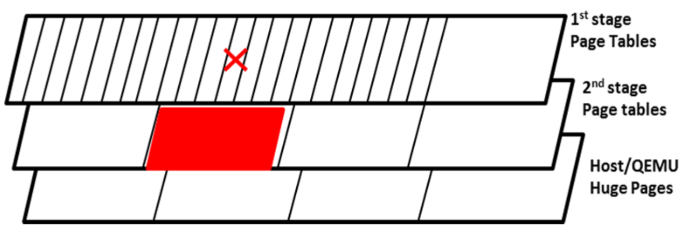


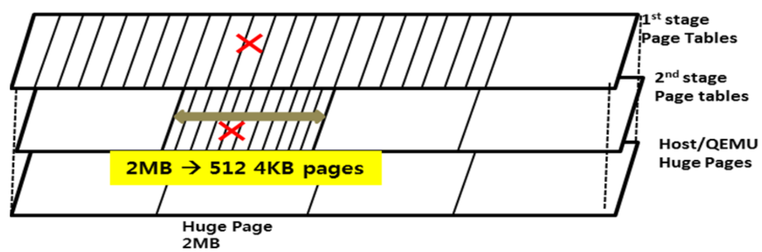
- Scale Vertically
  - Decompose VNF
  - In HA configuration
  - Posix like IPC

- Accelerated synchronization & message passing between VMs
- Slow path inter-guest interrupt
  - Like ivshmem – very slow
- HV Call interface – requires new code
- Fast path – dedicated synchronization support (ARM f.e.)
  - ARM SEV, WFE – wakes up everyone
  - SEV #imm, WFE #imm – associate with Guest
  - Instructions Hint, Scope unknown – most likely needs hw extensions
- Posix - like shared memory discovery – ivshmem a start

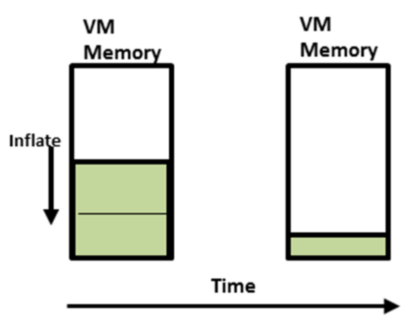
VM Management

- Rapid migration w/huge pages – EPS NE with memory DBs
  - Huge pages performance, slows migration –near idle loads succeed
  - Function of - mem size and dirty rate
    - Shorten downtime
  - To mitigate - split during migration, merge after
  - Much higher dirty rates supported





- Ballooning – unreliable
  - Close gap between issue request and execution – prevent lockup
  - Mix of locked rt and non-rt code



Other LMBench operation Latencies

- More to do's
- System Calls – **??**
  - lat_syscall
  - Read - Host .11uS, Guest .31uS
  - Write - Host .16uS, Guest .32uS
  - File - 1.5uS, vs. 3.82uS
- Signal Delivery – **??**
  - lat_sig catch Host .85uS vs. Guest 2uS
- latency on fork, exec, shell - 50% higher
  - To mitigate use threads – dont fork()/exec()
    - But in CG – threads hard to debug, unsafe
    - CG fault recovery model – save FDs, checkpoint state, restart
      - CG – use system("....") – do something
      - SAF services
- Conclusions –
  - Sys calls/Signals– **should be native???**
  - For/Exec/Shell – IPI costly

Other Gaps

- HW, enhancements, awareness i.e. -
  - vCPU and IO-Threads locality & NUMA
  - Host doesn’t swap kernel pages, Guests kernel pages can
    - Realtime – lock pages – but limits overcommit
  - Guests more then tiny, .., large – resource + behavior (preempt/voluntary)
  - Guest Overcommit – small guest don’t forget QEMU
  - AsyncPF – powerful feature – w/o temp CPU unplug
  - world switch costly
  - Interrupt injection -
    - Direct injection for IPIs, Device, Timers
    - IRQ affinity vCPU to pCPU – on exit return inject to vCPU
  - Device Pass-through
    - Some not behind IOMMU – i.e. HPI controller
    - Not all NICs – crypto devices

Migrating NFV Applications to KVM Guest

Q & A

Thank you.