Problems in DC’s and why ICCRG/TCPM should care
What’s *new* in DCs

- High-speed access links and getting to comparable speeds to the DC core
- Operators considering multipath topologies attempting to provide full bisection bandwidth
- Very low latency, between racks its in 100s of us
- Low statmux
- Moving towards scale out designs with commodity switches
- High-burst tolerance AND low latency AND high-throughput
- Need ability to assign any service to any server to prevent resource fragmentation
- Workloads can migrate often and can require preserving connectivity
Multi-tenancy: Server/Network Virtualization

• 8-32 VMs per server depending on who you ask
• Unprecedented scale that’s pushing all our protocol limits (ARMD is one such example)
• Don’t trust the VM
  – Even if you do trust the VM image they don’t deploy algorithms that are relevant for DC’s they are all designed for the Internet
• Performance isolation is super hard
  – State-of-art-capacity sharing algorithm is ..... TCP
  – TCP operates on the wrong granularity
Examples of fundamental problems

• Cost of ToRs – deep vs shallow buffers
  – Say 200k servers/20 per ToR – 10k ToRs
  – 10k * $7000 savings = $70 million

• Burst tolerance and Incast
  – Reduce MinRTO
  – DCTCP aims to change sender congestion control
  – ICTCP which is based on a receiver window

• Performance isolation using TCP?
  – Trying to solve flow fairness
Performance Isolation in DCs

SEAWALL – CONGESTION CONTROLLED TUNNELS
What Do We Want To Achieve?

• Protect tenants from availability attacks
  – Internal DoS can wreak havoc

• Enforce tenant-specific quota
  – Customers purchase their weights, or admins can assign customers’ weights

• Reduce network-performance interference among tenants
  – Elimination is more difficult
What Properties Do We Want?

• Traffic agnostic, lean service interface
  – Customers should be allowed to use any protocol, generate any traffic patterns
  – Have customers choose their network weights only

• Scalable
  – Support $O(10^5)$ VMs, $O(10^4)$ tenants, and $O(10^3)$ deployment events per day

• Work-conserving (efficient)
  – Allow tenants to use residual capacity
  – Max-min fairness

• Require no change to network topology and devices
  – Avoid relying on mechanisms that require VM cooperation or special features in network devices
Why Existing Solutions Fall Short?

• TCP
  – Wrong granularity
  – Can’t enforce quota
  – Doesn’t meet customers’ needs for UDP
  – Cannot trust TCP traffic from VMs anyway

• Link-local QoS (queueing and rate limiting)
  – Not scalable, can be wasteful, and expensive

• Bandwidth reservation (RSVP, MPLS TE)
  – Overly conservative at low loads, or overly lenient at high loads
  – Enforce isolation even when congestion doesn’t exist
Why Existing Solutions Fall Short?

- **VL2 (Oversubscription-free network)**
  - Hose-model incompliant traffic (UDP) will happen
  - Can’t enforce quota

- **DCTCP (Less-bursty TCP)**
  - Avoids performance interference among different types of apps, but not among different tenants
  - Can’t enforce quota

- **QCN (IEEE’s L2 congestion control)**
  - Limited to a single L2 domain
  - Wrong granularity
  - Can’t enforce quota
  - Introduces network-device changes
Very Basic Seawall

• Use congestion-controlled edge-to-edge tunnels
  – All 5-tuple flows between a pair of sending entities (e.g., VMs) are bundled
  – Receiver periodically sends feedback to sender, notifying congestion (if any)

• Upon congestion, weight-proportionally rate limit tunnels
  – Each sending entity is given a weight
  – Guarantee bandwidth proportional to this weight at every bottleneck link that the entity uses
  – Weighted AIMD (additive-increase, multiplicative decrease)
Birds’ Eye View

- Seawall introduces a shim layer to intercept packets.
Combining Feedback From Multiple Destinations

1: \textbf{Begin}(weight $W$)
2: \{ $rc_l$.\textbf{Begin}(W) \forall$ links $l$ used by sender \} \hspace{1cm} \triangleright \text{Initialize}
3: \textbf{.TakeFeedback}(feedback $f_{dest}$)
4: \{ store feedback \}
5: \textbf{.Periodically}()
6: \{
7: \text{proportion of traffic to } d, \; p_d = \frac{f_d.\text{bytesRecv}}{\sum f_i.\text{bytesRecv}}$
8: \textbf{for all} destinations $d$ \textbf{do}
9: \quad \textbf{for all} links $l$ on path to $d$ \textbf{do}
10: \quad $rc_l$.\textbf{TakeFeedback}($f_d$, $p_d$)
11: \quad \textbf{end for}
12: \textbf{end for}
13: n_l \leftarrow \text{count of dest with paths through link } l$
14: \text{allowed rate to } d, \; r_d \leftarrow \min_l \left( \left( \beta p_d + \frac{1-\beta}{n_l} \right) r_{cl}.rate \right) \hspace{1cm} \forall \; \text{links } l \; \text{in path to } d$
15: \}
Why should IETF/IRTF care?

• These are real problems
  – Lots of innovative work happening so they will get solved one way or the other
• Design teams aren’t sufficient we need to understand the landscape better and experiment
• Hypervisor as a middlebox is clearly not ideal and won’t scale
• Treat the DC problems as *mainstream* because SR-IOV will force these solutions into the OS
  – OR ISPs may move towards DC designs and ask for these
• Either way they will trickle to the Internet