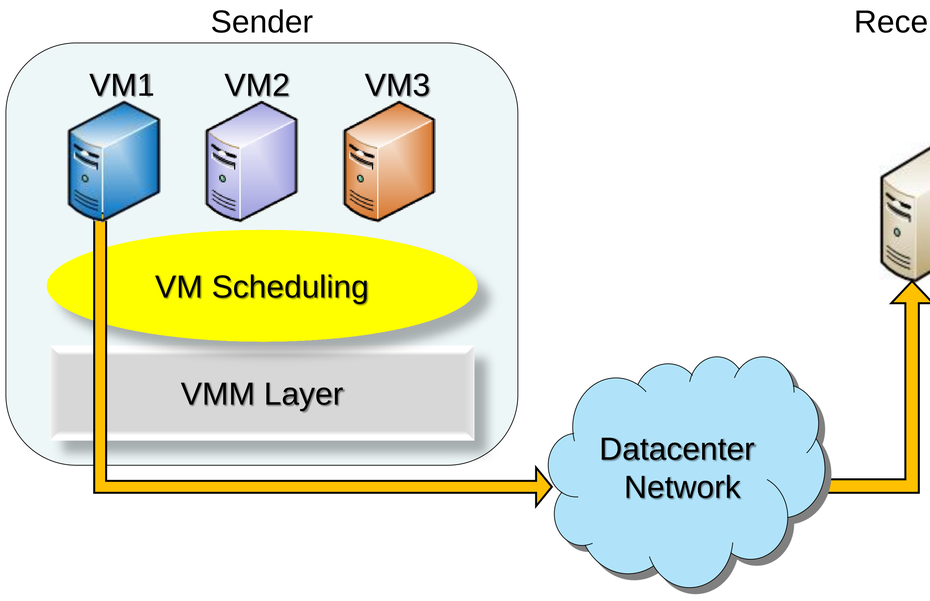
vFlood: Opportunistic Flooding to Improve TCP Transmit Performance in Virtualized Clouds

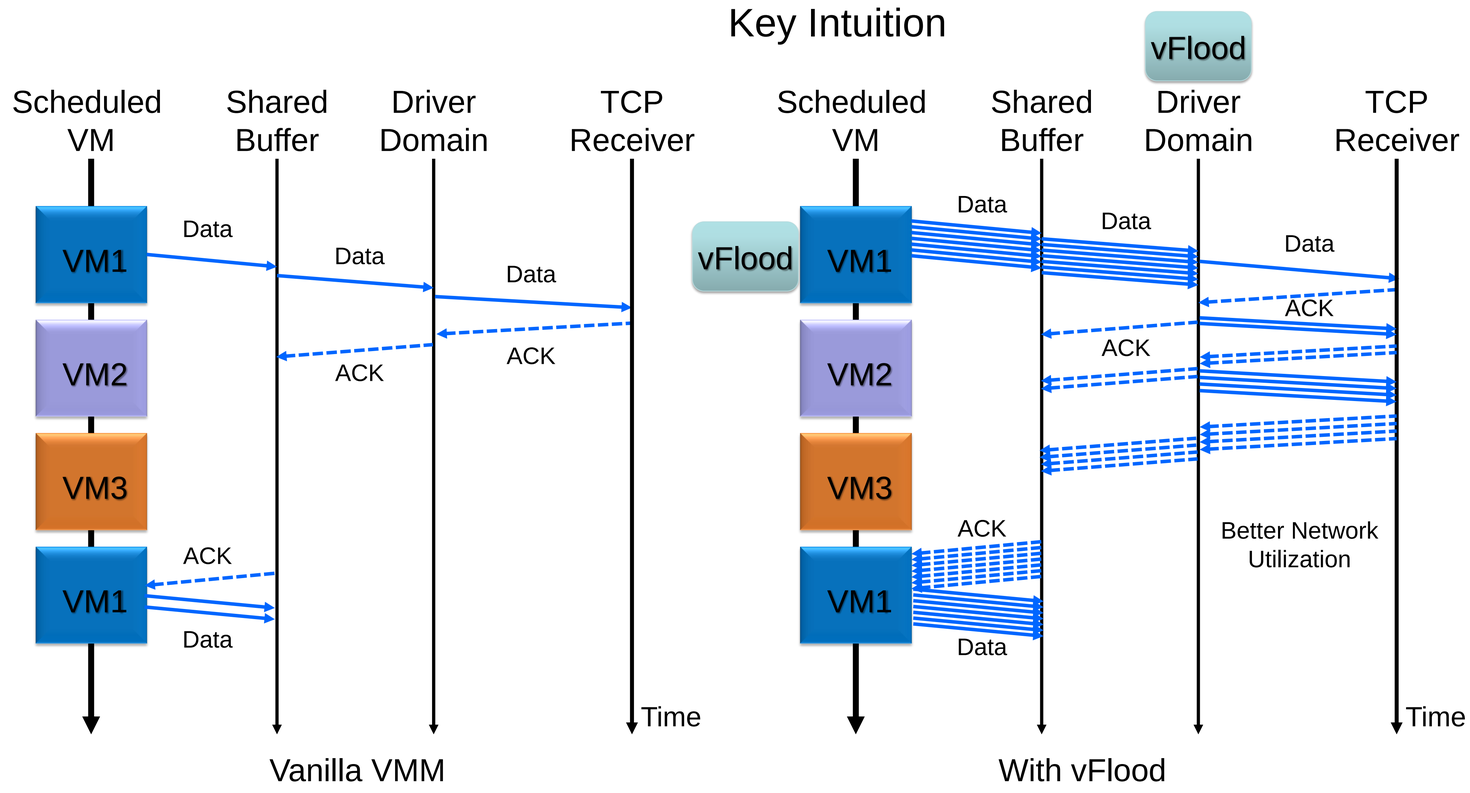
Sahan Gamage, Ardalan Kangarlou, Ramana Rao Kompella, Dongyan Xu
Department of Computer Science

Motivation



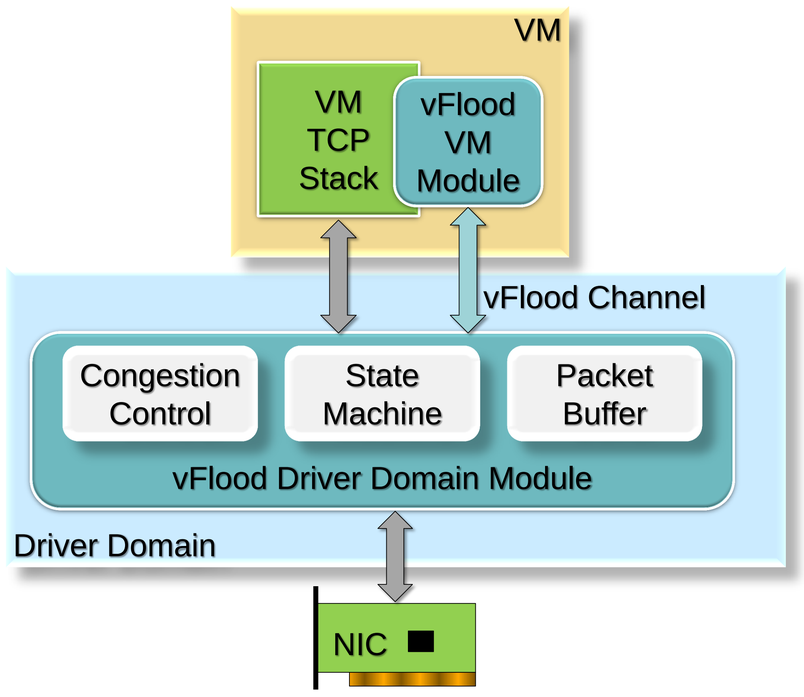
- Virtualization allows many VMs to share resources of a single host
  - Sharing a single CPU among multiple VMs
- CPU access latency is increased due to VM scheduling
- Data packets arriving from network is queued until target VM gets scheduled
- Datacenter environment : sub-millisecond network latency
  - Virtualization overhead dominates round trip time (RTT)

Key Intuition



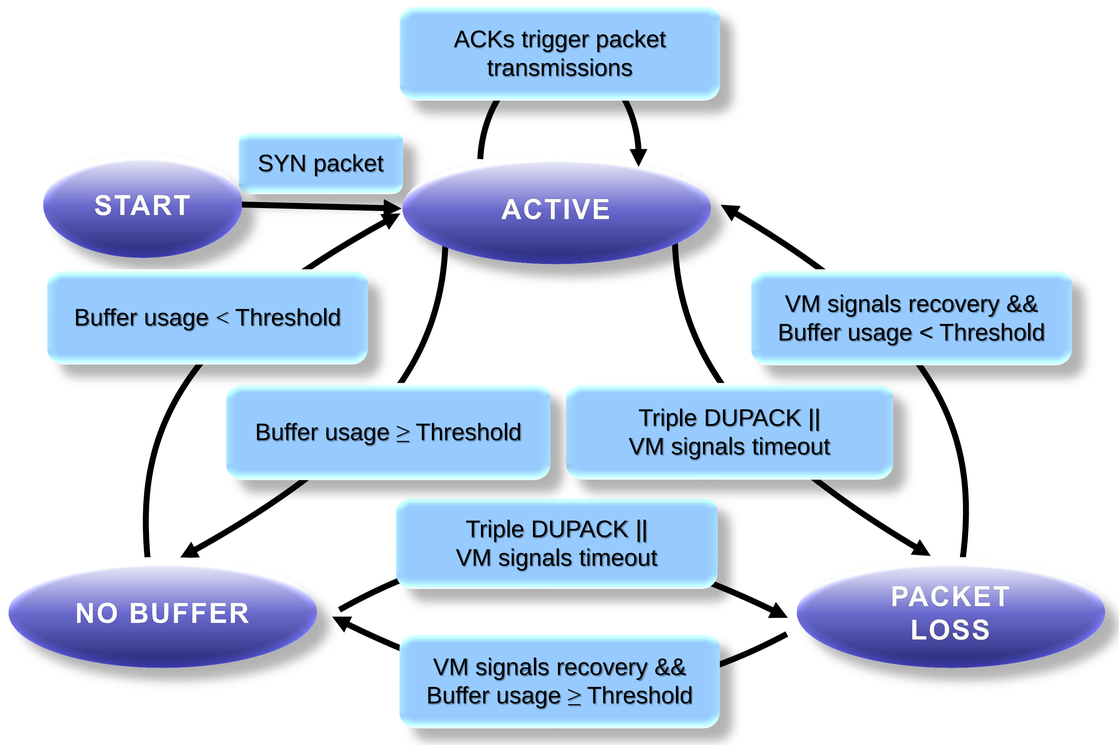
- VM 1 - VM3 sharing a single CPU
  - VM1 transmits data to a remote host
  - ACK packets arriving are queued until sending VM gets scheduled
  - Slow growth of congestion window ► slow progress of the connection
- vFlood : Flood as many packets as possible to the driver domain
- Driver domain executes congestion control on behalf of the VM
  - Congestion control offload

vFlood Design



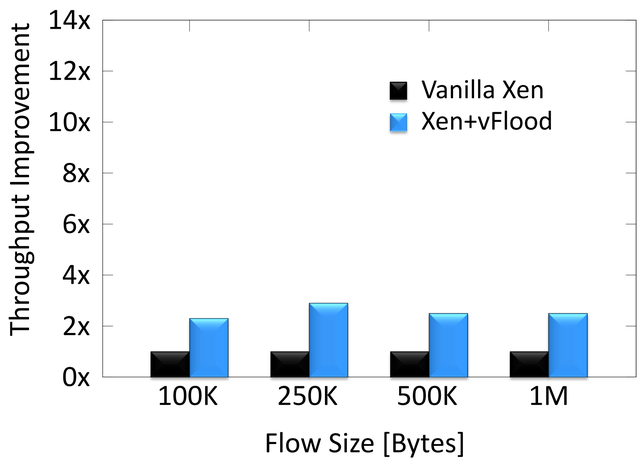
- Driver domain implements congestion control
  - Buffer packets coming from VMs (per flow packet queues)
- State machine is maintained per flow basis
  - Sequence numbers, congestion window size, current ACK number , current state of connection
- vFlood VM component sets VM TCP congestion window to current size of the driver domain buffer
  - Communicates with driver domain component to get buffer information
- Retransmissions are handled by the VM
  - vFlood is disabled when a packet loss is occurred

vFlood State Machine

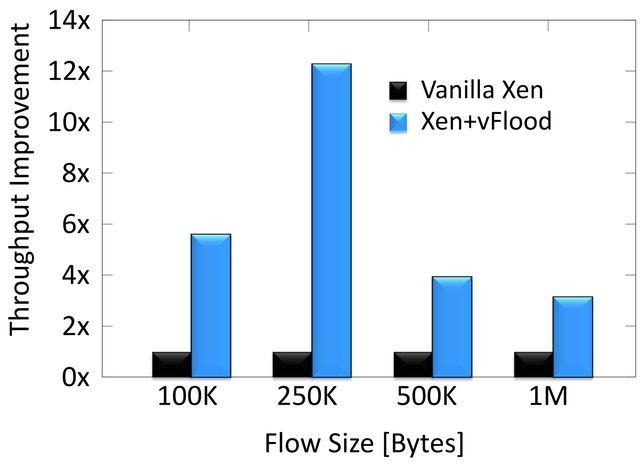


- Significant performance improvements for small flows
  - 3X to 12 X improvement when 3 VMS are sharing the same core
- Gains increase when more VMs are sharing the same CPU
- Non-trivial improvements for large flows
  - 23% - 31% for 100MB flows

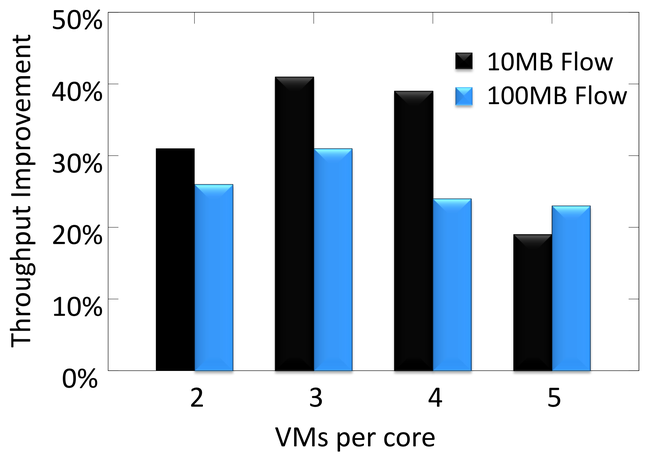
TCP Throughput Improvement



2 VMs sharing same CPU



3 VMs sharing same CPU



Improvement for large flows