Defeating Network Jitter for Virtual Machines

Luwei Cheng, Cho-Li Wang, Sheng Di
The University of Hong Kong

Dec. 6th, 2011
Melbourne, Australia

The 4th IEEE/ACM International Conference on Utility and Cloud Computing

Outline

- Research Motivation & Methodology
- Problem Analysis
- Our Solutions
- Implementation & Performance Evaluation
- Conclusion

Research Motivation

Cloud Computing

- Cloud Computing Service Model:
  - SaaS (Software as a Service)
    - Google Docs, ERP-related software, etc.
    - Service is directly provided to end-users
  - PaaS (Platform as a Service)
    - Windows Azure, Google AppEngine, etc.
    - For developers
  - IaaS (Infrastructure as a Service)
    - Amazon EC2, GoGrid, rackspace, etc.
    - Network administraters, architects

Go to Cloud?

- It is a campaign: move to cloud datacenters!
  - Low cost, elasticity, easy management, …

- Cloud datacenters: use virtual machines to provide hosting services.

Go to Cloud?

- **Question**: do ALL applications work well in cloud datacenters?

- **Observations**:
  - 1. The negative impact of virtualization on IP telephony applications [Patnaik et al. IPTComm'09]
    - For media applications, the setup in the virtualized environment can be very challenging.

  - 2. The unpredictable network behavior in Amazon EC2 platform [Barker et al., SIGMM'10] [Wang et al., Infocom'10]
    - Quite unstable network latency

The future…

- Reality: tens of VMs co-run in one physical server
  - Running forty to sixty VMs per physical host is not rare; A known case runs 120 VMs per host [Pfaff et al., HotNets'09]
- Trend: the hardware becomes increasingly powerful, which makes the consolidation level be higher and higher
  - More VMs share one physical core

- Therefore, it is necessary to investigate whether the network performance isolation solutions are effective

Research Methodology

- Application-driven
  - Today's applications are increasingly network-intensive
  - Audio/video streaming is highly demanded by internet users
  - Far more demanding for stable network condition
    - Very sensitive to network latency (desire low-jittered network)
- Top-down approach
  - Observe → Analyze → Solution → Verify

Problem Analysis

Network Performance Isolation

- For media streaming applications, network performance isolation means:
  - Predictable network bandwidth
    - The media data won't get lost too much
  - Low-jittered network latency
    - With client side buffer, long but stable network latency is tolerable
    - Largely varied latency affects QoS (RTP protocol)

Fig. 3.7 Original Video

Fig. 3.8 100ms ±4ms delay

Fig. 3.9 100ms±10ms delay

Fig. 3.10 100ms±16ms delay

Source (available online):

"Effect of Delay/Delay Variable on QoE in Video Streaming", Master Thesis, Blekinge Institute of Technology, May, 2010

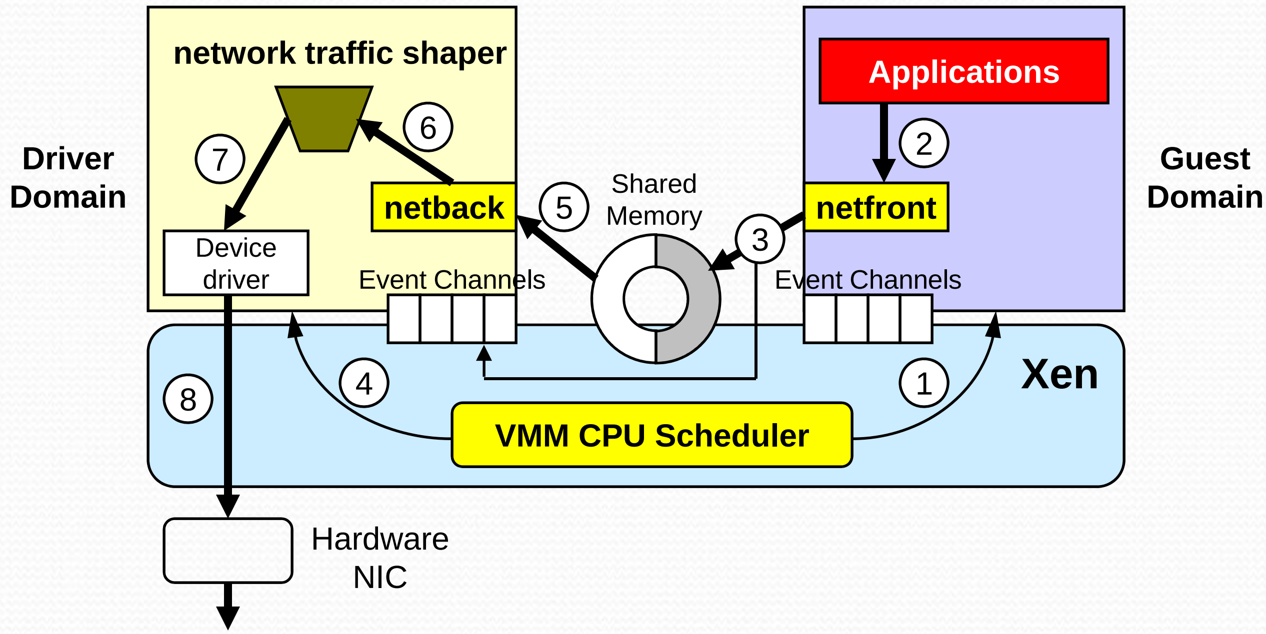
Problem Analysis

- The current resource sharing methods for VMs:
  - Mainly focus on resource proportional share
  - CPU amount, memory size, network bandwidth
- I/O latency is mostly related to resource provisioning rate
  - Even the VM is allocated with adequate resources such as CPU time and network bandwidth, large I/O latency can still happen if the resources are provisioned at inappropriate moments.
  - For example: 50% = 5ms/10ms, 50% = 500ms/1000ms.
    - BUT, 5ms/10ms != 500ms/1000ms (service latency)

Problem Analysis

- The resource allocation with only quantitative promise does not sufficiently guarantee performance isolation
- The problem is not only *how many/much* resources each VM gets, but more importantly whether the resources are provisioned in a *timely* manner.
- For resource allocation methods, there are two goals to be achieved:
  - Resource proportional share
  - Resource provisioning rate (for I/O latency)

Problem Analysis—a technical view



- Network Latency in virtualized hosted platform:
  - (1) VMM CPU scheduler
  - (2) Network traffic shaper

So…

- The I/O latency problem should be solved in two components
  - Reduce VM scheduling delay in VMM CPU scheduler
    - CPU proportional share
    - Provide real-time support for specific domains
  - Smooth packet delay in network traffic shaper
    - Limit network bandwidth consumption
    - Provide smoothed packet delays

The CPU scheduler in Xen

- Credit Scheduler:
  - Each VM is allocated with certain *credits*, according to its *weight*

- Boost Mechanism:
  - Temporarily give the VM that receives external events a BOOST priority with preemption, which is higher other VMs in UNDER and OVER state.
  - Reduce VM's scheduling delay for I/O in a best-effort way

The CPU scheduler in Xen

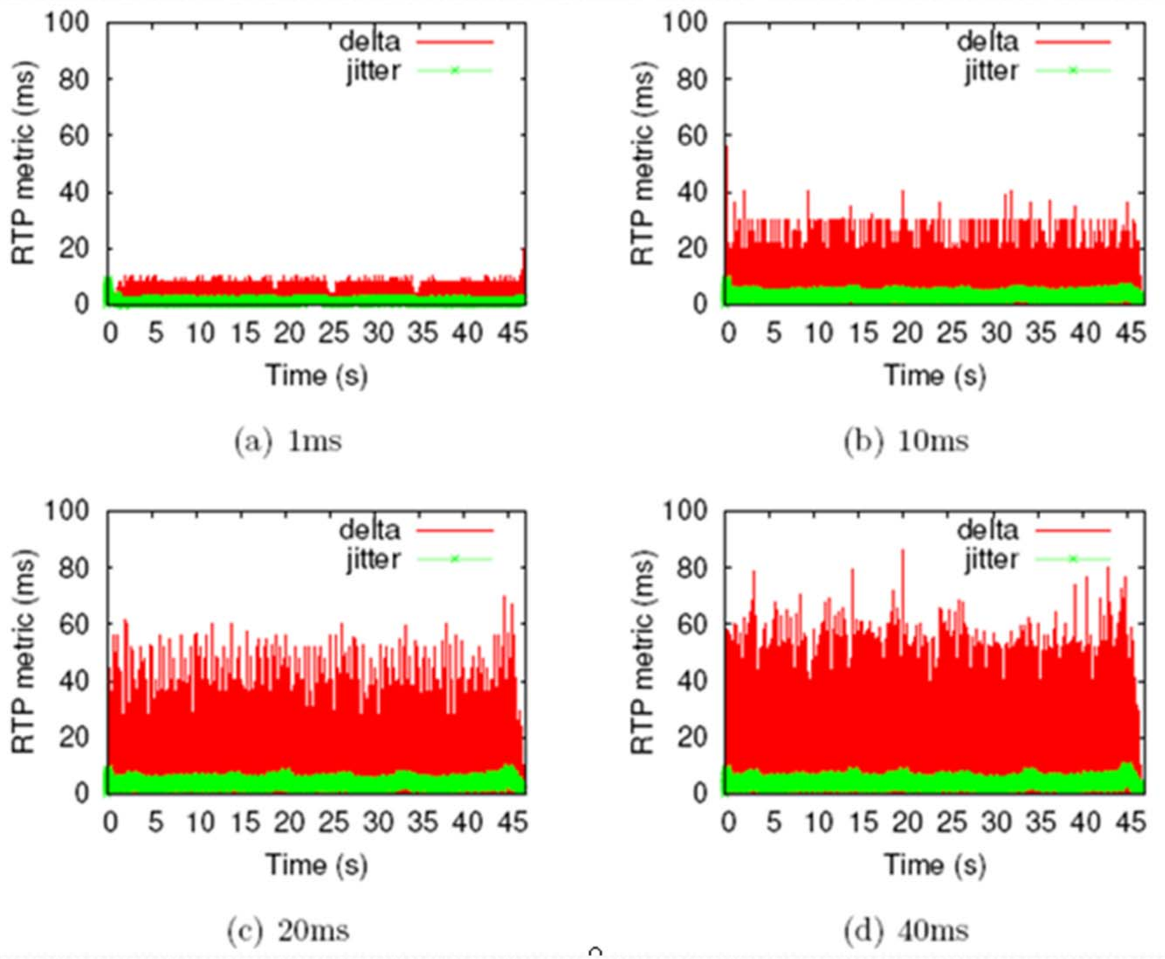
- Why Boost mechanism in Xen's Credit Scheduler does not work well?
  - It makes an assumption on "external events"
  - To virtual machines, ingress I/O is presented as "external events" (virtual interrupt)

- BUT, not all VM's I/O is "event-triggered"!
  - Ingress I/O: user data → VM (get notified by event)
  - Outgoing I/O: VM data → user (no event for VM)

Characterizing VM's I/O type

- We classify it into two types:
  - Event-triggered I/O
    - User request → VM reply
    - Only when external event comes, the VM needs to be scheduled as soon as possible
    - Aperiodic real-time domains
  - Self-initiated I/O
    - No external triggering during I/O data transmission
      - Media streaming applications are of this type!
    - VM needs to be scheduled periodically
    - Periodic real-time domains

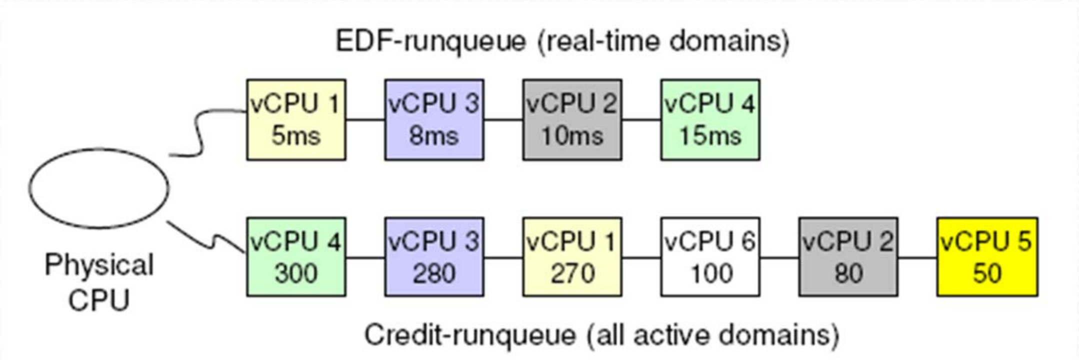
Self-initiated I/O (RTP video streaming)



- The VM runs alone on a dedicated CPU core
- Under Xen's Credit Scheduler, the VM is activated every 1ms, 10ms, 20ms and 40ms respectively

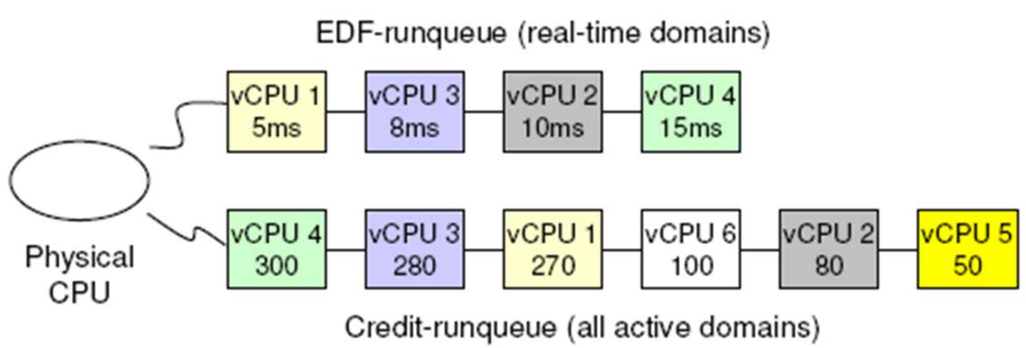
Solution for VMM CPU scheduler

- Double-runqueue design for each physical core
  - Credit-runqueue
    - Maintain CPU time proportional allocation
  - EDF-runqueue
    - Provide real-time scheduling support for specific VMs



Solutions – VMM CPU scheduler

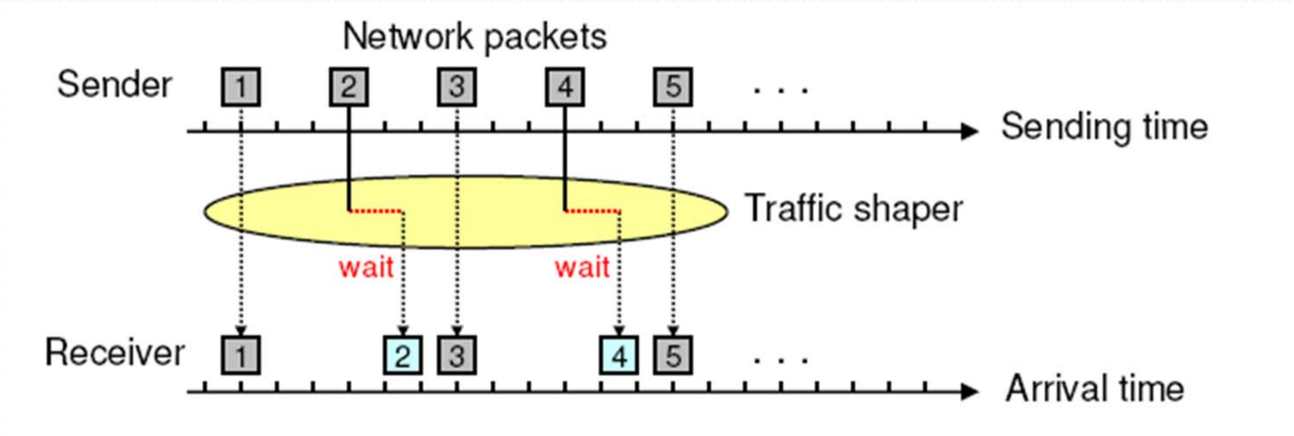
- VMs are classified as:
  - Normal VMs
    - Only stay in Credit-runqueue
  - Periodic real-time VMs
    - Stay in both Credit-runqueue and EDF-runqueue
  - Aperiodic real-time VMs
    - Only when they receive external events, they can enter EDF-runqueue



Network traffic shaping

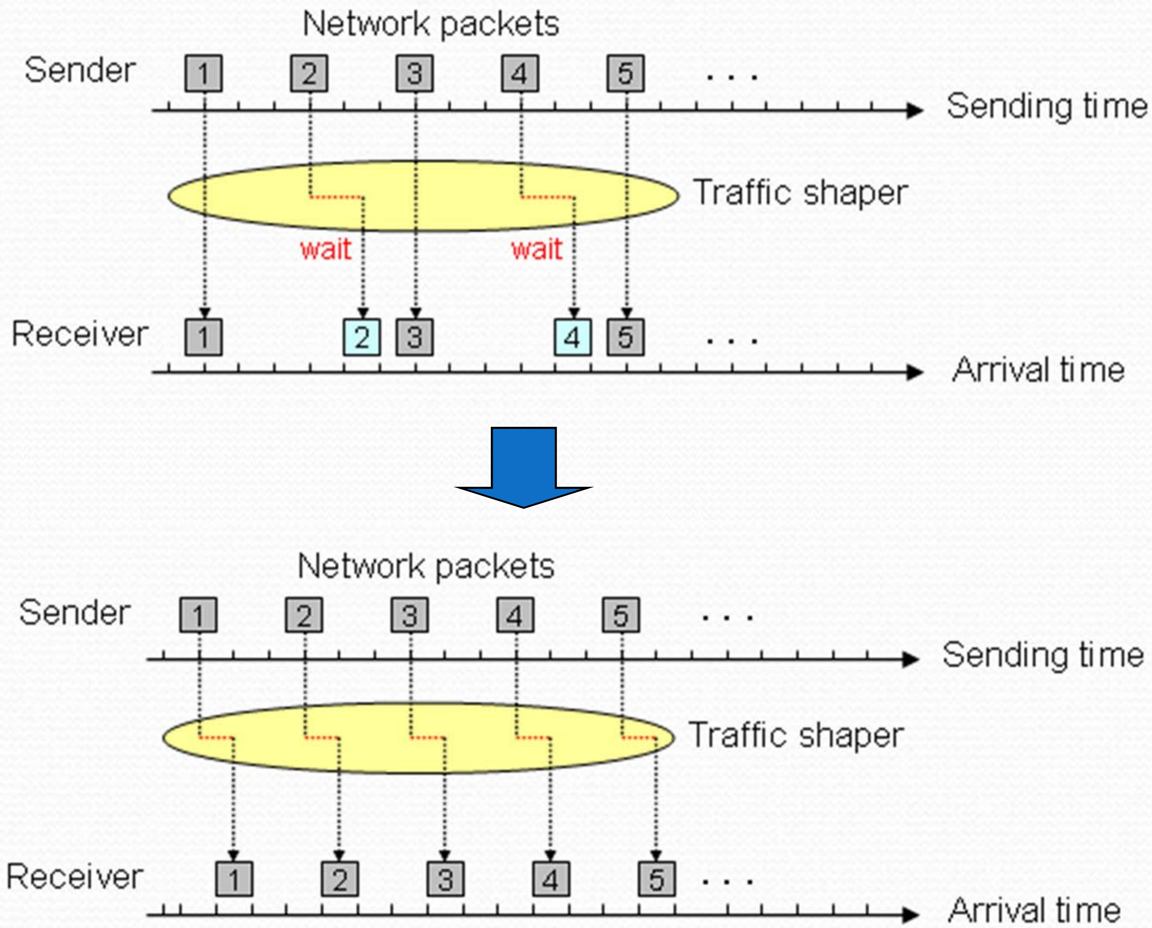
- Traffic shaping (rate limiting) is always achieved by delaying packets
- Xen implements token-bucket algorithm. It works as:
  - If the tokens are enough, packets are sent at once
  - Otherwise, packets have to wait for new tokens. It depends on how frequent credits are replenished.
- Token-bucket algorithm works well in bandwidth shaping, but has no guarantee for the delay of each packet.

Network traffic shaping



- Improper delays to each packet cause significant network jitter

Proper way to add delay



Problem

- How to determine the delay of each packet?
  - Long delay
    - The packets are sent too slowly
    - Low network resource utilization
  - Short delay
    - The packets are sent too fast
    - Violates the bandwidth allocation

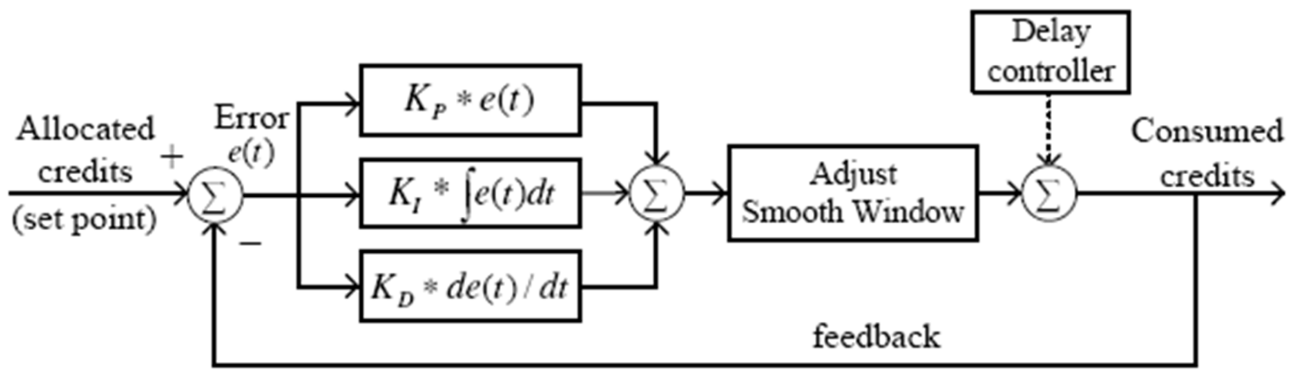
Goals

- The delay should be adaptive
  - As long as it does not significantly vary within a certain period!
- Two goals:
  - Does not violate network bandwidth allocation
    - No over-consumption, no under-utilization
  - Provides smoothed delay

Solutions

- Smooth window [d_{min}, d_{max}]
  - Control the sending delay of network packets
  - Guarantee that the delay does not significantly vary within a certain period
- Feedback control
  - Dynamically adjust window position according to bandwidth assumption
  - Why do we use feedback control?
    - Applications' network behaviors are unpredictable.
    - It is impossible to accurately model it.

Feedback control (PID controller)



- Measure "credit control error"
  - Consumes too much?
    - Longer delay for subsequent network packets
    - [3ms, 6ms] → [5ms, 8ms]
  - Too low utilization?
    - Shorter delay for subsequent network packets
    - [5ms, 8ms] → [3ms, 6ms]

Implementation

- VMM CPU scheduler
  - In Xen 4.1.0
  - Based on current Credit Scheduler
- Network traffic shaper
  - Network backend driver in Linux 2.6.32.13
  - Based on token-bucket algorithm
- Xen-tools are extended
  - Allow users to specify VM's real-time requirements
  - For example: type = periodic, deadline=5ms

Performance Evaluation

Experimental Setups

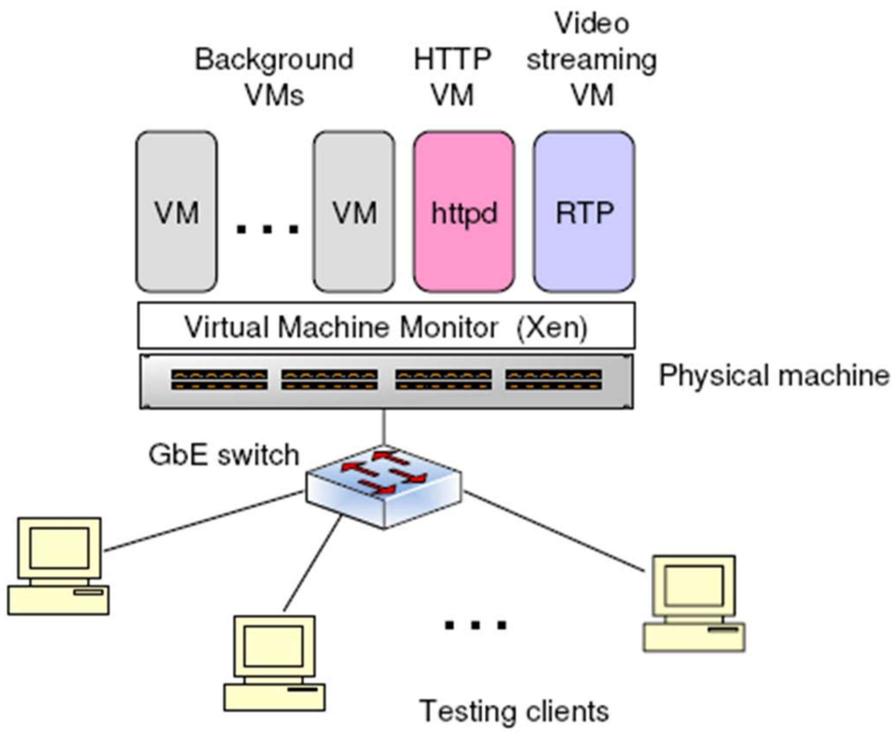


Figure 6.1: Experimental setup

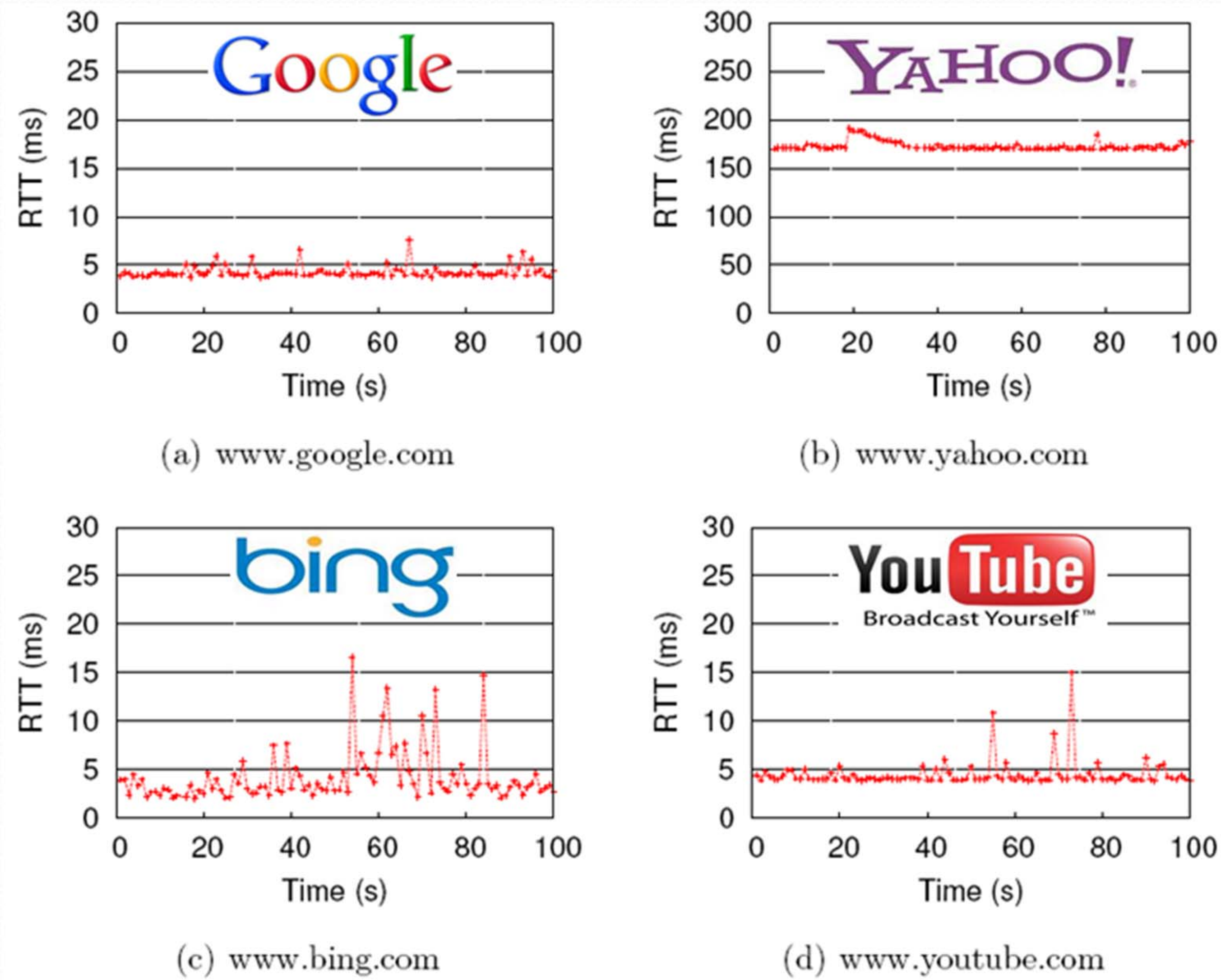
- Hardware
  - CPU: two quad-core Intel Xeon 5540 2.53GHz
  - Memory: 16GB
  - Network: Gigabit Ethernet Switch
- Software
  - Xen 4.1.0
  - Linux 2.6.32.13

VMM CPU scheduler

- Evaluation goals:
  - The ability to reduce network jitter
  - The ability to maintain CPU time proportionality

- Benchmarks:
  - Ping and Iperf
  - RTP video streaming

Network jitter on internet?



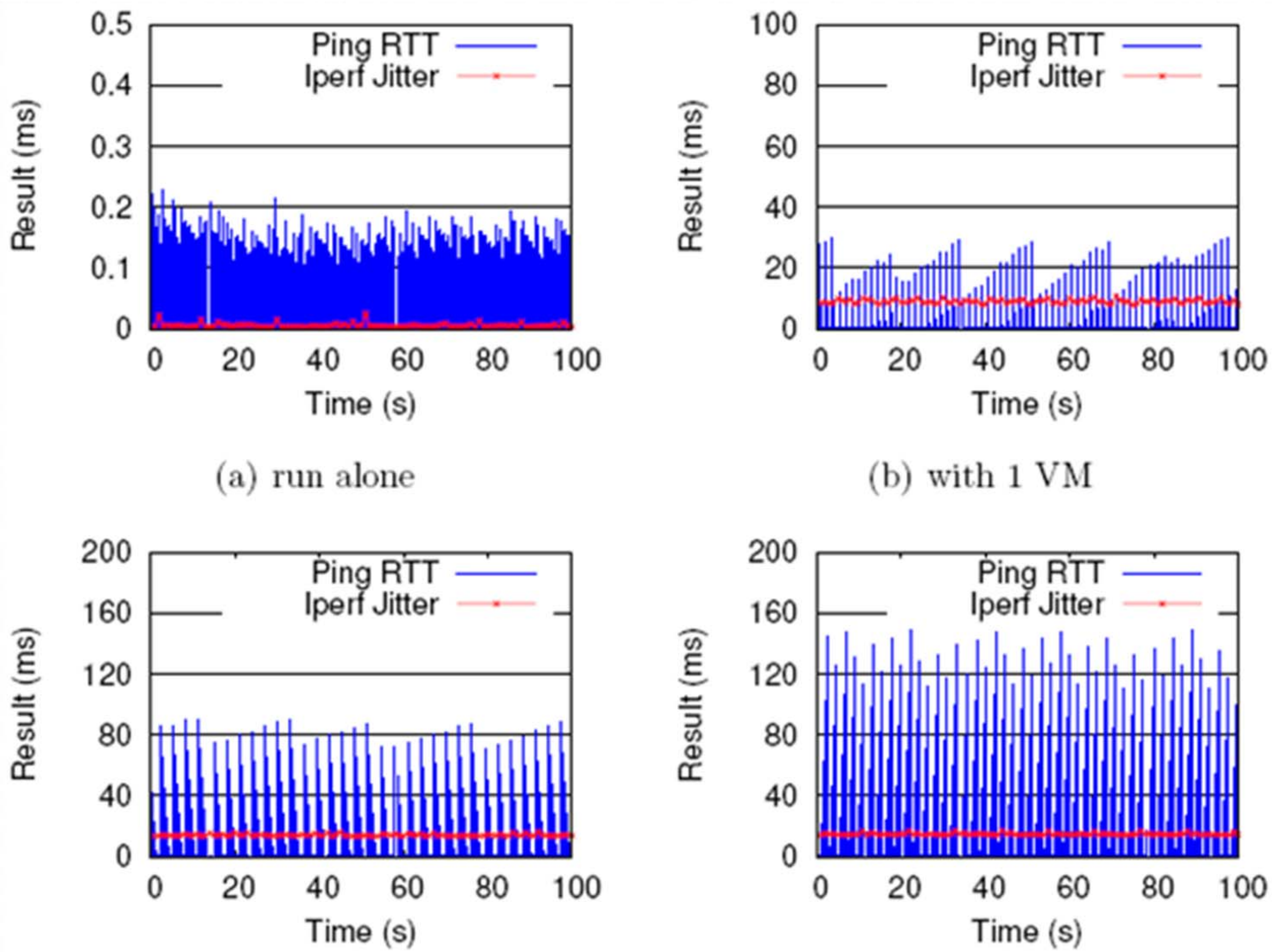
(a) www.google.com

(b) www.yahoo.com

(c) www.bing.com

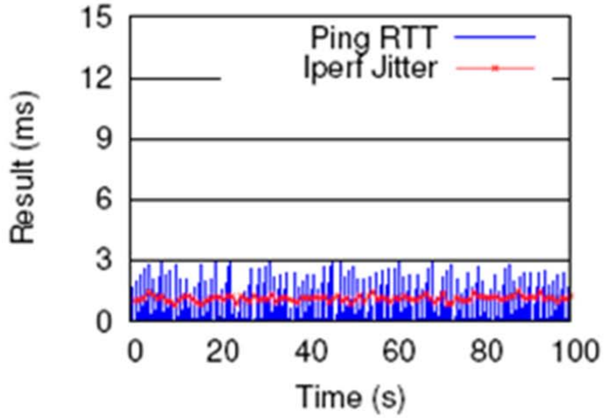
(d) www.youtube.com

With Xen's Credit Scheduler

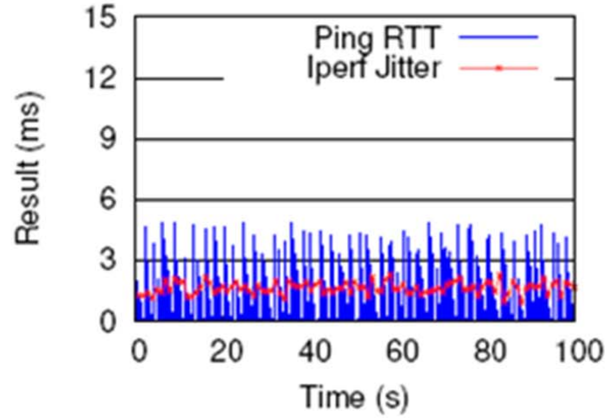


(a) run alone
(b) with 1 VM
(c) with 3 VMs
(d) with 5 VMs

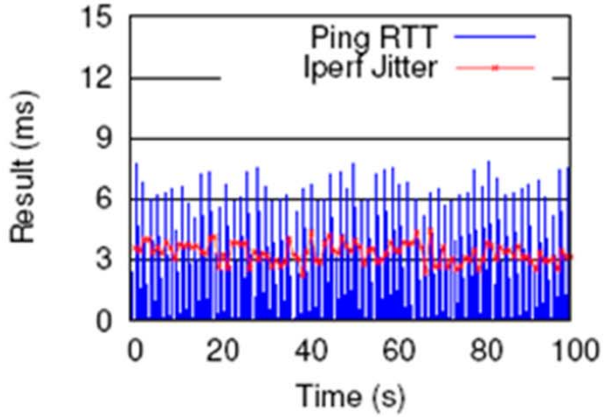
With our new CPU scheduler



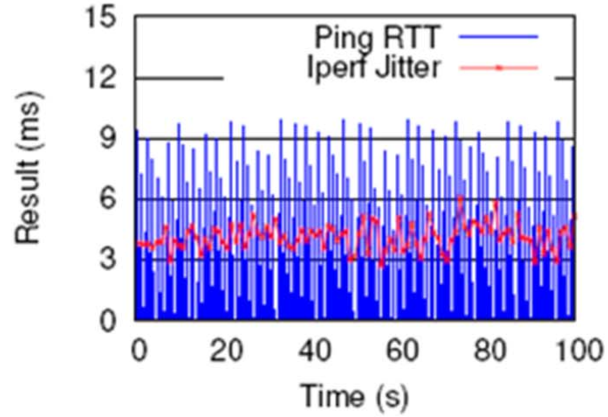
(a) our scheduler, deadline = 3ms



(b) our scheduler, deadline = 5ms

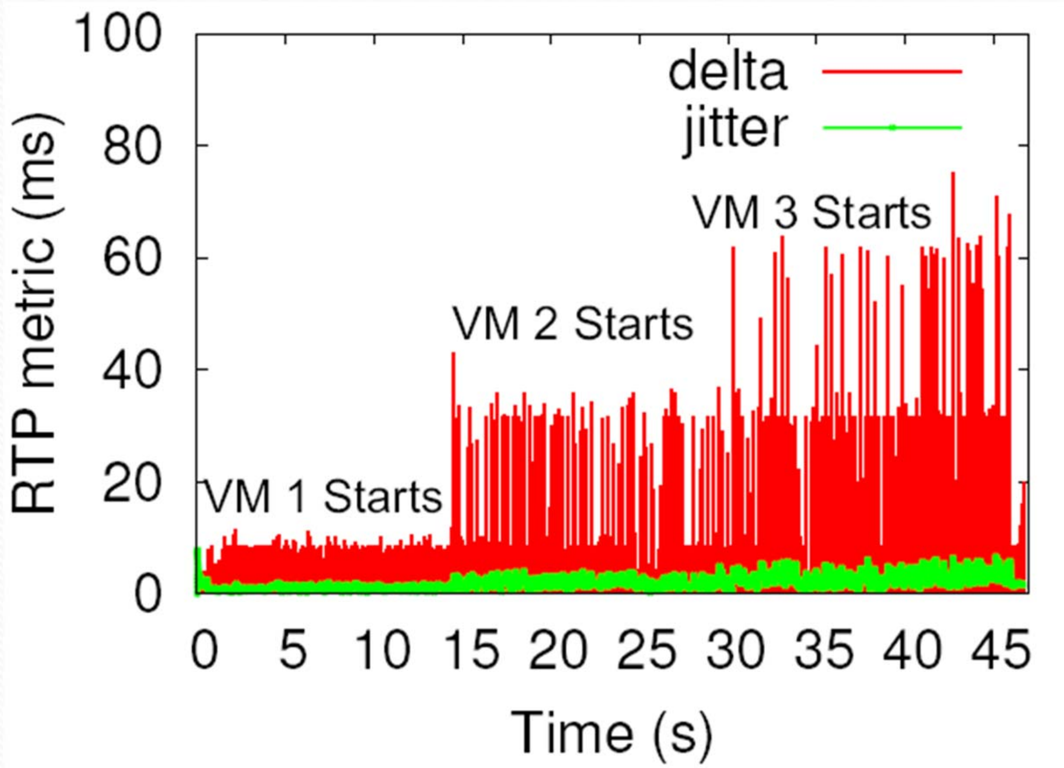


(c) our scheduler, deadline = 8ms



(d) our scheduler, deadline = 10ms

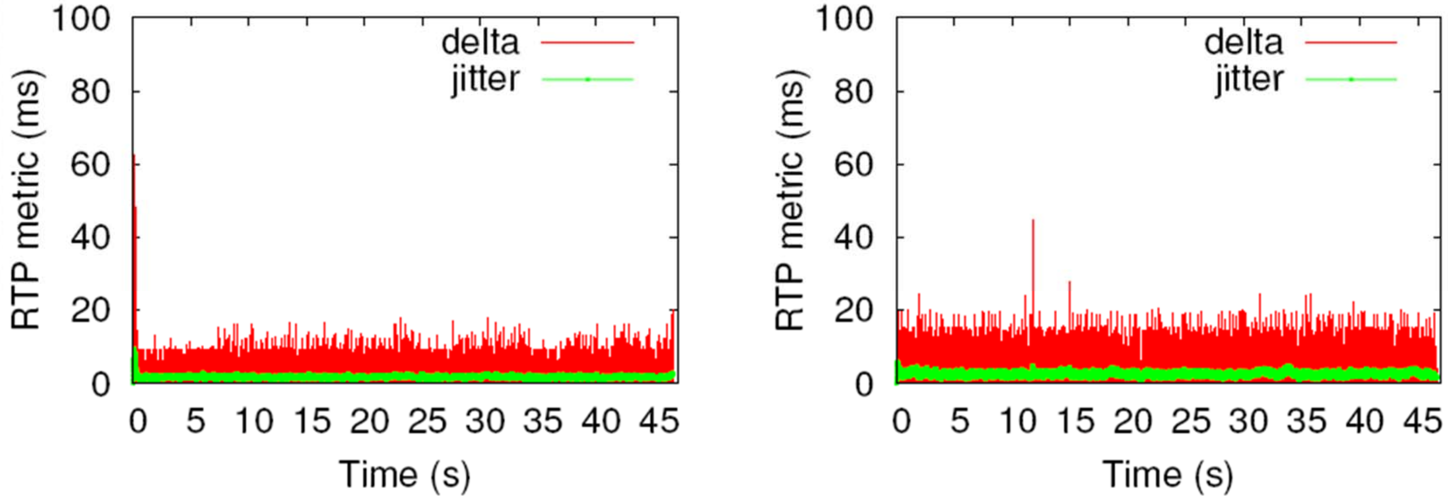
With Xen's Credit Scheduler



- When runs alone, VM1 consumes no more than 55% CPU time

- VM1, VM2 and VM3 co-locate on one CPU core
- VM1 → 60%; VM2 → 20%; VM3 → 20%.

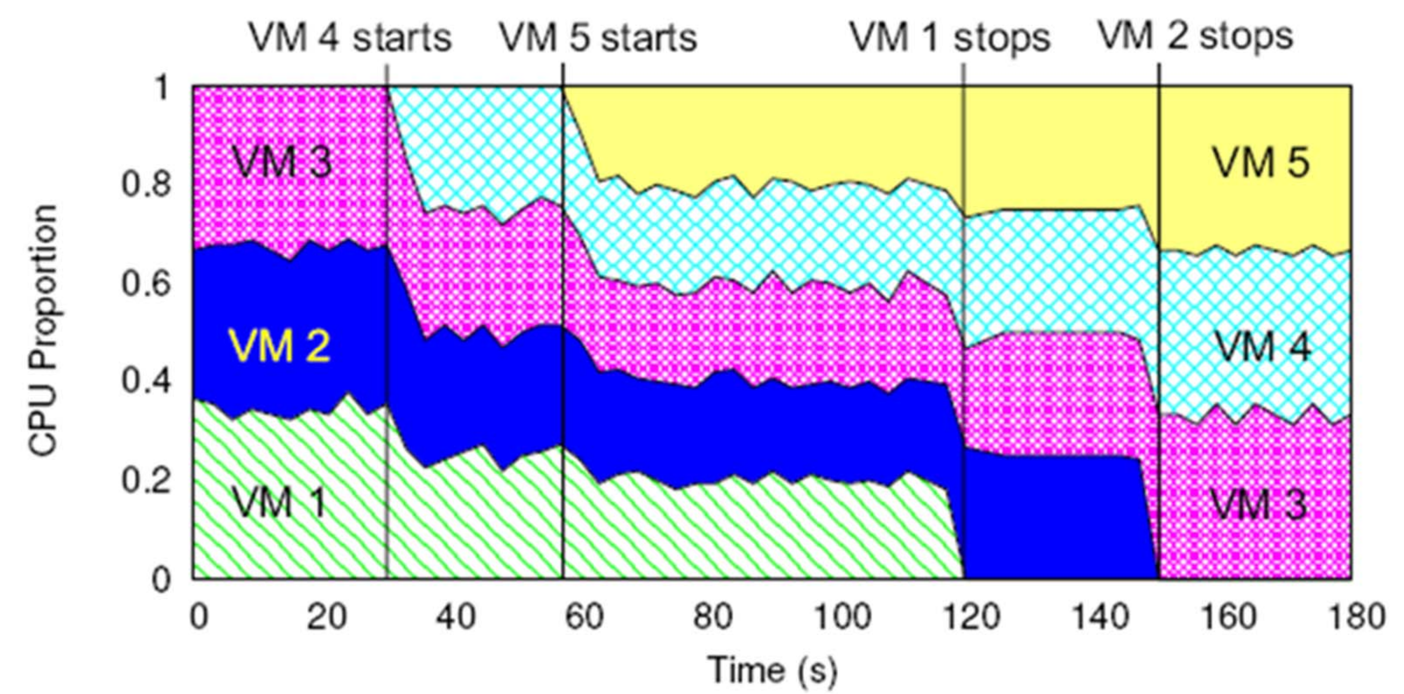
Our New CPU scheduler



(b) deadline=3ms (c) deadline=5ms

- VM1, VM2 and VM3 co-locate on one CPU core
- They run together all the time

CPU time proportional share

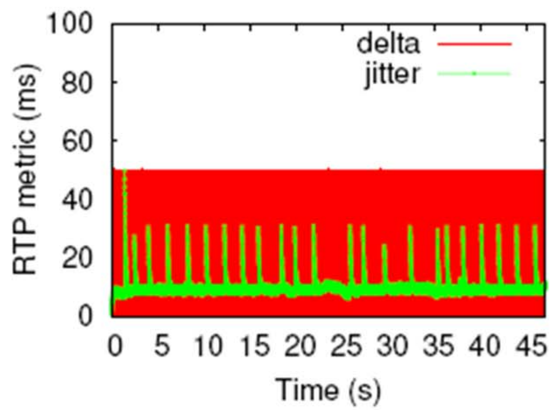


- Recorded by every three seconds

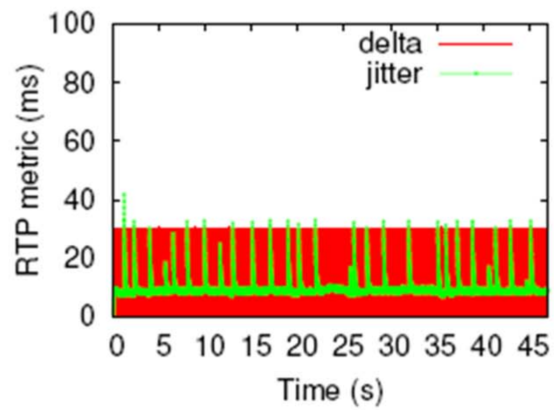
Network traffic shaper

- Evaluation goals:
  - The ability to reduce network jitter
  - The ability to maintain bandwidth allocation
- Benchmarks:
  - RTP video streaming, Apache web server
  - Netperf
- Two tunable parameters:
  - Smooth Window size (currently set at 3ms)
  - Window adjusting internal (currently set at 1 second)

Xen's rate limiting (RTP streaming)

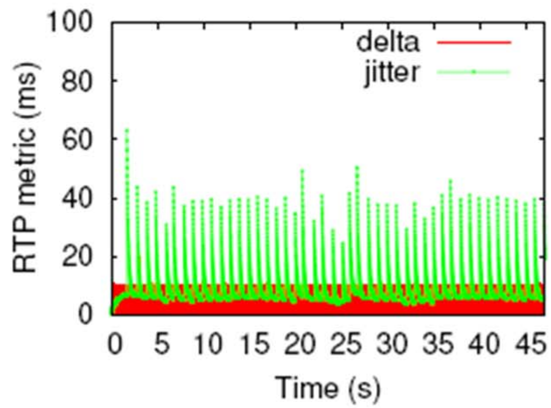


(a) interval=50ms (Xen's default setting)

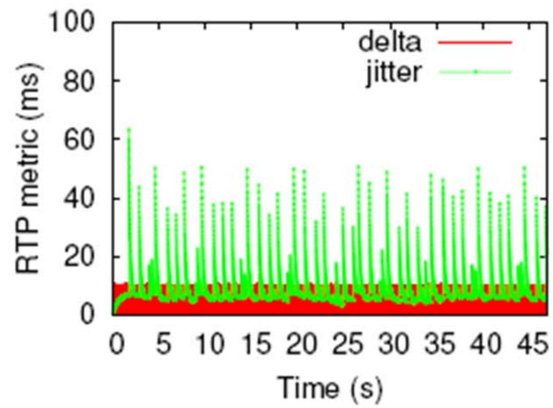


(b) interval=30ms

2Mbps

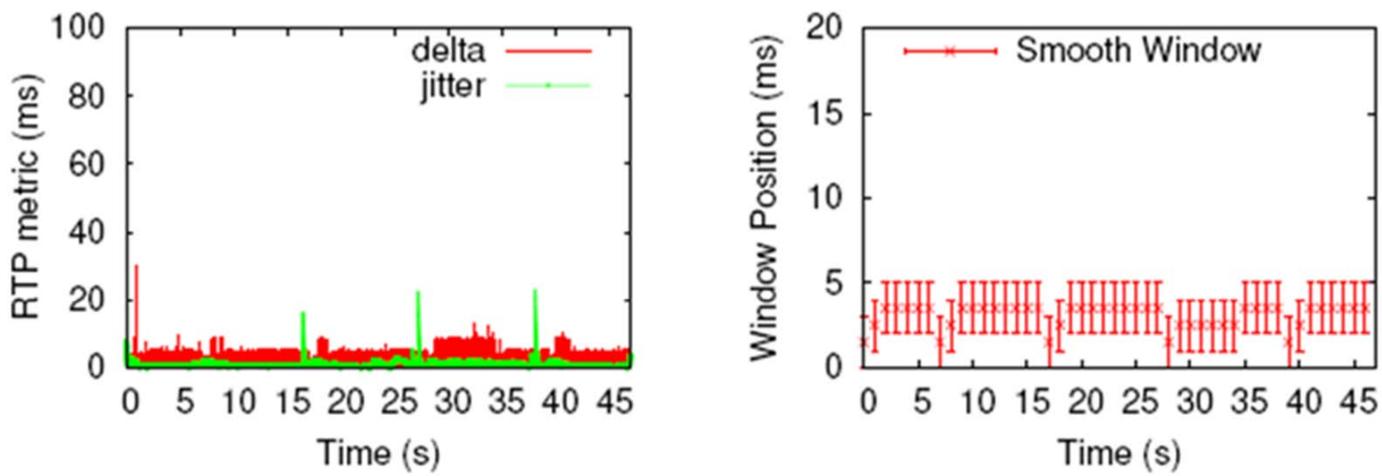


(c) interval=10ms



(d) interval=5ms

Our rate limiting (RTP streaming)

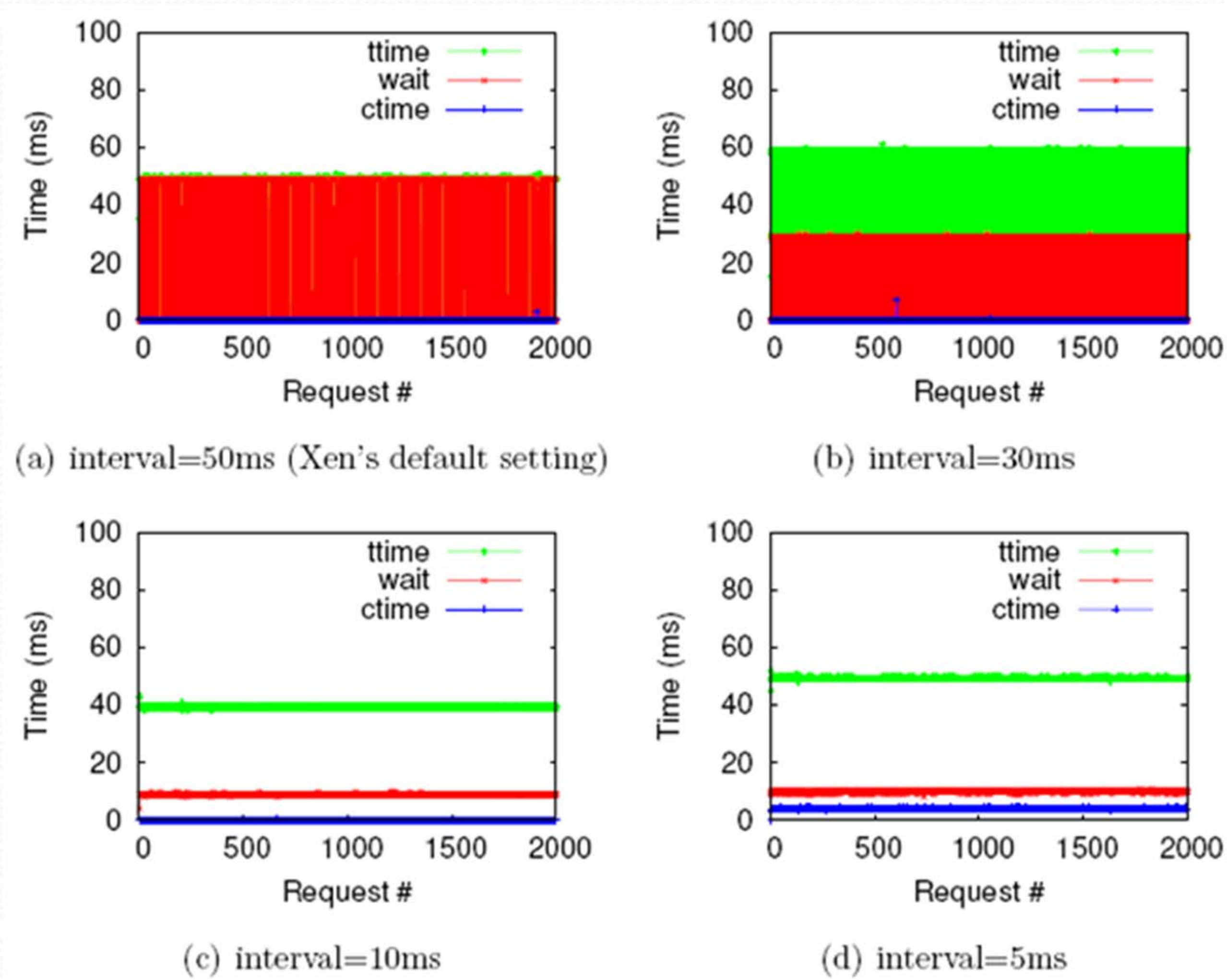


(a) after smoothing

(b) auto-adjusting of smooth window position

- Network jitter is greatly reduced
- Smooth Window position is automatically adjusted

Xen's rate limiting (ApacheBench)



(a) interval=50ms (Xen's default setting)

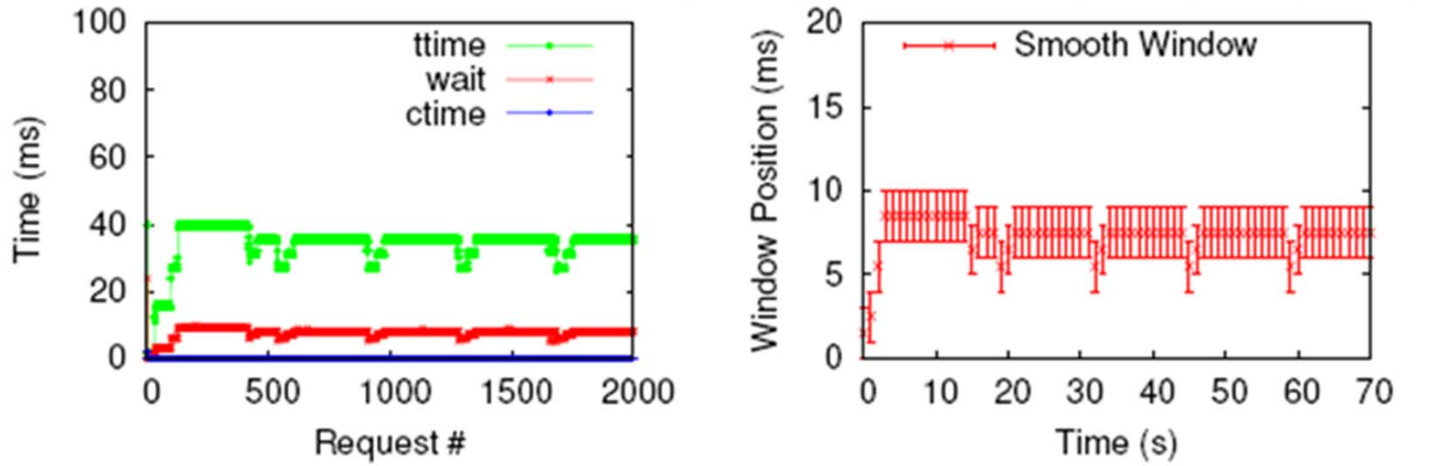
(b) interval=30ms

(c) interval=10ms

(d) interval=5ms

- 2Mbps
- 8KB file
- 2000 requests

Our rate limiting (ApacheBench)

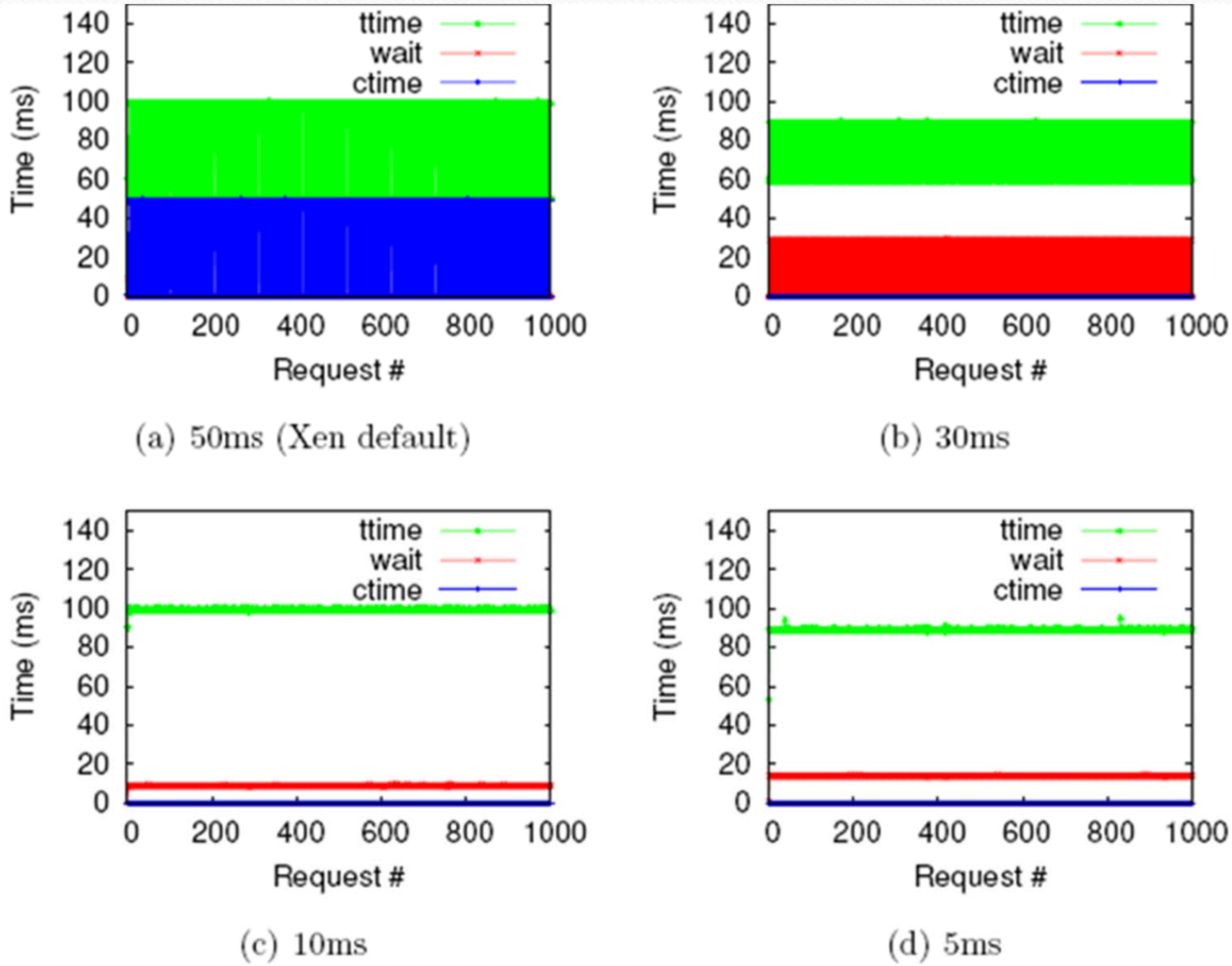


(a) after smoothing (b) auto-adjusting of smooth window position

- 2Mbps
- 8KB file
- 2000 requests

- Network jitter is greatly reduced
- Smooth Window position is automatically adjusted

Xen’s rate limiting (ApacheBench)



(a) 50ms (Xen default)

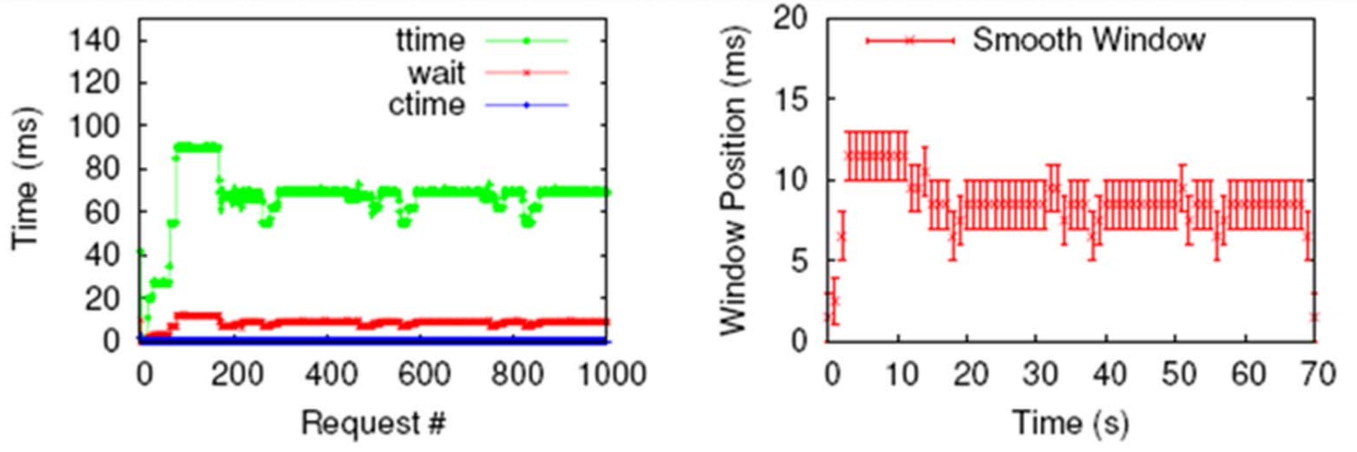
(b) 30ms

(c) 10ms

(d) 5ms

- 2Mbps
- 16KB file
- 1000 requests

Our rate limiting (ApacheBench)

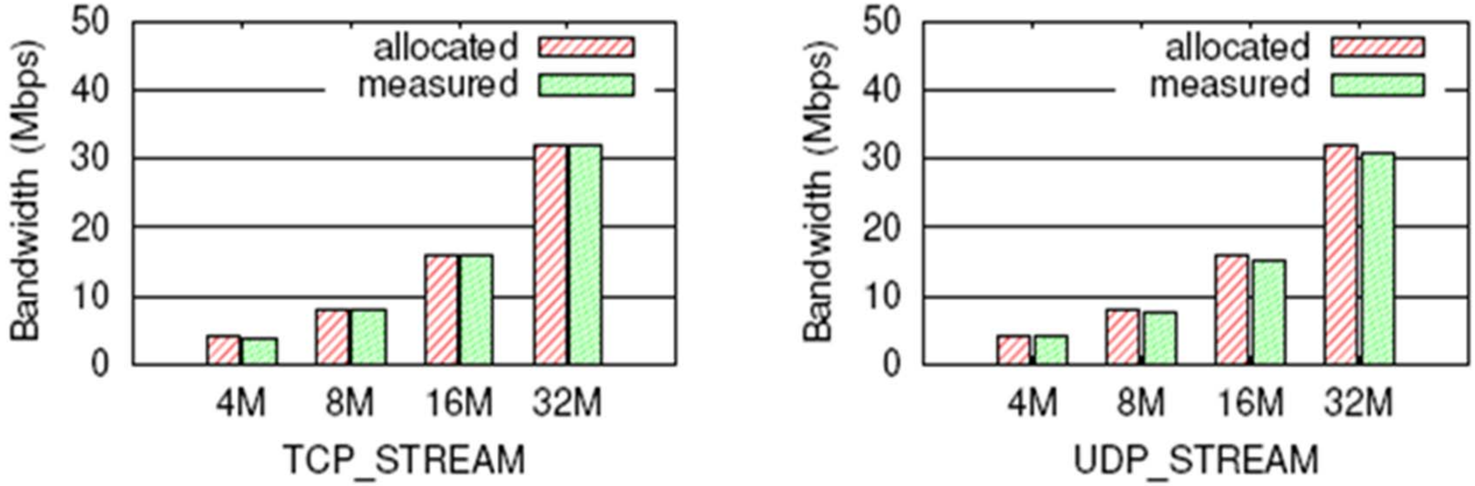


(a) after smoothing (b) smooth window position adjusting

- 2Mbps
- 16KB file
- 1000 requests

- Network jitter is greatly reduced
- Smooth Window position is automatically adjusted

Network bandwidth shaping

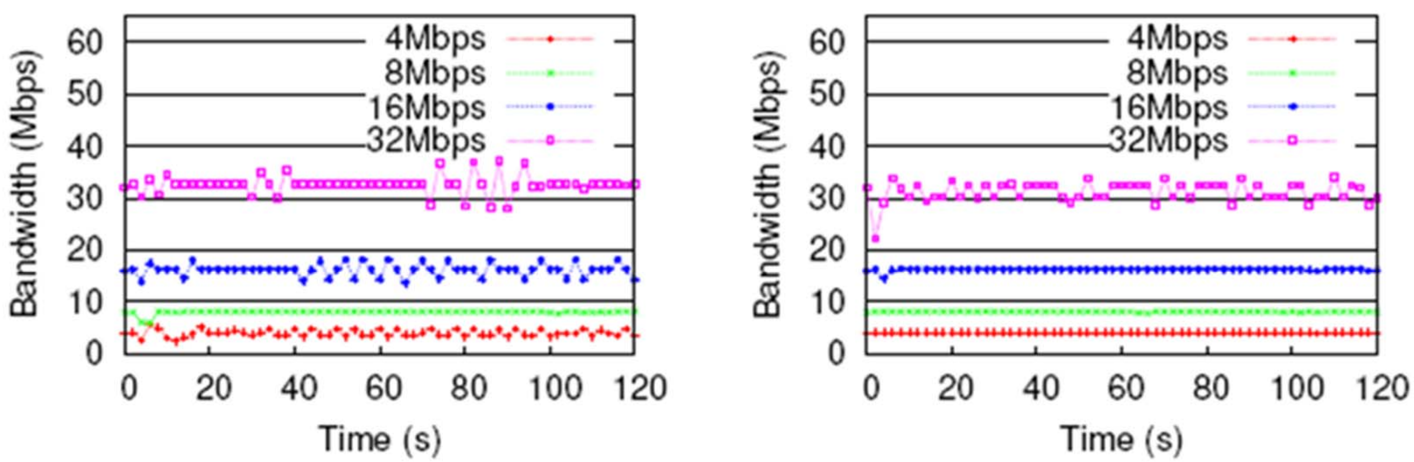


(a) TCP bandwidth test

(b) UDP bandwidth test

Macro-view of bandwidth shaping

Network bandwidth shaping



(a) TCP bandwidth test (b) UDP bandwidth test

Micro-view of bandwidth shaping (recorded by every 2 seconds)

Conclusion

- Problem:
  - How to mitigate network jitter in virtualized hosted platform, under the condition that resource proportional share is not affected

- Our Solution:
  - Real-time support in VMM CPU scheduler
  - Latency smoothing in Network traffic shaper

Thank you!

Q&A