Pricing Intra-Datacenter Networks with Over-Committed Bandwidth Guarantee

Jian Guo[1], **Fangming Liu**[1], Tao Wang[1], and John C.S. Lui[2]

[1]Cloud Datacenter & Green Computing/Communications Research Group

[1]Huazhong University of Science and Technology, China

[2]The Chinese University of Hong Kong

Talk outline

How to enable (quantified intra-DCN bandwidth performance + pricing) for VMs in IaaS clouds?

SoftBW: Motivation → Idea → Solution → Performance

Talk outline

1. Motivation

What kind of bandwidth performance & pricing do current clouds provide?

Performance metrics in IaaS clouds

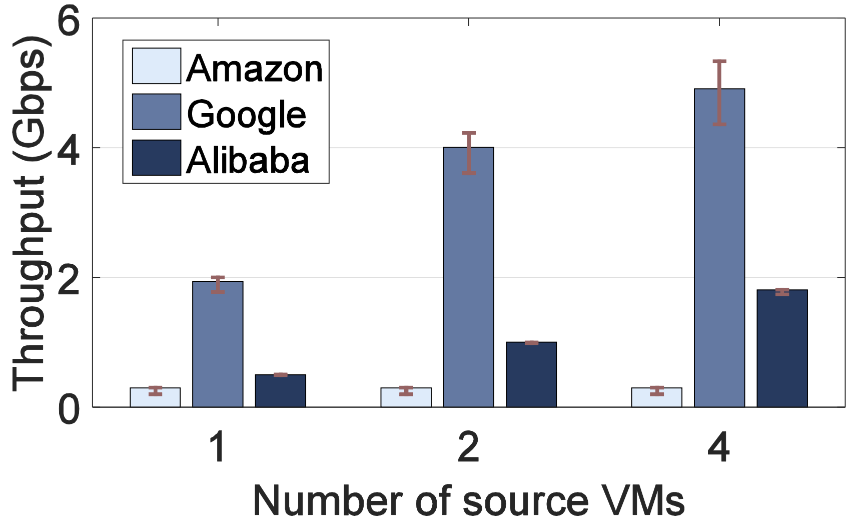
	amazon web services™	Google Cloud Platform	Alibaba Cloud
Performance	**vCPU, Memory, GPU, FPGA**	**vCPU, Memory, GPU**	**vCPU, Memory**
CPU	Number of cores, CPU model (Frequency, Architecture)	Number of cores, CPU model	Number of cores, CPU model
Memory	GB, DDR 3/4	GB	GB, DDR 3/4
Network	Low to moderate / Moderate / High	N/A	N/A

No quantitative network performance

No clear definition on VM bandwidth performance and pricing in today's top IaaS clouds

Measurements in clouds

- Different clouds
  - Same price: 16x difference in performance
- Different VMs
  - Performance: Cheap VM > Expensive VM
- Different time
  - Varying and highly unpredictable



Price-performance anomaly

Talk outline

2. Idea

Can we guarantee & price bandwidth performance?

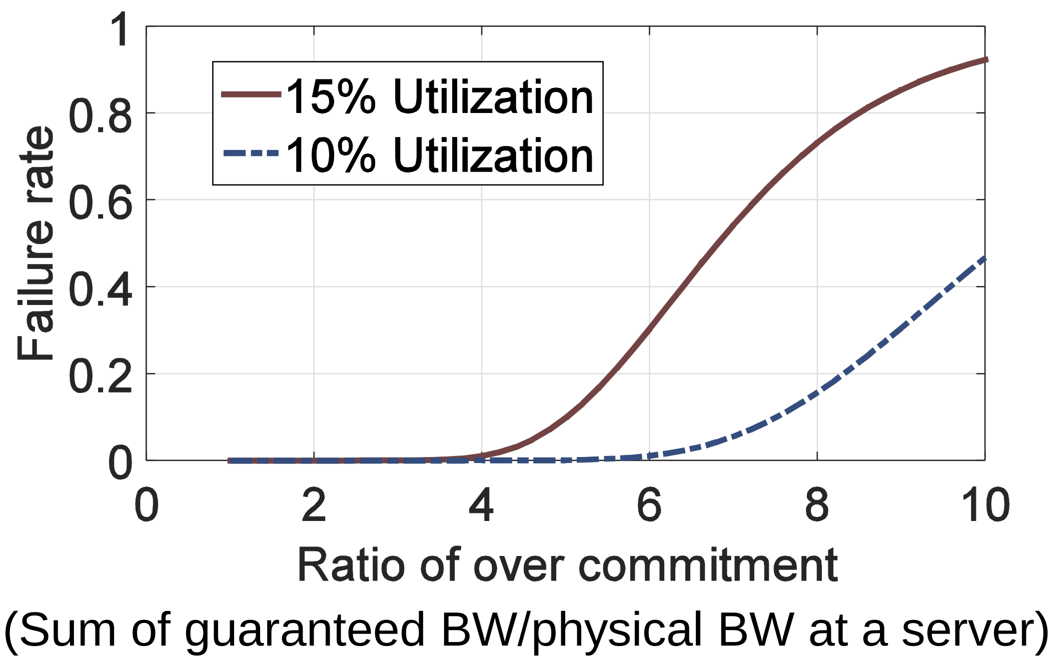
When considering performance & pricing…

- Benefit both providers & users
- Over-commitment (OC)
  - A economical and practical solution for bandwidth in cloud-scale DCN
- Why Rational
  - Opportunity: 99% links < 10% loaded in cloud-scale DCN (SIGCOMM’15)
  - Worst case performance guaranteed: N$\times$ OC means 1/N guarantee

Though, the bandwidth guarantee may fail…
how often tenants may not obtain their paid bandwidth?

Analysis on failure rate of bandwidth guarantee at cloud-scale

- Modeling with traffic trace (IMC'10)
  - n VMs, exponential distribution
  - $P_n = e^{-\alpha C} \sum_{i=1}^{n} \frac{(\alpha C)^{i-1}}{(i-1)!}$ $\quad \delta = C_B/C \quad \alpha = n/\rho C_B$
- Low failure rate (OC tolerance)
  - If utilization = 10%
  - Then, 4x over-commitment (OC): < 5% failure



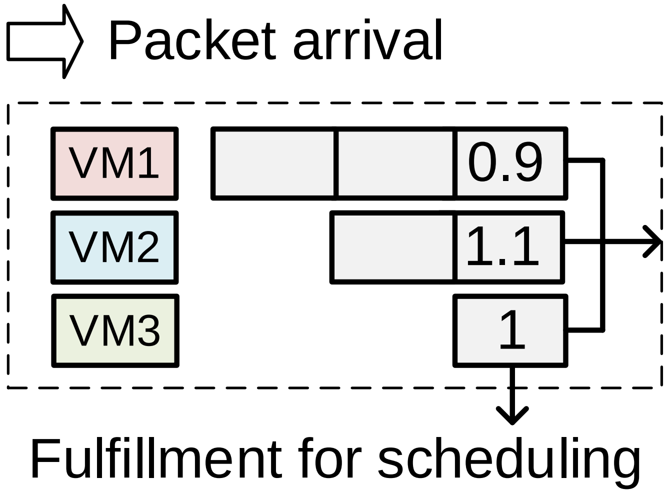
Providers: increase network utilization & revenue with OC

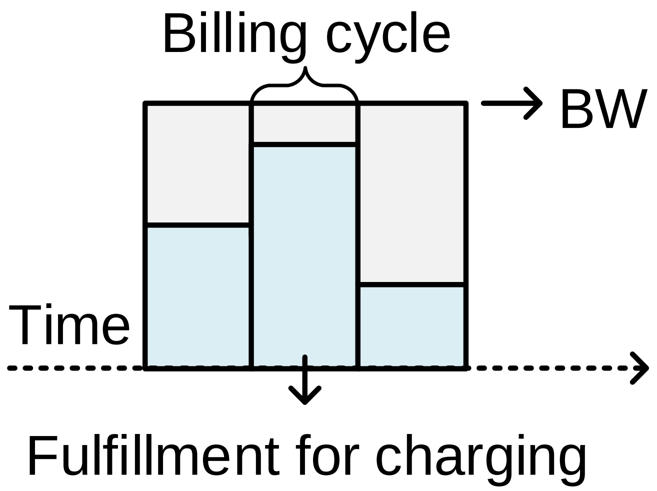
Tenants: how to guarantee their benefits (performance, fairness)?

Share bandwidth with over-commitment

- **Fulfillment** ratio: F= $rate/(bandwidth\ guarantee)$
- **Metric** for performance guarantee ➔ meet traffic demand while maintaining fairness of fulfillment
- **Quota** for pricing ➔ measure fulfillment per billing cycle

Abstraction➔





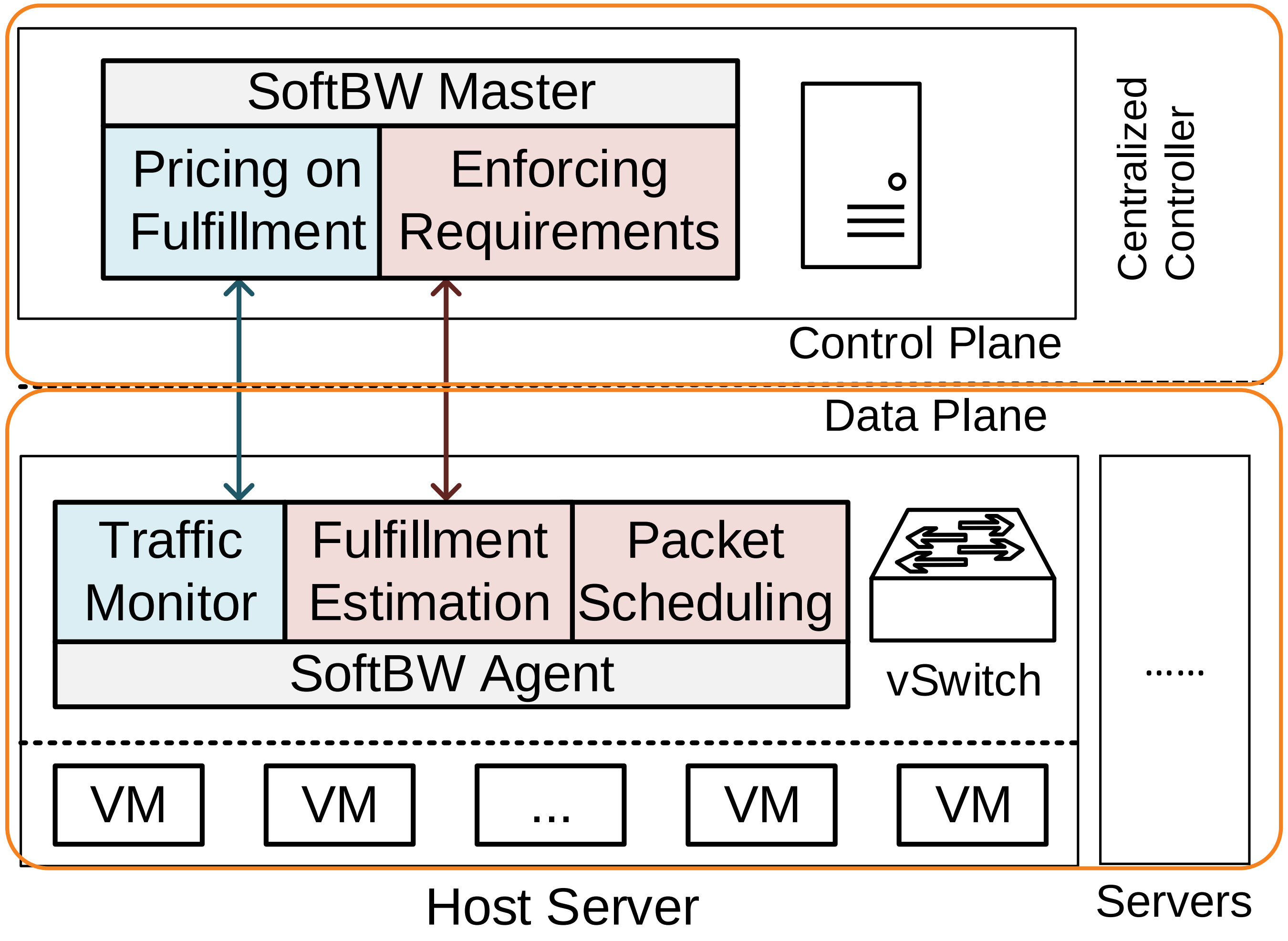
Co-design: price-performance consistency

Talk outline

3. Solution: **SoftBW**

How to achieve bandwidth guarantee and usage-based pricing under over-commitment?

SoftBW overview



Design goals

- Price-performance consistency
- Over commitment tolerance
- Short flow friendly

Data plane

- Scheduling: bandwidth guarantee with OC
- Work-conserving

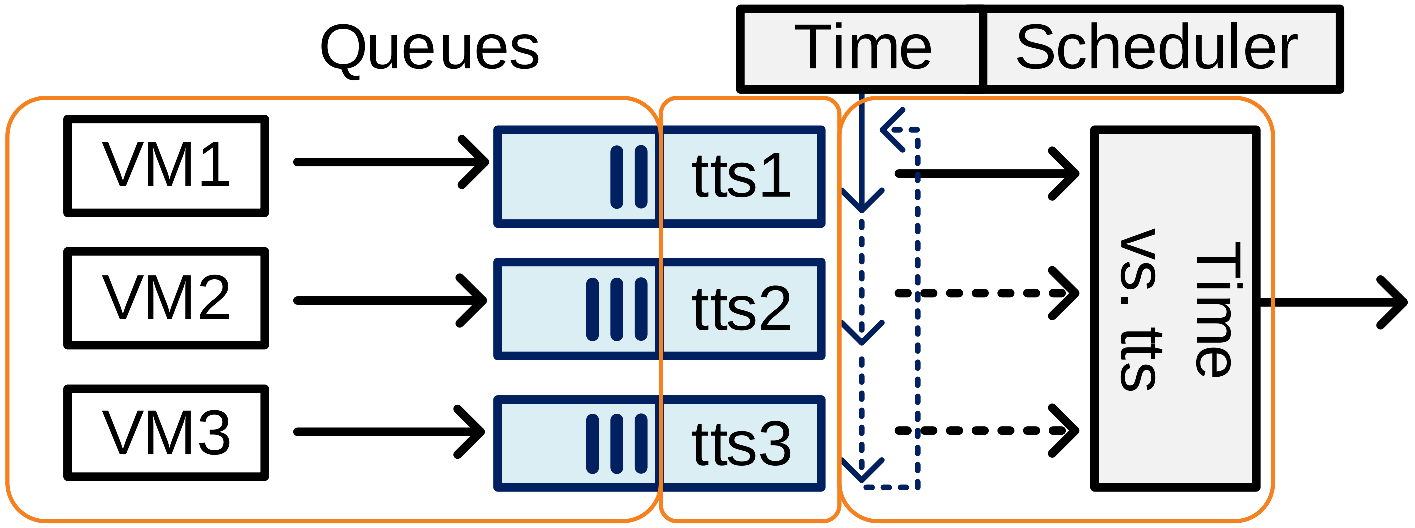
Control plane

- Pricing: usage-based charging

Enable provider to take advantage of low network utilization to oversell bandwidth

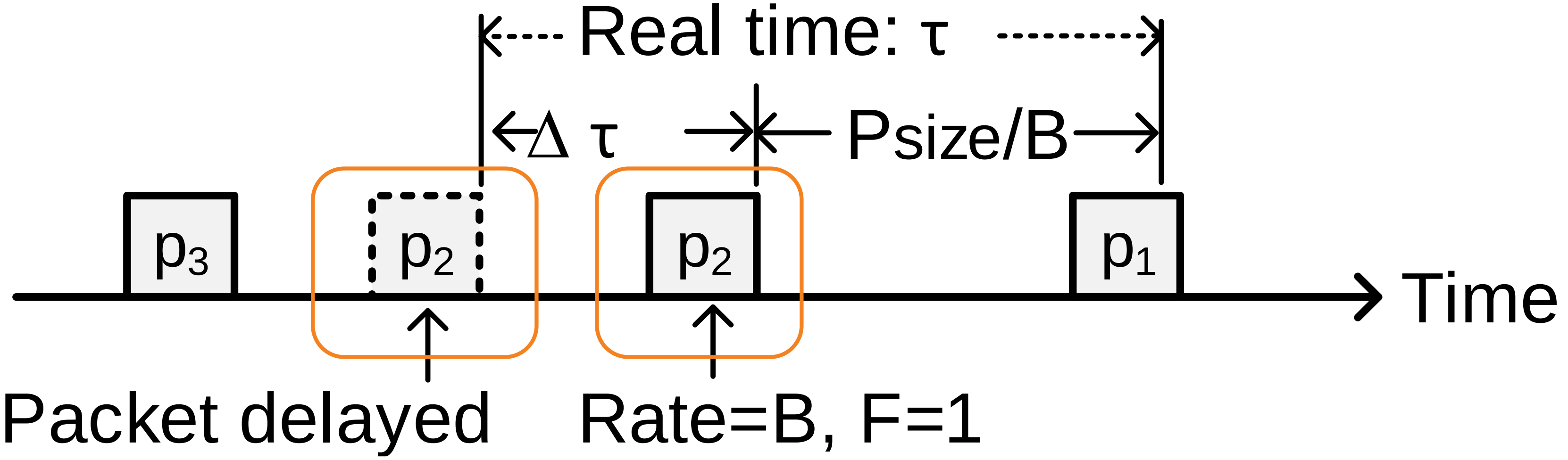
Fulfillment-based scheduling

- Round-robin for each VM queue
- Estimation of fulfillment: F> or <1
- Scheduling of packets: current time vs. time to send (tts)



Estimation of fulfillment

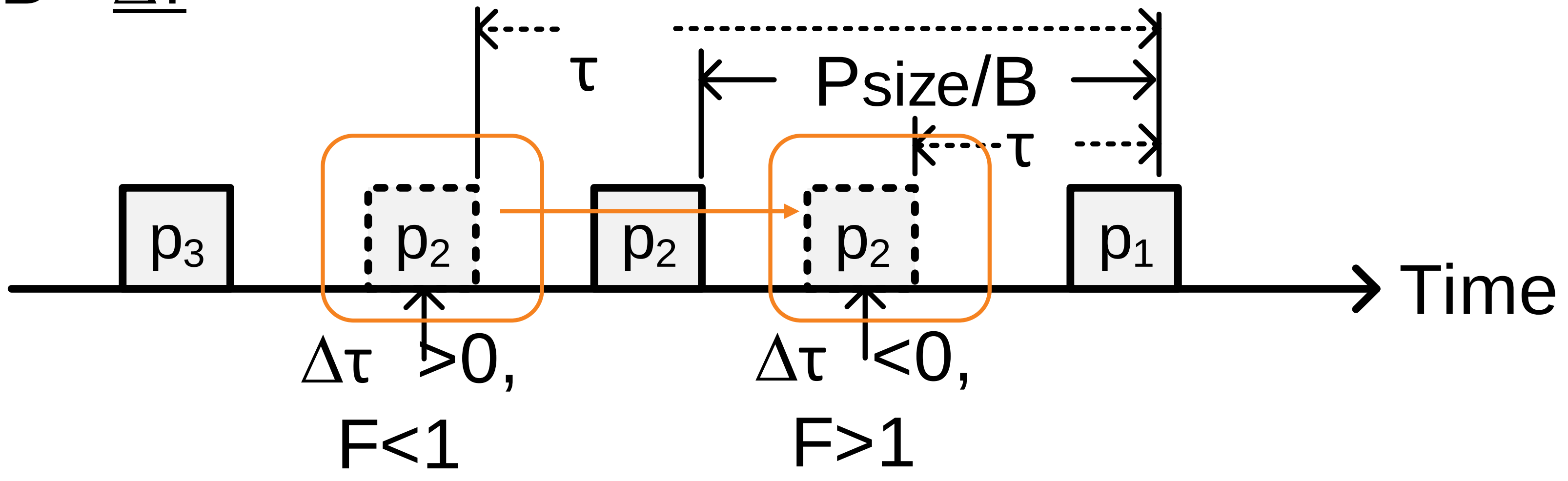
- Update: after transmitting each packet
- Fulfillment: F=rate/B, rate=packet size/time
- Expected transmission time: P_{size}/B
- Inter-department time: F<1→ (delayed) $\Delta\tau > 0$; F>1→ $\Delta\tau = \tau - P_{size}/B < 0$



Estimation of fulfillment

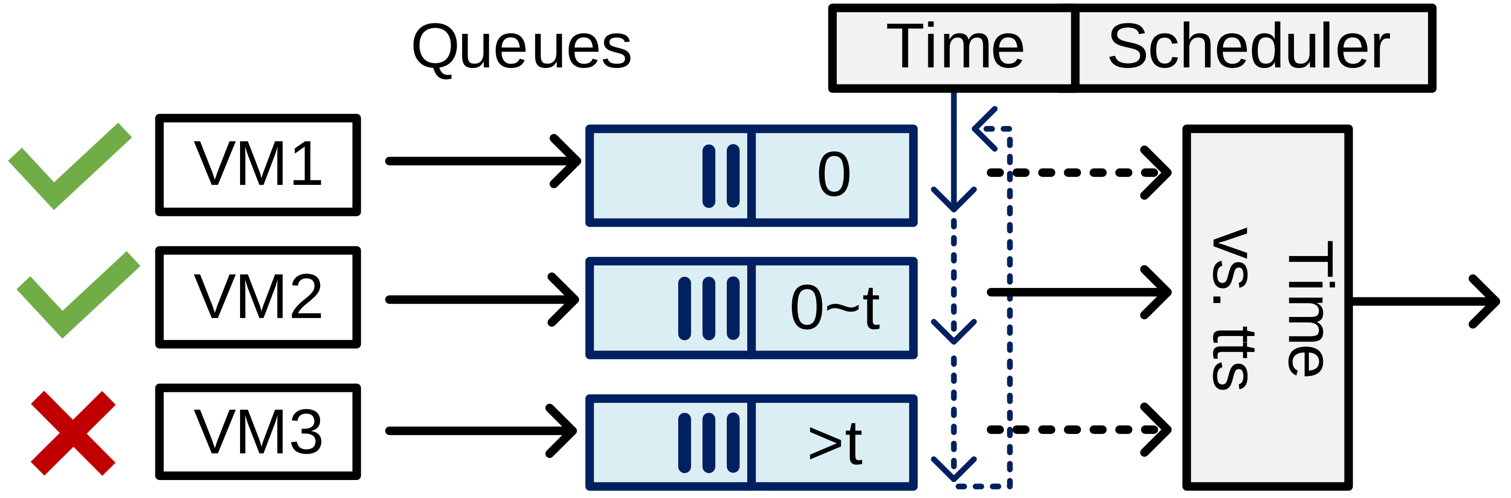
Maintain (update): $\underline{\Delta\tau} \leftarrow \underline{\Delta\tau} + \Delta\tau$, as accumulation from many packets

- $\underline{\Delta\tau} \geq 0$: bandwidth guarantee not satisfied, tts = 0: allow to send
- $\underline{\Delta\tau} < 0$: rate exceeds bandwidth guarantee, tts = time + P/B
- $\underline{\Delta\tau}$ from positive to negative: from unsatisfied to satisfied, tts = time + P/B - $\underline{\Delta\tau}$



Scheduling of packets

- tts = 0: not satisfied, allow to send
- 0 < tts < current time: missed the expected transmission time
- tts > current time: send if there is residual bandwidth



Pricing model & Performance metrics

(Open to customized designs, based on general rules)

- Differentiated performance metrics: different applications

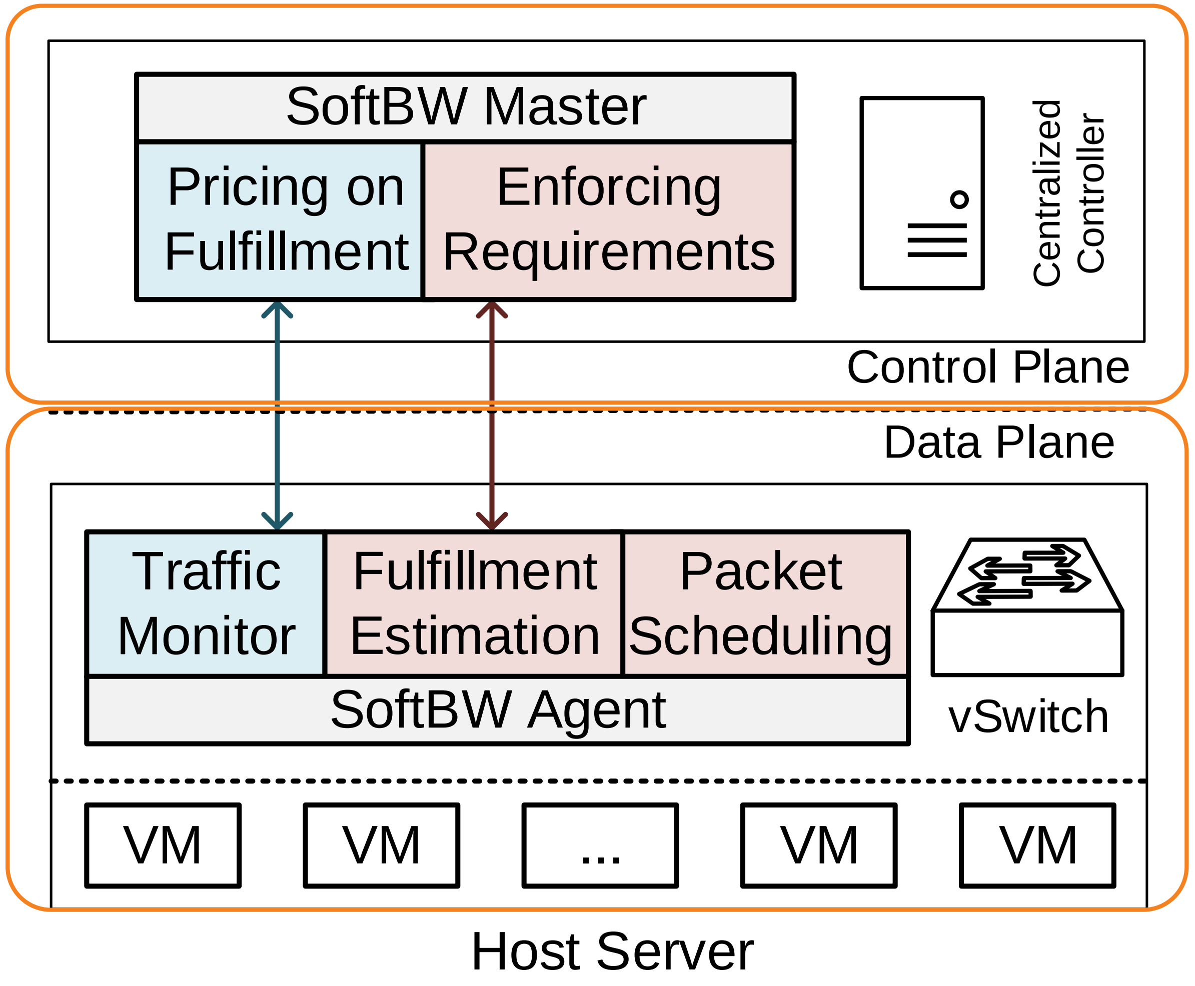
e.g., real-time jobs, deadline jobs, delay-tolerant background backup

- Usage-based charging: performance-price consistency
- Non-decreasing pricing function: true requirement declaration

Guarantee	**Performance**	**Price** (e.g.)
Strict	Bandwidth B (< physical bandwidth C)	r*(1+B/C)P0
Dynamic	Data (traffic) size S, deadline time T	r*(1+S/TC)P1
Fairness	VM-level fairness	r*(r/C)P1
Best effort	No bandwidth guarantee	Free

- *e.g., for strict, if actual rate $r > B$, then $(r-B)$ is charged as fair share*
- *May fail under OC but pay less (price-performance consistency): see example in page 24*
- *P0 is unit price, $P1 = \beta P0$, $\beta<1$ to encourage tenants to use dynamic guarantee for reducing guarantee failure*

SoftBW implementation



A SDN based solution

Pricing

- Control plane application
- Centralized control
- Opendaylight platform

Scheduling (bandwidth allocation)

- Data plane function
- Distributed scheduling on each server
- Open vSwitch

Talk outline

4. Evaluation: questions

- How efficiently does SoftBW allocate bandwidth under over-commitment?
- What is the impact of over-commitment on the network utilization and guarantee failure?

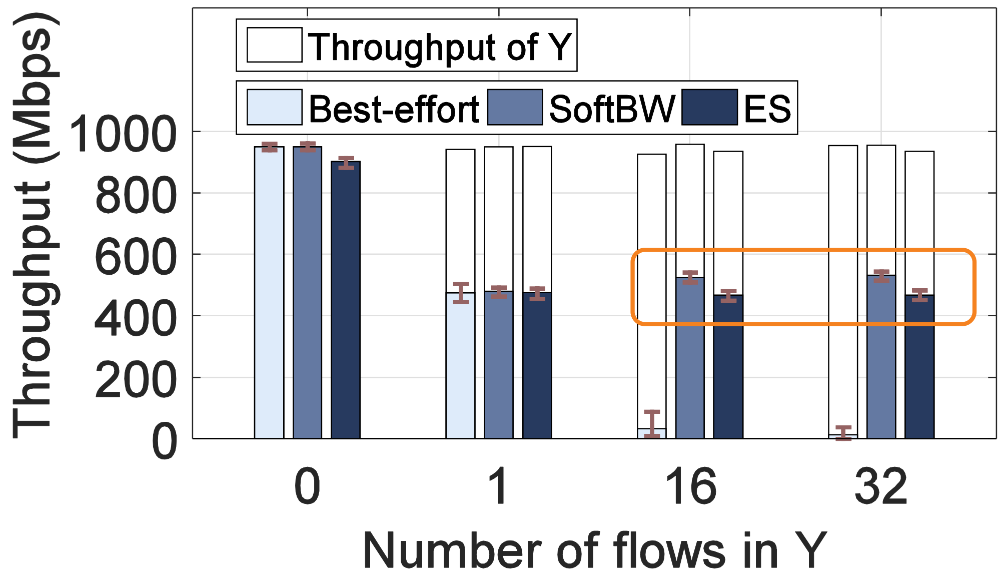
Evaluation setup

- Comparison: rate-limit based bandwidth allocation
  - ElasticSwitch, Best effort (no performance guarantee)
- Testbed: performance of SoftBW
  - 14 servers, KVM and Open vSwitch, 1Gbps NIC
- Simulator: impact of over-commitment
  - 2,000 servers, each server with 4 VMs, 1s interval
- Traffic trace:
  - Based on the distribution in existing measurement work (IMC’11, SIGCOMM’15)

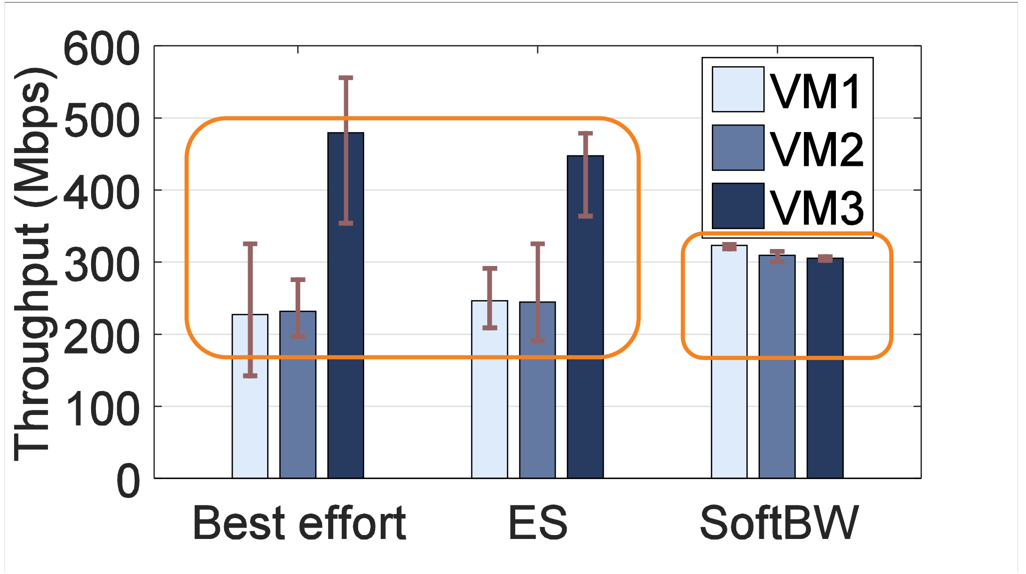
Performance: bandwidth allocation

- SoftBW guarantees fairness under over-commitment

2 co-located VMs: each with 450 Mbps BW guarantee
physical bandwidth: 1 Gpbs



3 co-located VMs: each 450 Mbps BW guarantee
physical bandwidth: 1 Gpbs

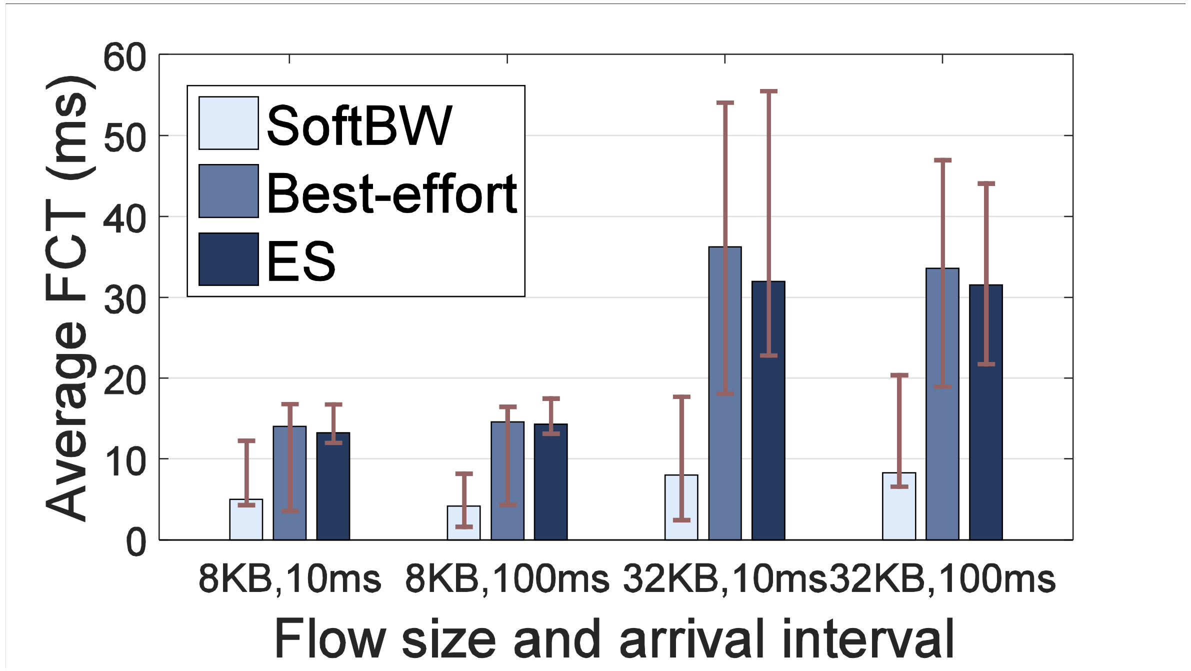


Sufficient bandwidth: achieve bandwidth guarantee

Not enough bandwidth: ES, best effort fail to achieve fairness

Performance: short flows

- Quickly obtain the bandwidth: improve short flow performance (completion time) by 2.8x - 4.5x

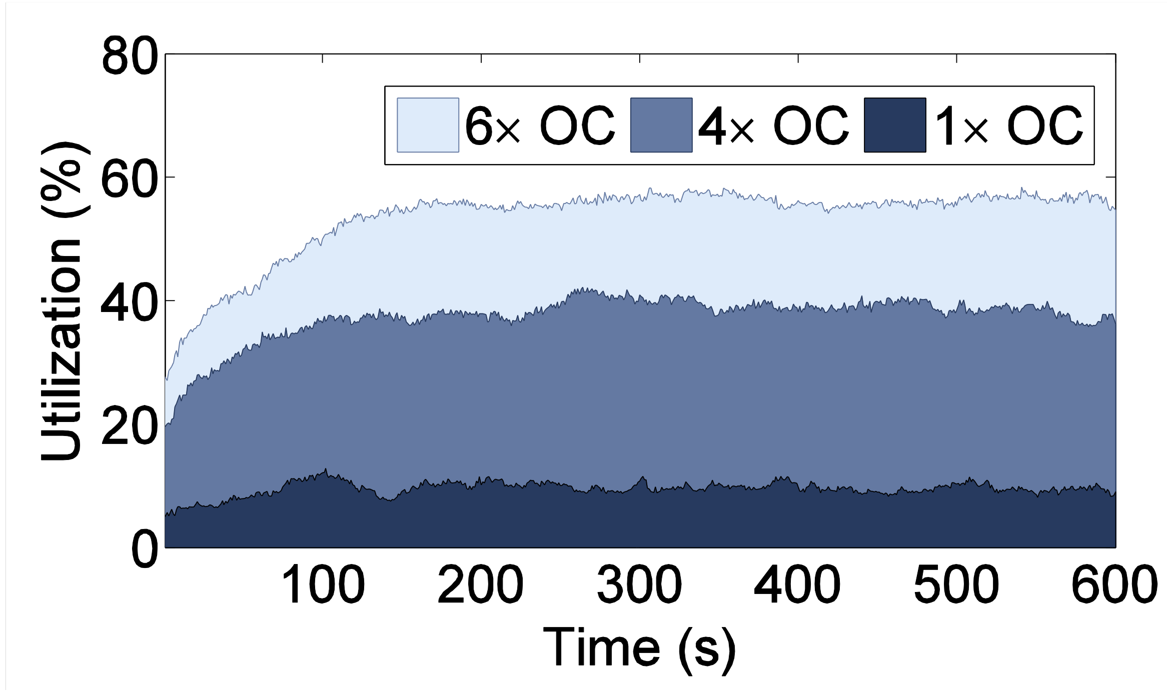


ES: completion time less than the update interval (e.g. 50ms) of rate-limit based solution ➔ **hurt short flows**

SoftBW: packets in a queue can be scheduled in each round
➔ **short flows friendly**

Over-commitment: utilization

- Improve about 3.9x bandwidth utilization with 4x OC



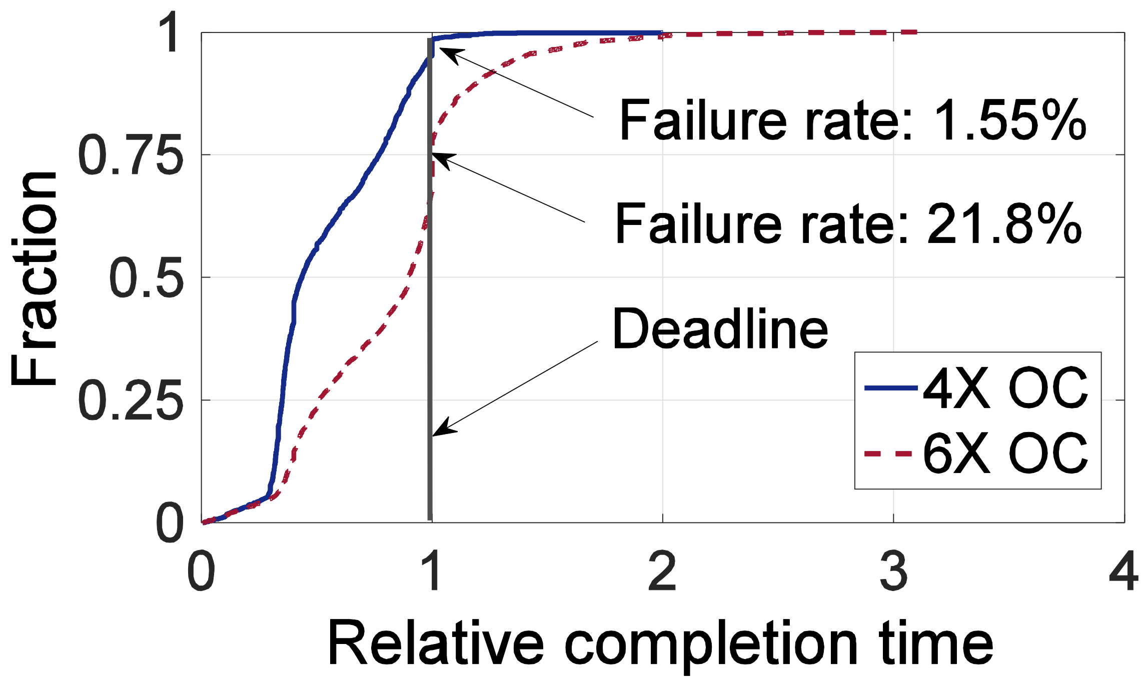
Overall bandwidth utilization

Traffic without OC: 9.5% bandwidth utilization

4x OC: 37.4% utilization

Over-commitment: failure rate

- Dynamic guarantee: only 1.55% failure rate with 4x OC



98.4% finish in time

99.9% less than 1.4x deadline

Worst performance 2x deadline

Summary: *A feasibility study for…*

How to enable (quantified bandwidth performance + pricing) for cloud VMs ➔ Beneficial for both providers and tenants

SoftBW:

1. Use over commitment
2. Scheduling + pricing = price-performance consistency

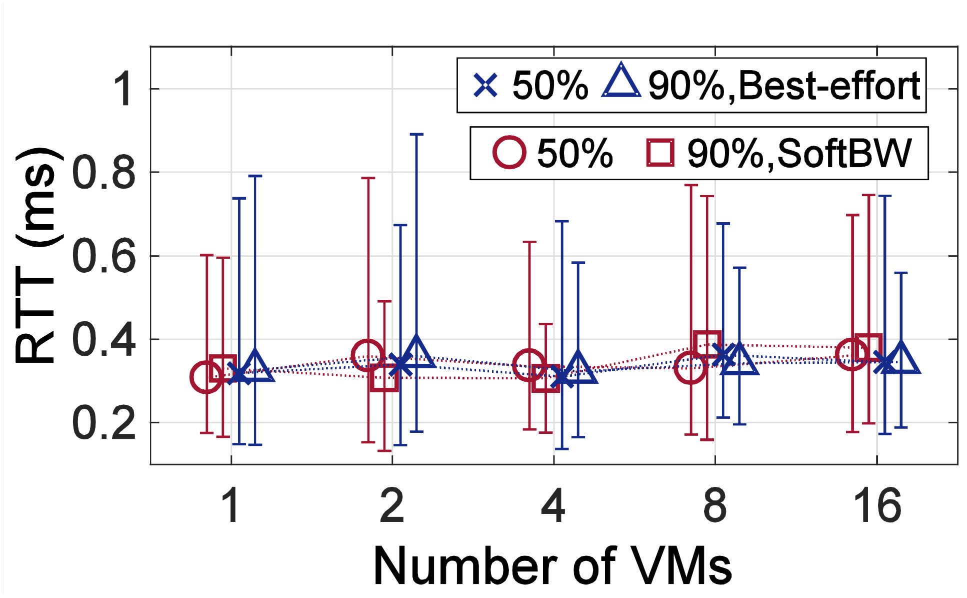
Thank you!

Cloud Datacenter & Green Computing/Communications research group
Huazhong University of Science & Technology

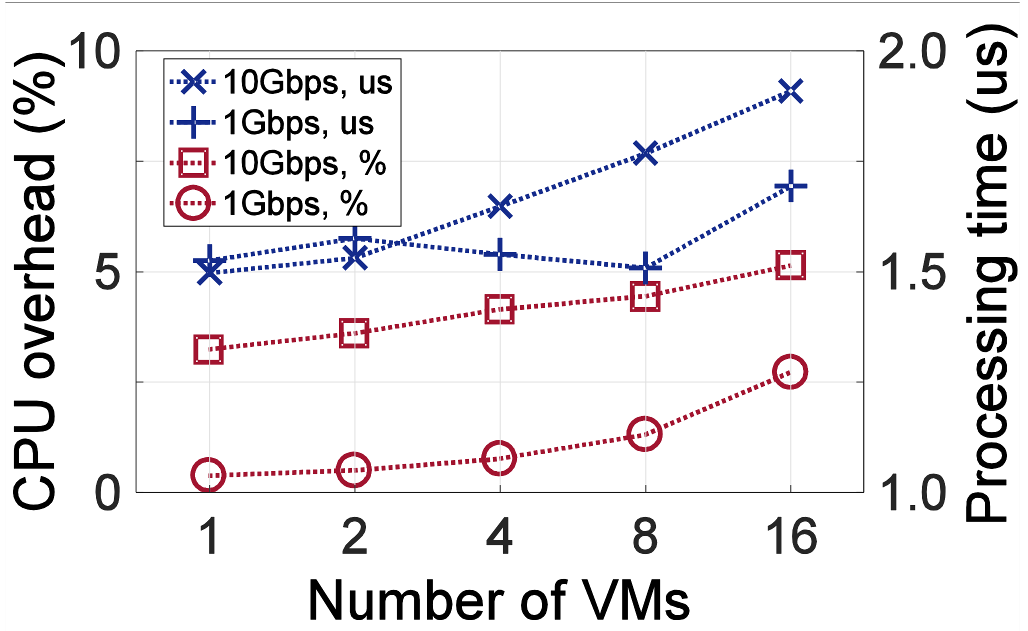
Fangming Liu: http://grid.hust.edu.cn/fmliu/

Performance: overhead

- SoftBW involves very little overhead



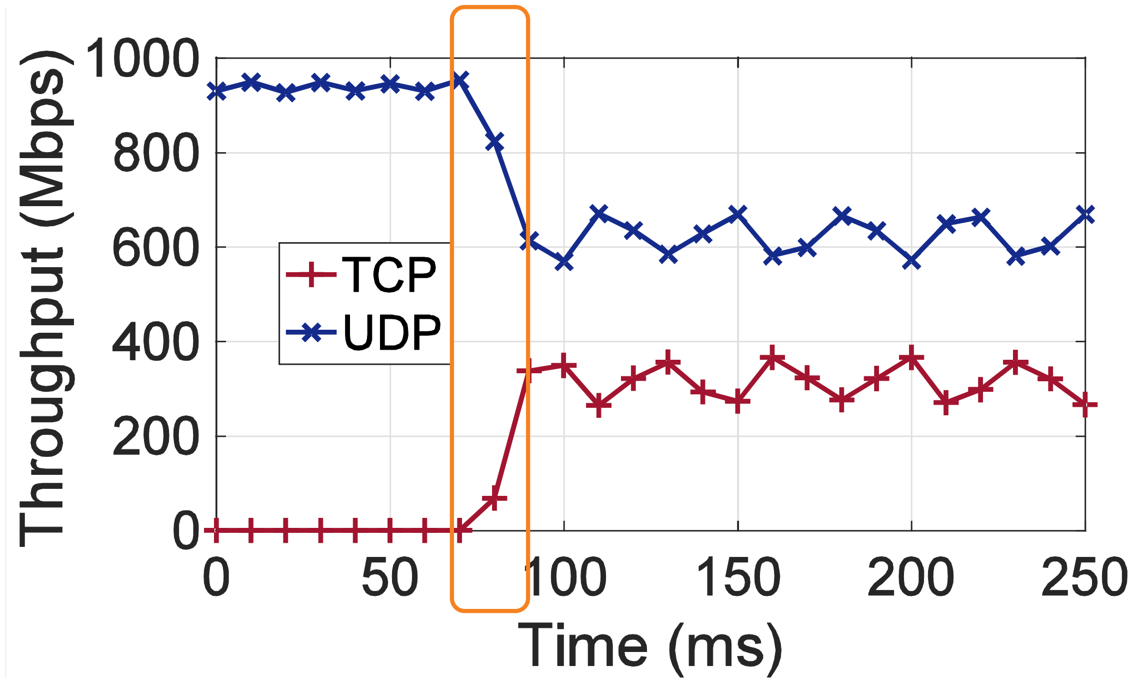
Latency: no obvious increase as compared with 350 μs RTT



CPU: 10Gbps transmission at MTU packet size costs 5.1%

Performance: fast convergence

- SoftBW converges in about 20 milliseconds

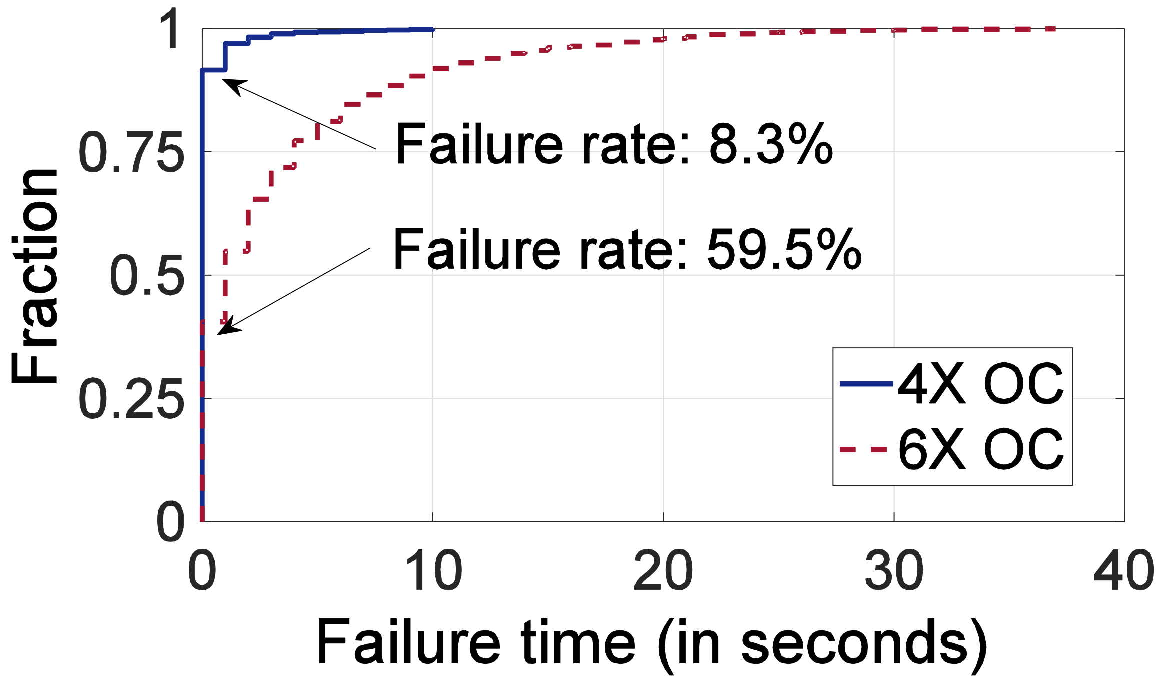


TCP vs. UDP: the rate of VMs

Throughput is stable

Over-commitment: failure rate

- Strict guarantee: only 8.3% failure rate with 4x OC

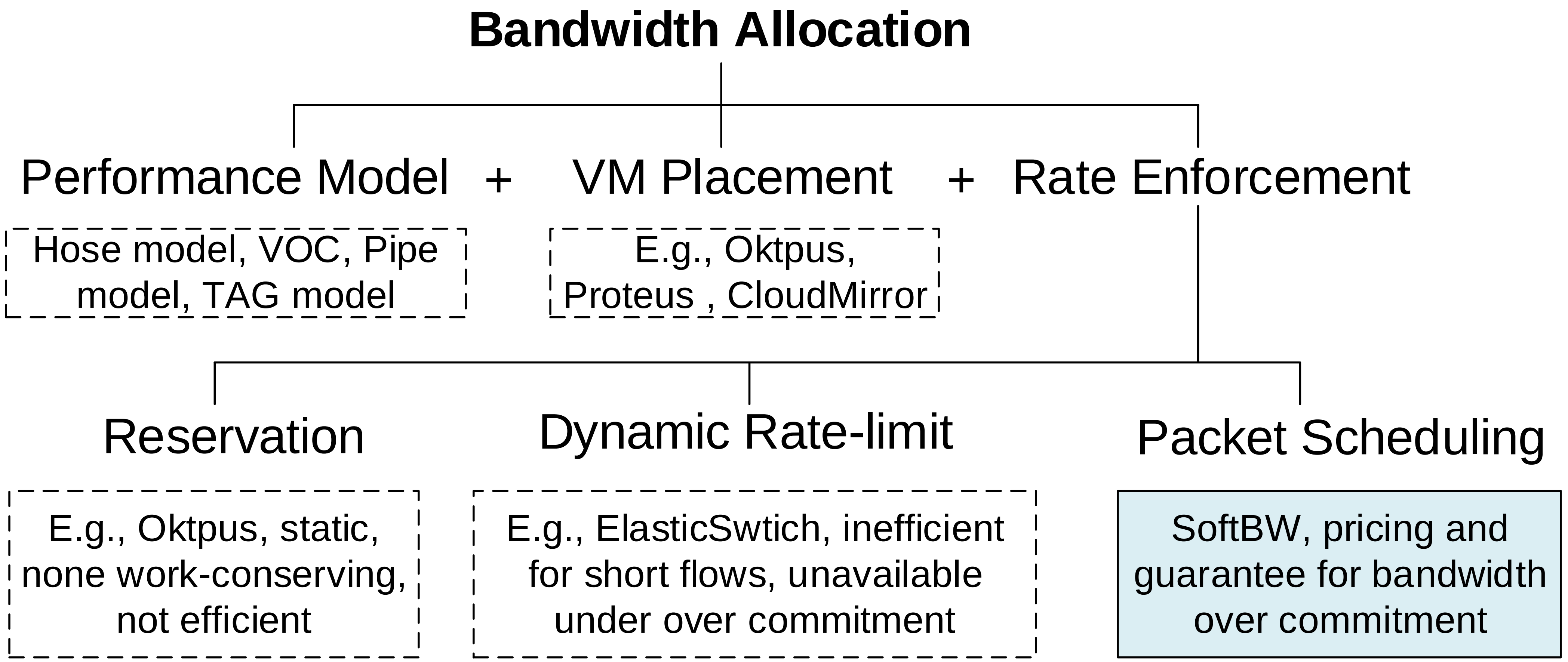


91.7% no failure

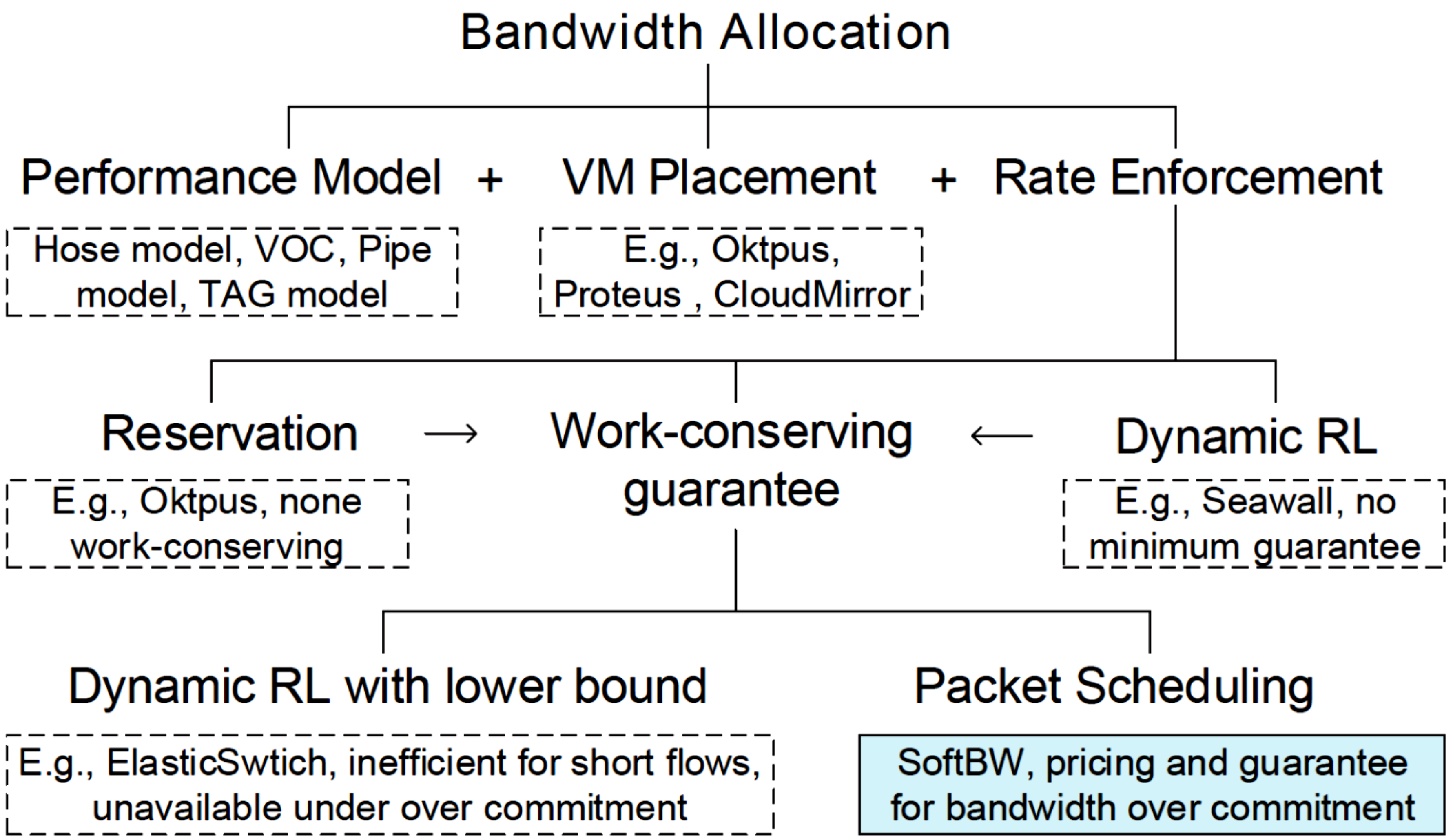
99.9% less than 5s failure (total simulation time: 600s)

Worst performance 10s failure

Summary: position of SoftBW in the literature


Bandwidth Allocation
Performance Model
+
VM Placement
+
Rate Enforcement
Hose model, VOC, Pipe model, TAG model
E.g., Oktpus, Proteus , CloudMirror
Reservation
Dynamic Rate-limit
Packet Scheduling
E.g., Oktpus, static, none work-conserving, not efficient
E.g., ElasticSwtich, inefficient for short flows, unavailable under over commitment
SoftBW, pricing and guarantee for bandwidth over commitment

Summary: position of SoftBW


Bandwidth Allocation
Performance Model + VM Placement + Rate Enforcement
Hose model, VOC, Pipe model, TAG model
E.g., Oktpus, Proteus , CloudMirror
Reservation → Work-conserving guarantee ← Dynamic RL
E.g., Oktpus, none work-conserving
E.g., Seawall, no minimum guarantee
Dynamic RL with lower bound
Packet Scheduling
E.g., ElasticSwtich, inefficient for short flows, unavailable under over commitment
SoftBW, pricing and guarantee for bandwidth over commitment

Questions

- Q: The novelty of our paper as compared to existing work. The contribution of this work.
- A: We focus on addressing price-performance anomaly and over commitment of bandwidth guarantee. We have our own contributions. First, existing work does not consider these two goals. Mostly, they focus on the efficiency of bandwidth allocation (also, they assume not over-subscribed access BW). Second, as shown in our experiments, existing rate-limit based solution in data center bandwidth allocation cannot work well for fairness and short flows, under over commitment. Third, they do not provide a pricing strategy (via a coherent fulfillment metric to co-design scheduling & pricing) to guarantee price-performance consistency.

Assumptions & focus

- Assumes the datacenter fabric to be a non-blocking switch [10, 11, 7], and our main focus is to schedule the traffic at the virtual ports connected to VMs

- Our bandwidth allocation focuses on end-based rate control
  - The choice comes from the fact that today's datacenters see rapid advances in achieving full bisection bandwidth [8, 9], and the providers have a growing concern about the over committed access bandwidth on each server rather than the aggregation and core level.
  - By leveraging the software virtual switch at each server, the cost of implementation can be reduced and the scale of rate control is limited to the number of VMs on each server.

Questions

- Q1: Can you explain “true requirement”?
- Q2: Why you use non-decreasing pricing function?
- A: For example, when transmitting 1 Gb data, using 100 Mbps bandwidth will cost 10 seconds, while using 200 Mbps bandwidth only costs 5 seconds. Both situations cost 1000P, where P denotes the price of using 1 Mbps for 1 seconds. Hence, to keep performance-price consistency, the unit price of higher bandwidth guarantee should also be higher. In this way, tenants cannot declare higher bandwidth than their requirements to achieve higher performance under the same price. We call this “true requirement”. That’s why we use non-decreasing pricing function.

Questions

- Q: how bandwidth will be charged when failed or over-fulfilled, for example, exceeding the strict bandwidth guarantee or missing the deadline?
- A: For strict guarantee, traffic that exceeds bandwidth guarantee will be charged the same as the pricing of fairness guarantee, since it only gets a fair sharing.
- For dynamic guarantee, traffic that exceeding the deadline will not be charged, since the provider does not realize their SLA.
- All these strategies are used to guarantee price-performance consistency.

Questions

- Q: How you realize dynamic guarantee? Why it is called dynamic guarantee?
- A: For dynamic guarantee, the underlying implementation is similar with strict guarantee. But we update the guaranteed bandwidth in each billing cycle, by dividing the residual data S with residual transmission time t. The guarantee is dynamically adjusted according to the available bandwidth. If there is residual bandwidth on the server, the VM can utilize it and reduce the guarantee in the next update. As a result, the total bandwidth guarantee on a server is reduced, and the probability of guarantee failure also decreases. If the bandwidth is not guaranteed for some periods, the VM can increase the guarantee and still finish the transmission within the expected time.

Questions

- Q: How does SoftBW interact with the underlying TCP protocol? Will it make TCP unstable?
- A: As we have shown in the experiments, the overhead on RTT is about 1.9us. This is negligible as compared with the round-trip-time between VMs. It is even less than the jitter of RTT in real systems. So, it will not interfere the underlying TCP flows.
- Also, in our experiment on convergence, the throughput of the TCP flow is very stable.

Questions

- Q: Do you have an indication that increasing the utilization in the network by 3x would not first hit the limits of other resources, e.g., CPU or memory?
- A: In real-world situation, this does depend on applications. This may happen when the application is CPU or memory intensive. However, such applications are beyond our discussion. For applications using network resource, filling up 10 Gbps bandwidth needs only one CPU core (with 1500B MTU size). In such application scenario, SoftBW will benefit the network utilization for the providers. Hence, in our simulation, we only consider the bandwidth resource.