

Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud

- Qingxi Li

Outline

- Amazon Elastic Compute Cloud
- Checkpointing

Cloud Computing

- *Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable **computing resources** (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

NIST Sep 2010

EC2: Instance Type - Hardware

- Standard instance

instance	CPU	Memory	Disk
Small	1 core	1.7 GB	160 GB
Large	4 cores	7.5 GB	850 GB
Extra-large	8 cores	15 GB	1650 GB

EC2: Instance Type - Hardware

- Standard instance
- Micro instance
 - Lower throughput applications need significant compute cycles
- High-Memory instance
- High-CPU instance
- Cluster compute instance
- Cluster GPU instance

EC2: Instance Type - Software

- Operating System
- Database
- Batch processing
- Web hosting
- Application development environment
- Application server
- Video encoding & streaming

Pricing Models

- On-Demand Instance
 - Pay by hour and without long-term commitment

Price – On-Demand

Region:	US East (Virginia) ▼	
	Linux/UNIX Usage	Windows Usage
Standard On-Demand Instances		
Small (Default)	\$0.085 per hour	\$0.12 per hour
Large	\$0.34 per hour	\$0.48 per hour
Extra Large	\$0.68 per hour	\$0.96 per hour
Micro On-Demand Instances		
Micro	\$0.02 per hour	\$0.03 per hour
Hi-Memory On-Demand Instances		
Extra Large	\$0.50 per hour	\$0.62 per hour
Double Extra Large	\$1.00 per hour	\$1.24 per hour
Quadruple Extra Large	\$2.00 per hour	\$2.48 per hour
Hi-CPU On-Demand Instances		
Medium	\$0.17 per hour	\$0.29 per hour
Extra Large	\$0.68 per hour	\$1.16 per hour
Cluster Compute Instances		
Quadruple Extra Large	\$1.60 per hour	N/A*
Cluster GPU Instances		
Quadruple Extra Large	\$2.10 per hour	N/A*
* Windows® is not currently available for Cluster Compute or Cluster GPU Instances		

Pricing Models

- On-Demand Instance
- Reserved Instance
 - One-time payment for reserved capacity
 - May have discount
 - Long-term commitment

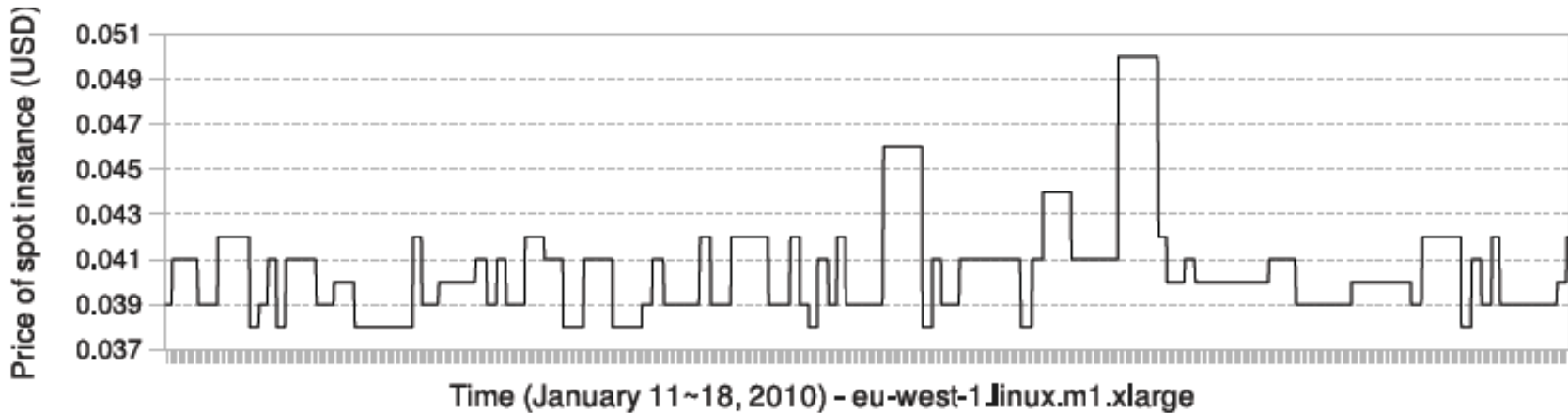
Price - Reserved

Region:	US East (Virginia) ▼			
	1 yr Term	3 yr Term	Linux/UNIX Usage	Windows Usage
Standard Reserved Instances				
Small (Default)	\$227.50	\$350	\$0.03 per hour	\$0.05 per hour
Large	\$910	\$1400	\$0.12 per hour	\$0.20 per hour
Extra Large	\$1820	\$2800	\$0.24 per hour	\$0.40 per hour
Micro Reserved Instances				
Micro	\$54	\$82	\$0.007 per hour	\$0.013 per hour
High-Memory Reserved Instances				
Extra Large	\$1325	\$2000	\$0.17 per hour	\$0.24 per hour
Double Extra Large	\$2650	\$4000	\$0.34 per hour	\$0.48 per hour
Quadruple Extra Large	\$5300	\$8000	\$0.68 per hour	\$0.96 per hour
High-CPU Reserved Instances				
Medium	\$455	\$700	\$0.06 per hour	\$0.125 per hour
Extra Large	\$1820	\$2800	\$0.24 per hour	\$0.50 per hour
Cluster Compute Reserved Instances				
Quadruple Extra Large	\$4290	\$6590	\$0.56 per hour	N/A*
Cluster GPU Reserved Instances				
Quadruple Extra Large	\$5630	\$8650	\$0.74 per hour	N/A*
* Windows® is not currently available for Cluster Compute or Cluster GPU Instances				

Pricing Models

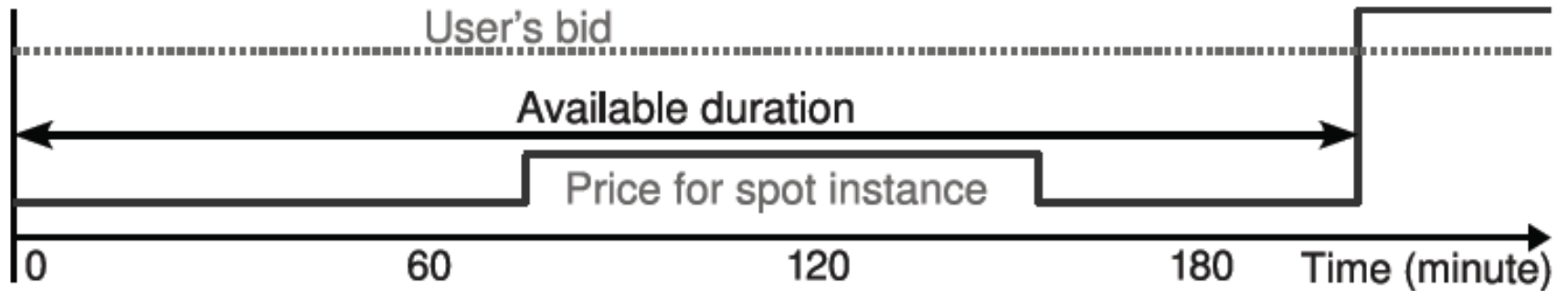
- On-Demand Instance
- Reserved Instance
- Spot Instance
 - Bid the capacity unused
 - Cheaper than on-demand instance
 - Can be cut at any time

Spot Price fluctuation

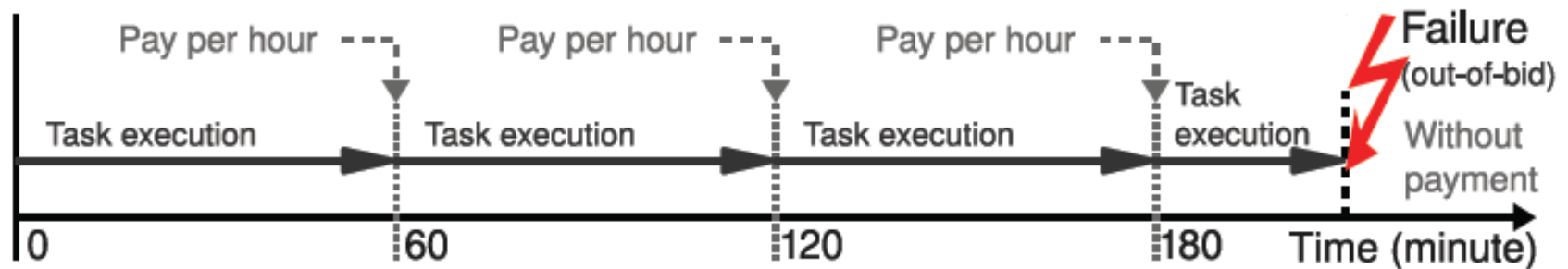


- Rising edges
 - More bidders
 - Less resource
 - High bids from users

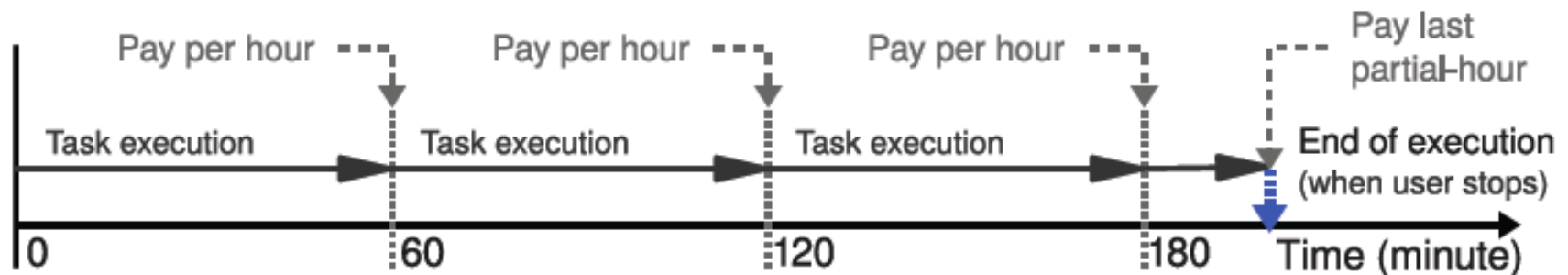
Spot Instance Model -Detail



Spot Instance Model -Detail

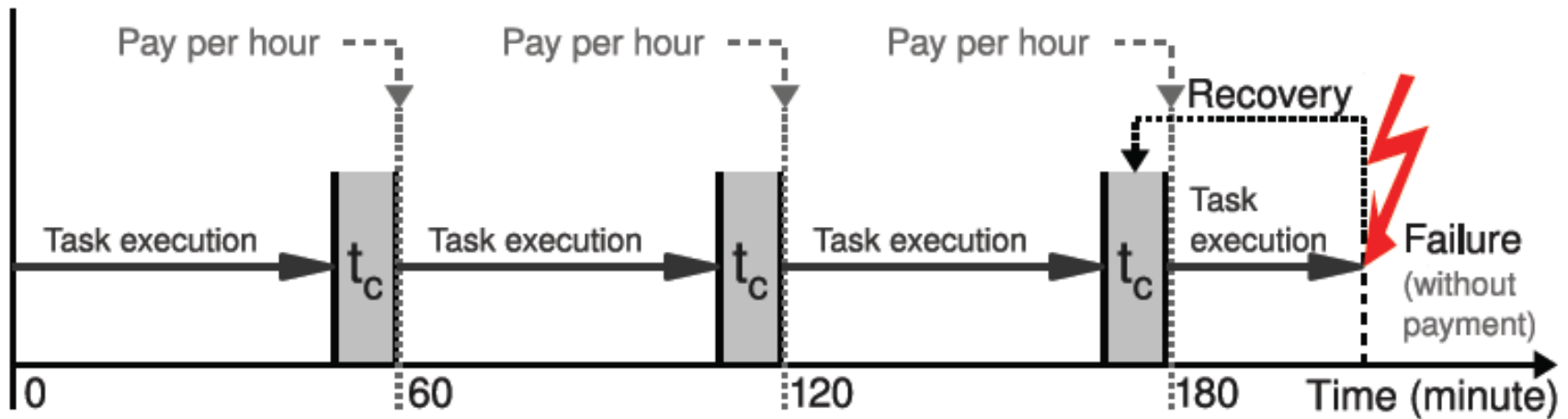


(a) When a user's spot instance is out-of-bid



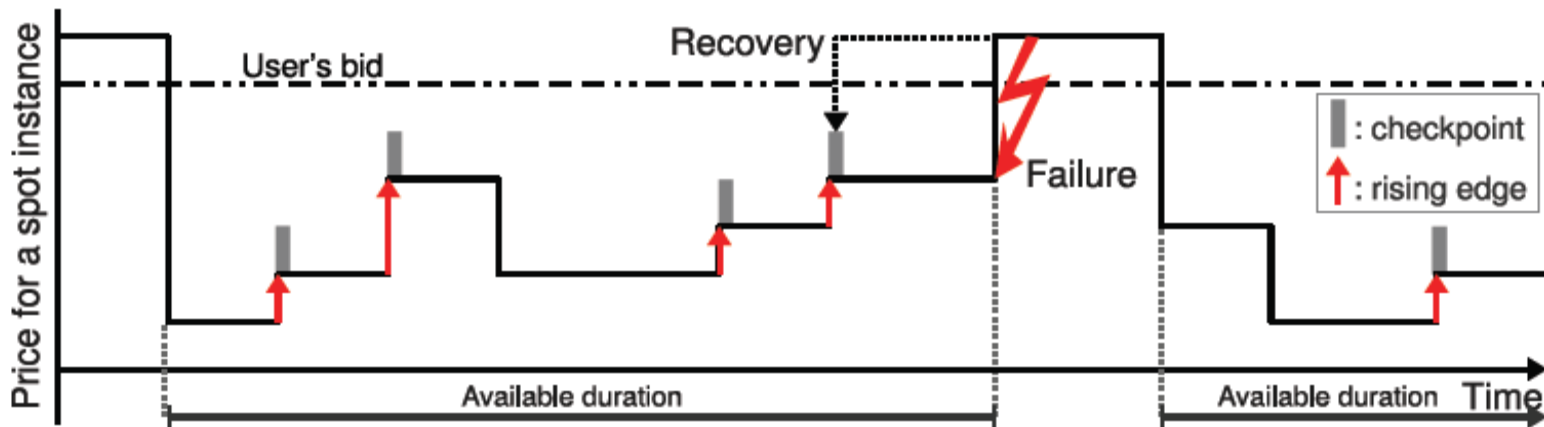
(b) When a user stops using spot instance

CheckPointing - Hourly



- One hour is the smallest unit of pricing

CheckPointing – Rising edge

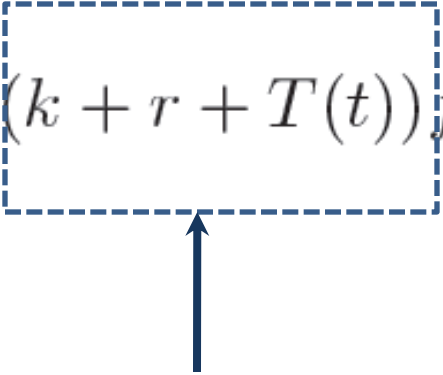


- Rising edges:
 - The aborting possibility is rising

CheckPointing - Adaptive

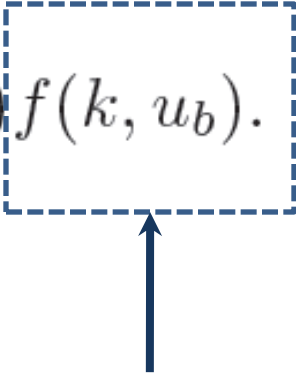
- Taking hourly checkpointing if $H_{\text{skip}}(t) > H_{\text{take}}(t)$
 - $H_{\text{skip}}(t)$: Expected recovery time if we skip the hourly checkpointing.
 - $H_{\text{take}}(t)$: Expected recovery time if we take the hourly checkpointing.
 - t : this checking point is t time units after the previous checkingpoint.
- Taking edge rising checkpointing if $E_{\text{skip}}(t) > E_{\text{take}}(t)$

$$H_{\text{skip}}(t)$$

$$H_{\text{skip}}(t) = \sum_{k=0}^{t_r-1} (k + r + T(t)) f(k, u_b).$$


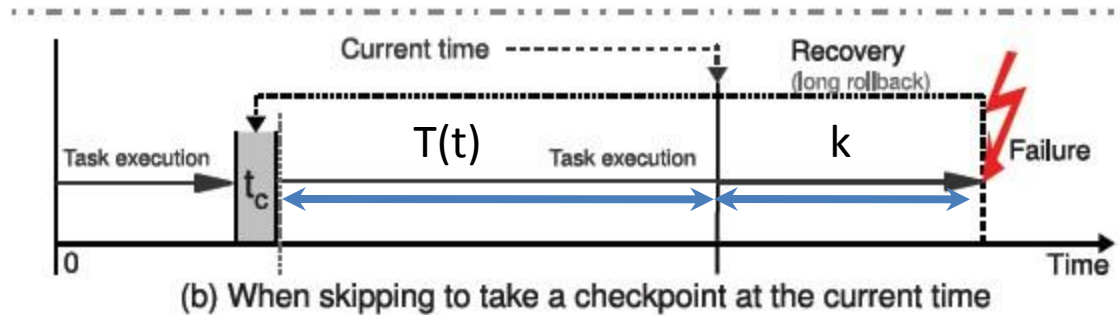
Recovery time when failure
happened after k time units

$$H_{\text{skip}}(t)$$

$$H_{\text{skip}}(t) = \sum_{k=0}^{t_r-1} (k + r + T(t)) f(k, u_b).$$


The possibility that failure happened
with k time units & bid price as u_b

$$H_{\text{skip}}(t)$$



$$H_{\text{skip}}(t) = \sum_{k=0}^{t_r-1} (k + r + \boxed{T(t)}) f(k, u_b).$$

Expected execution time from the last checkpointing to now

r : restart time

k : re-execute time of the k time units

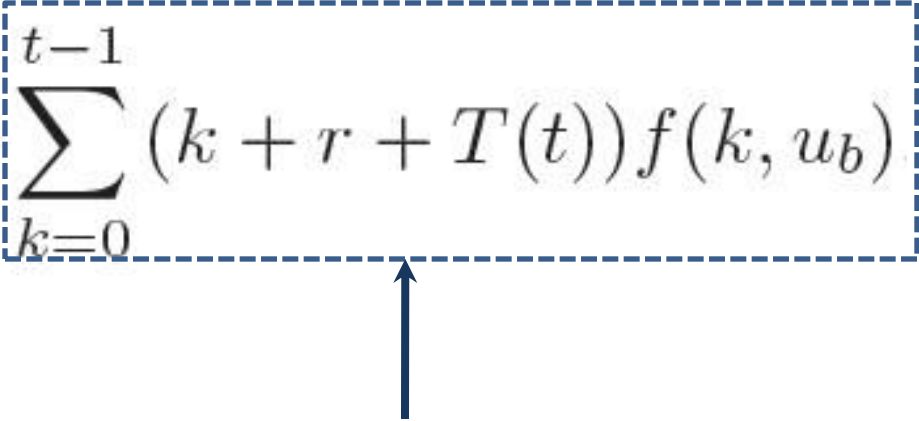
$T(t)$

$$T(t) = \sum_{k=t}^{\infty} t f(k, u_b) + \sum_{k=0}^{t-1} (k + r + T(t)) f(k, u_b)$$



Failure happened after this t time units

$T(t)$

$$T(t) = \sum_{k=t}^{\infty} t f(k, u_b) + \sum_{k=0}^{t-1} (k + r + T(t)) f(k, u_b)$$


Failure happened during this t time units

$$T(t)$$

$$T(t) = \frac{t + \sum_{k=0}^{t-1} (k + r - t) f(k, u_b)}{1 - \sum_{k=0}^{t-1} f(k, u_b)}.$$

$$H_{\text{take}}(t)$$

$$H_{\text{take}}(t) = \sum_{k=0}^{t_c-1} (k + r + T(t)) f(k, u_b) + \sum_{k=t_c}^{t_r-1} (k + r) f(k, u_b) + T(t_c)$$

Overhead of taking
checkpointing



$$H_{\text{take}}(t)$$

$$H_{\text{take}}(t) = \sum_{k=0}^{t_c-1} (k + r + T(t)) f(k, u_b) + \sum_{k=t_c}^{t_r-1} (k + r) f(k, u_b) + T(t_c)$$



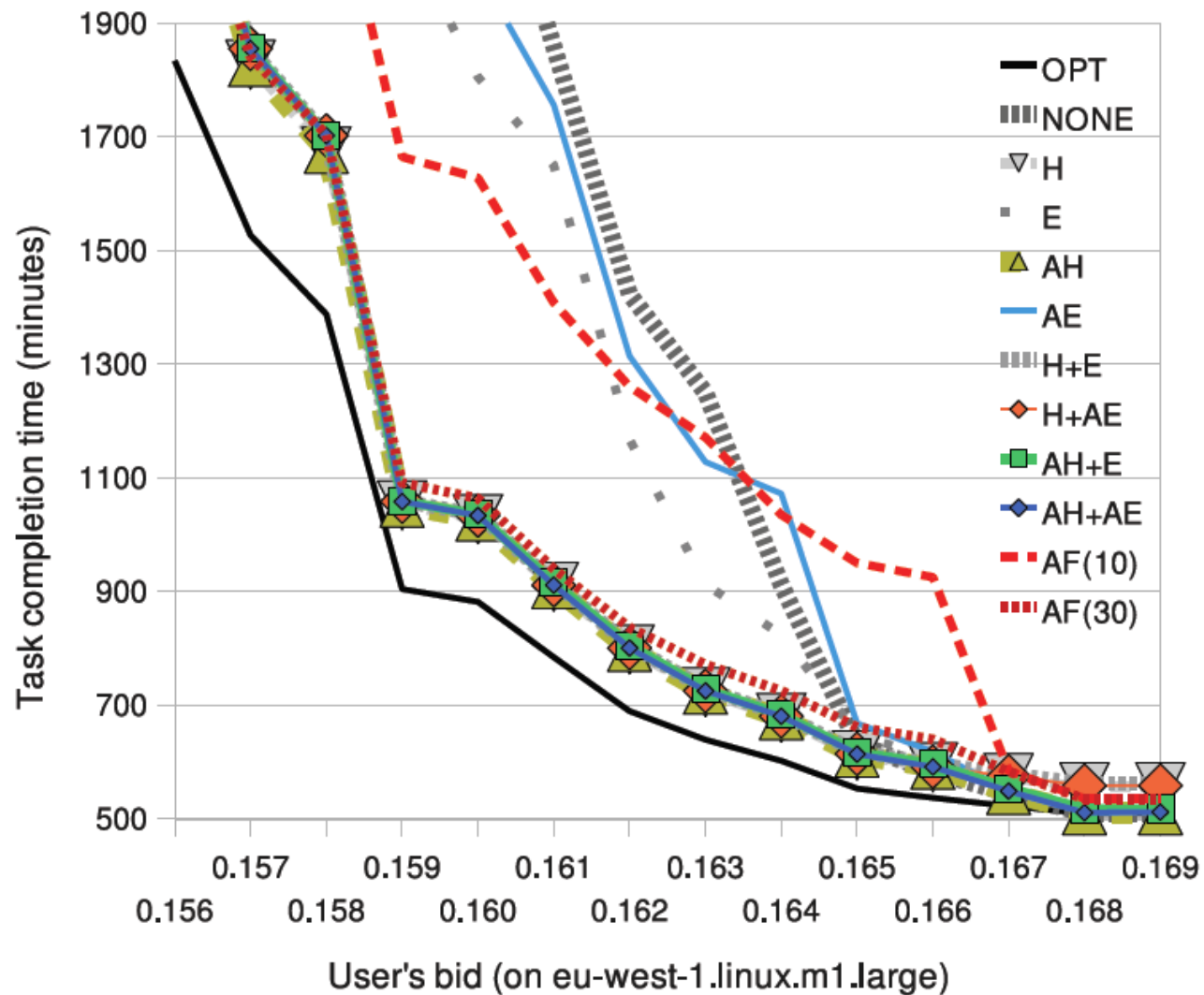
Failure happened when we are making the checkpointing.

$$H_{\text{take}}(t)$$

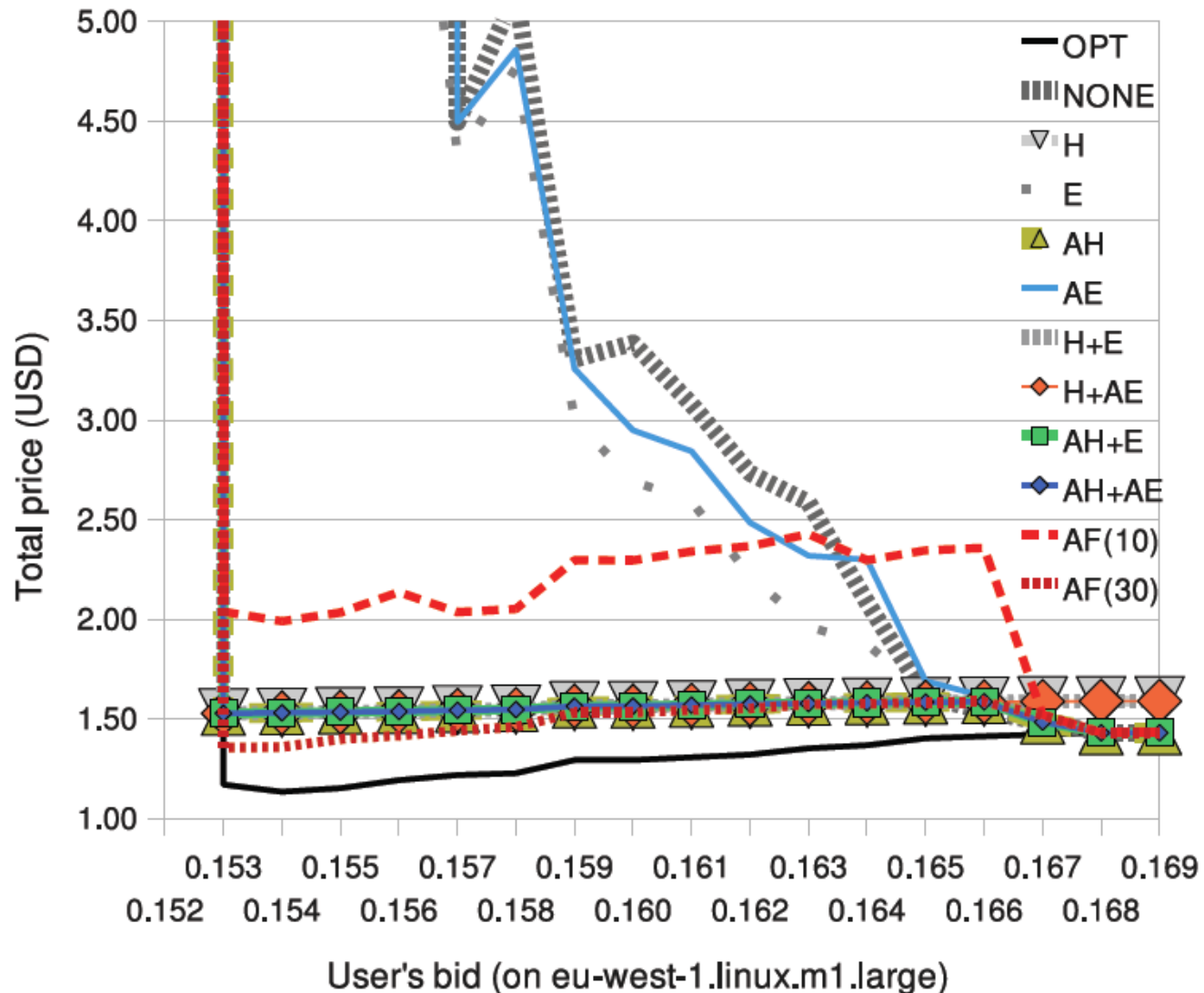
$$H_{\text{take}}(t) = \sum_{k=0}^{t_c-1} (k + r + T(t)) f(k, u_b) + \sum_{k=t_c}^{t_r-1} (k + r) f(k, u_b) + T(t_c)$$

Failure happened after taking checkpointing.

Result – Completion Time



Result – Total Price



Discussion Questions

- Besides taking checkpointing, are there any other ways can save the completion time or cost of the tasks?
- Compared with on-demand price model, what applications will prefer spot price model?

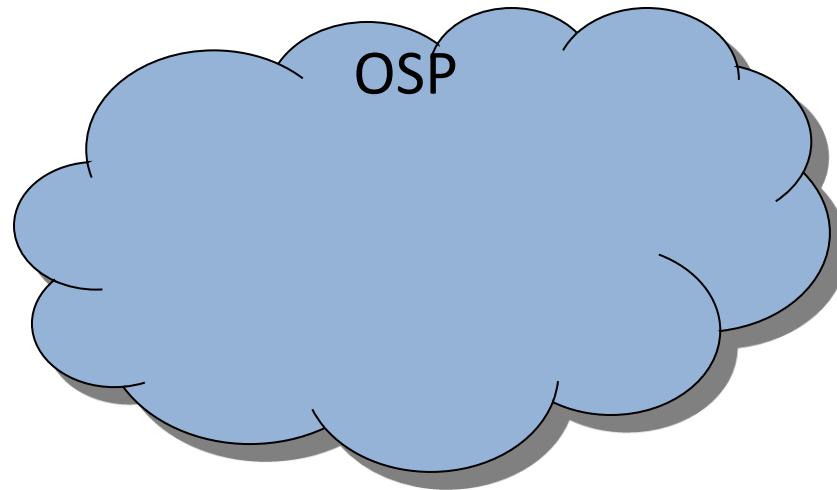
Optimizing Cost and Performance in Online Service Provider Networks

Ming Zhang

Microsoft Research

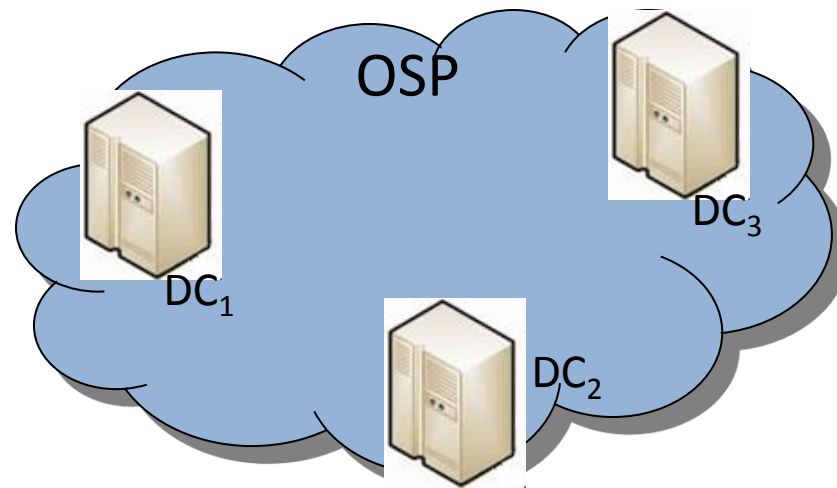
Based on slides by Ming Zhang

Online Service Provider (OSP) network



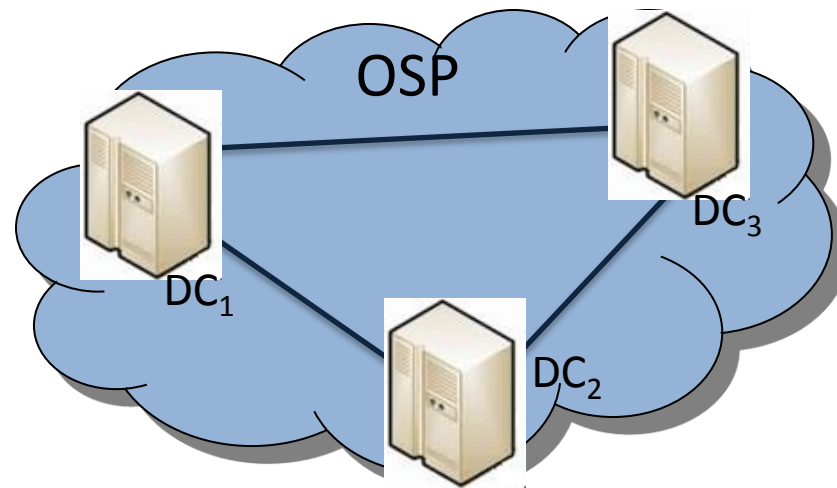
Google Microsoft YAHOO! amazon.com

OSP network



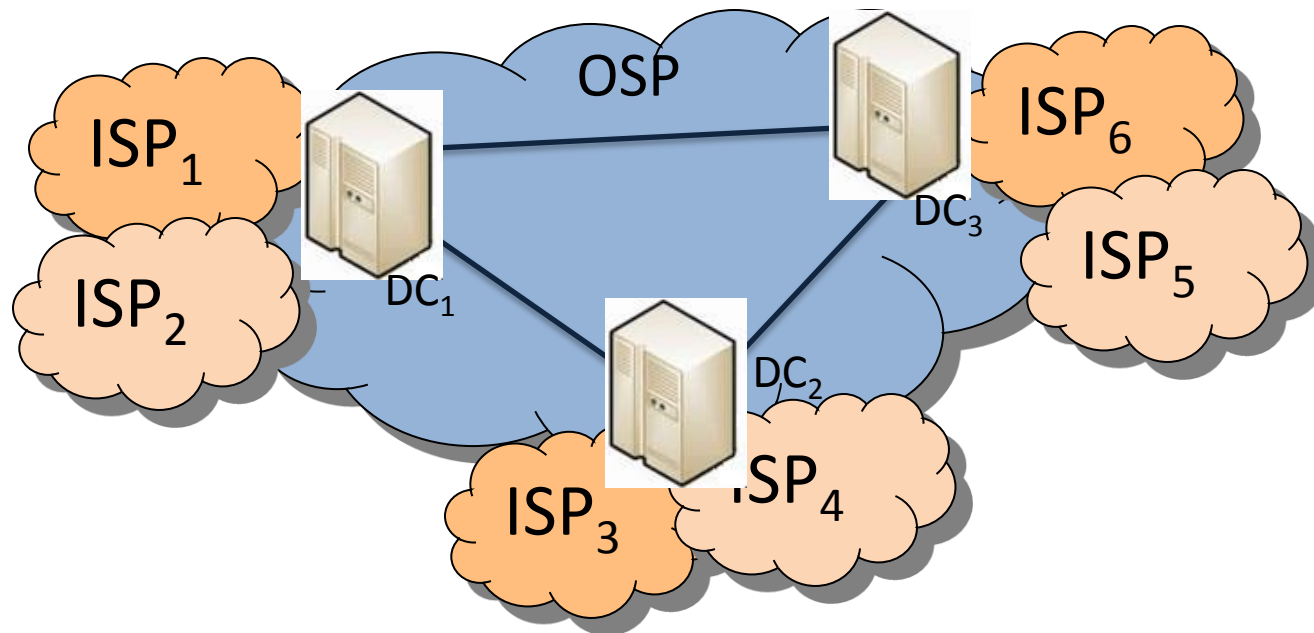
Google Microsoft YAHOO! amazon.com

OSP network



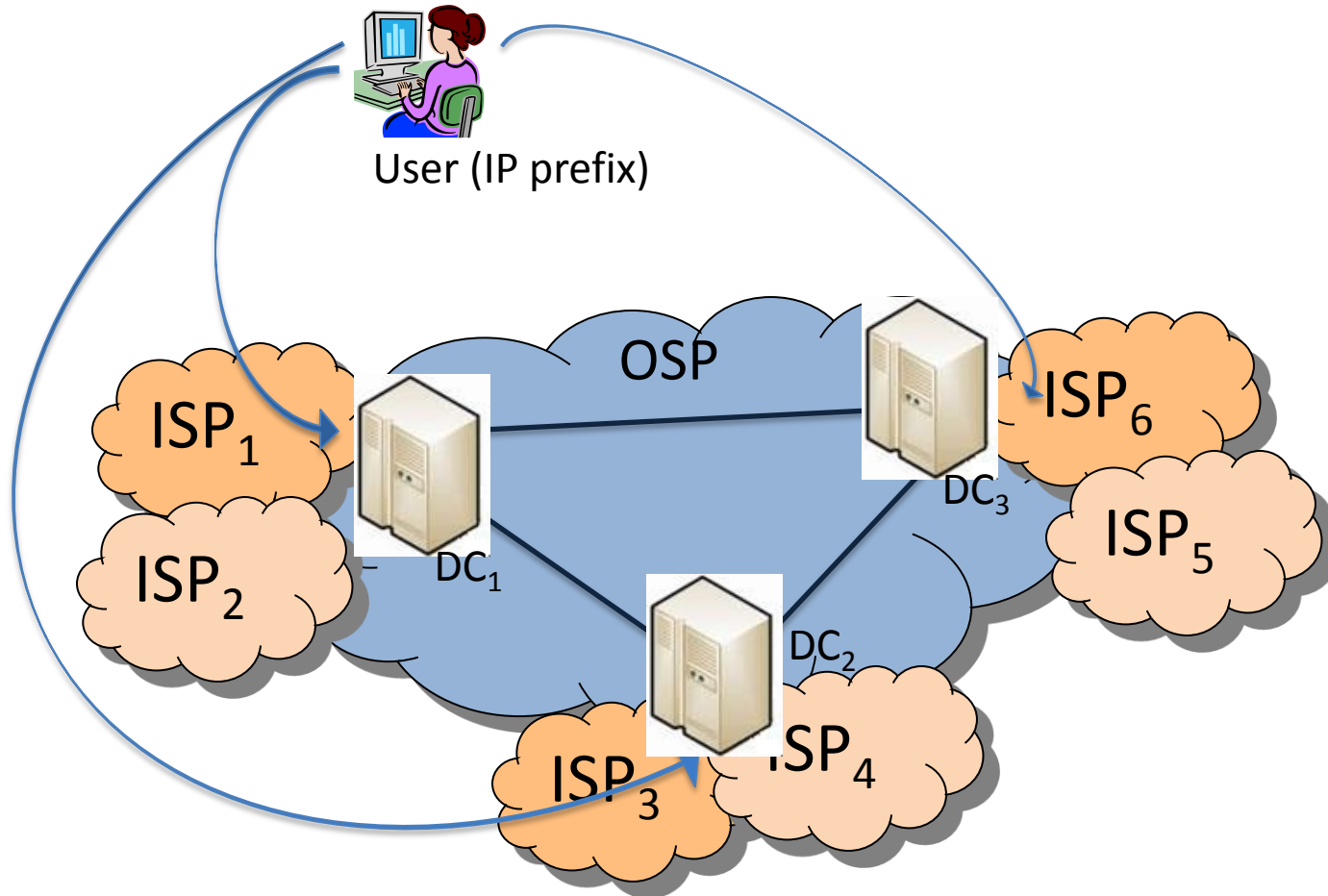
Google Microsoft YAHOO! amazon.com

OSP network



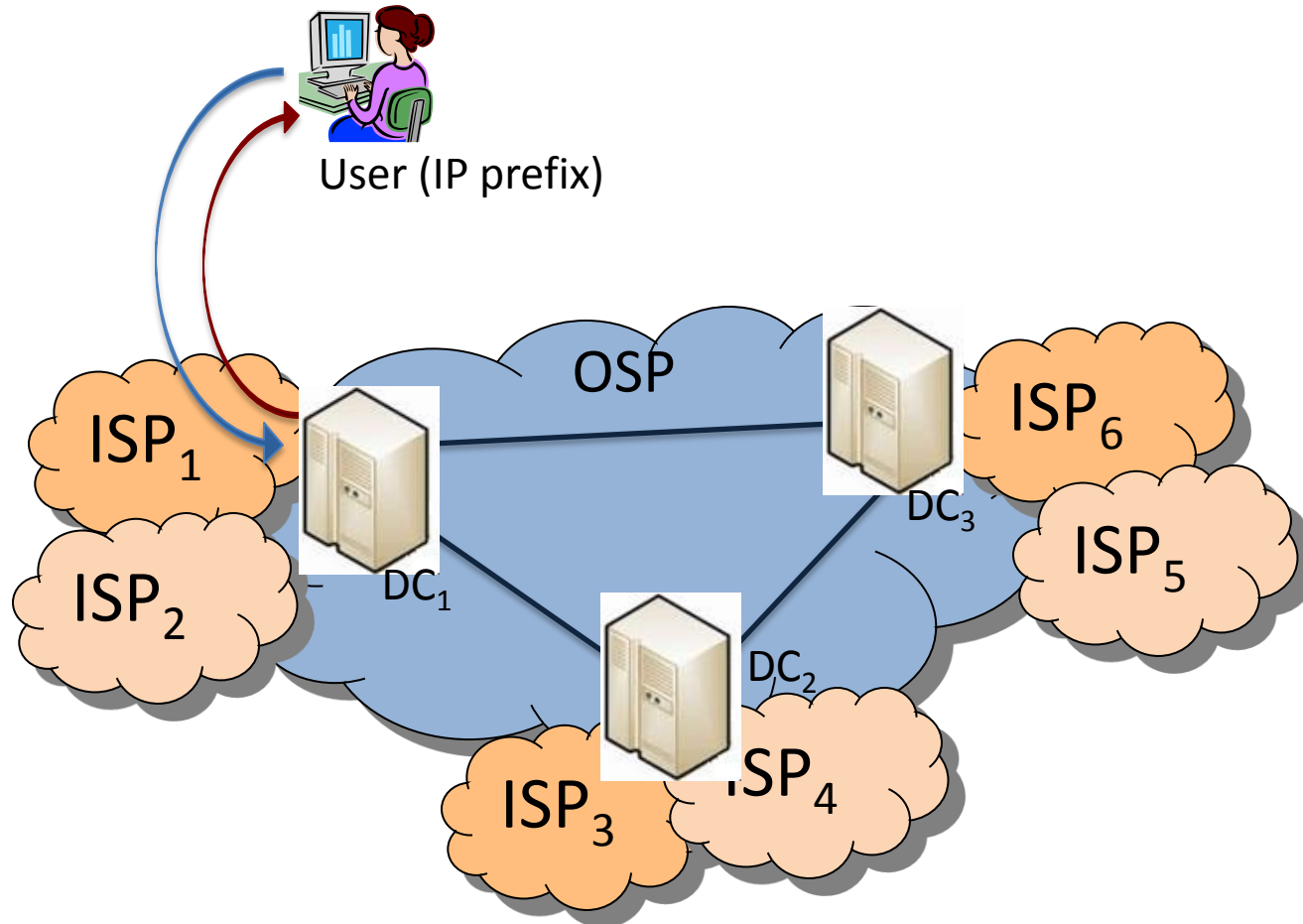
Google Microsoft YAHOO! amazon.com

OSP network



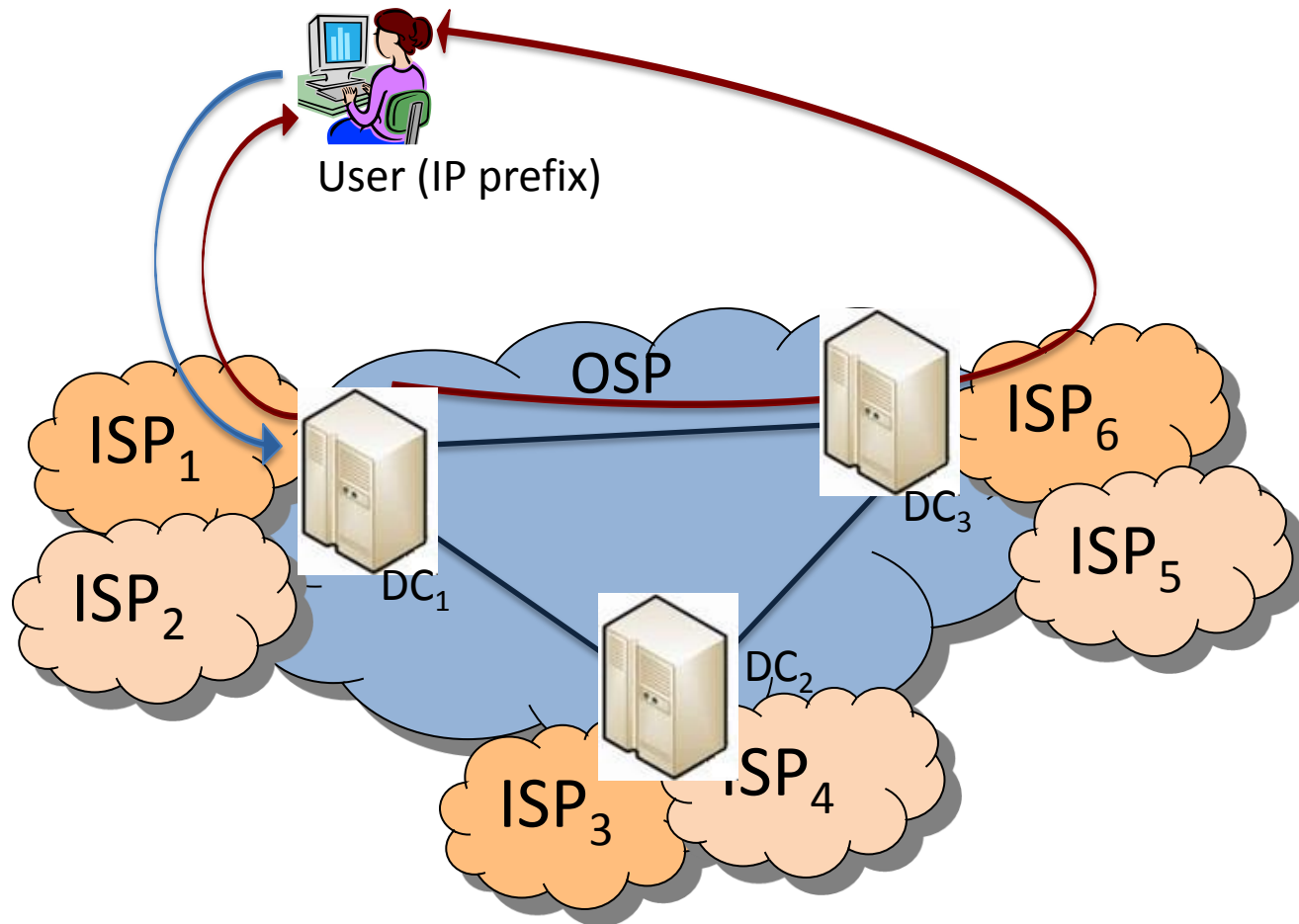
Google Microsoft YAHOO! amazon.com

OSP network



Google Microsoft YAHOO! amazon.com

OSP network



Google Microsoft YAHOO! amazon.com

Key factors in OSP traffic engineering

- Cost
 - Google Search: 5B queries/month
 - MSN Messenger: 330M users/month
 - Traffic volume exceeding a PB/day
- Performance
 - Directly impacts user experience and revenue
 - Purchases, search queries, ad click-through rates

Current TE solution is limited

- Current practice is mostly manual
 - Incoming: DNS redirection, nearby DC
 - Outgoing: BGP, manually configured
- Complex TE strategy space
 - ($\sim 300\text{K}$ prefixes) \times (~ 10 DC) \times (~ 10 routes/prefix)
 - Link capacity creates dependencies among prefixes

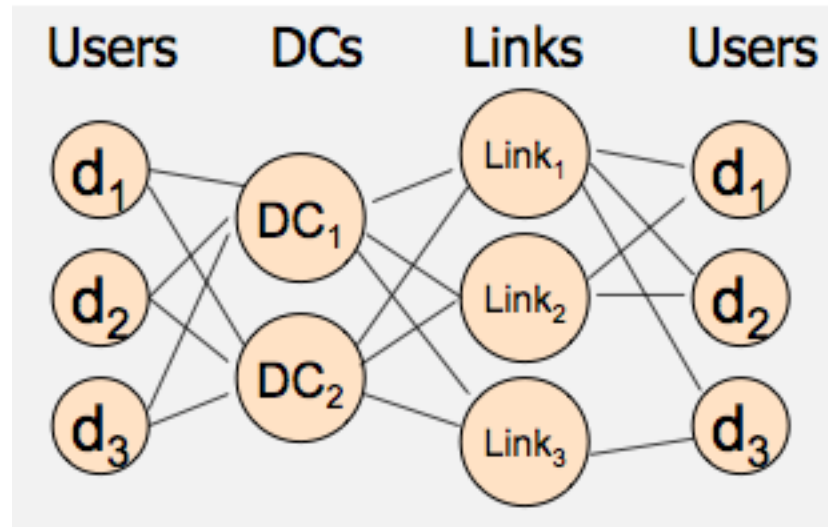
Prior work on TE

- Intra-domain TE for transit ISPs
 - Balancing load across internal paths
 - Not considering end-to-end performance
- Route selection for multi-homed stub networks
 - Single site
 - Small number of ISPs

Contributions of this work

- Formulation of OSP TE problem
- Design & implementation of Entact
 - A route-injection-based measurement
 - An online TE optimization framework
- Extensive evaluations in MSN
 - 40% cost reduction
 - Low operational overheads

Problem formulation

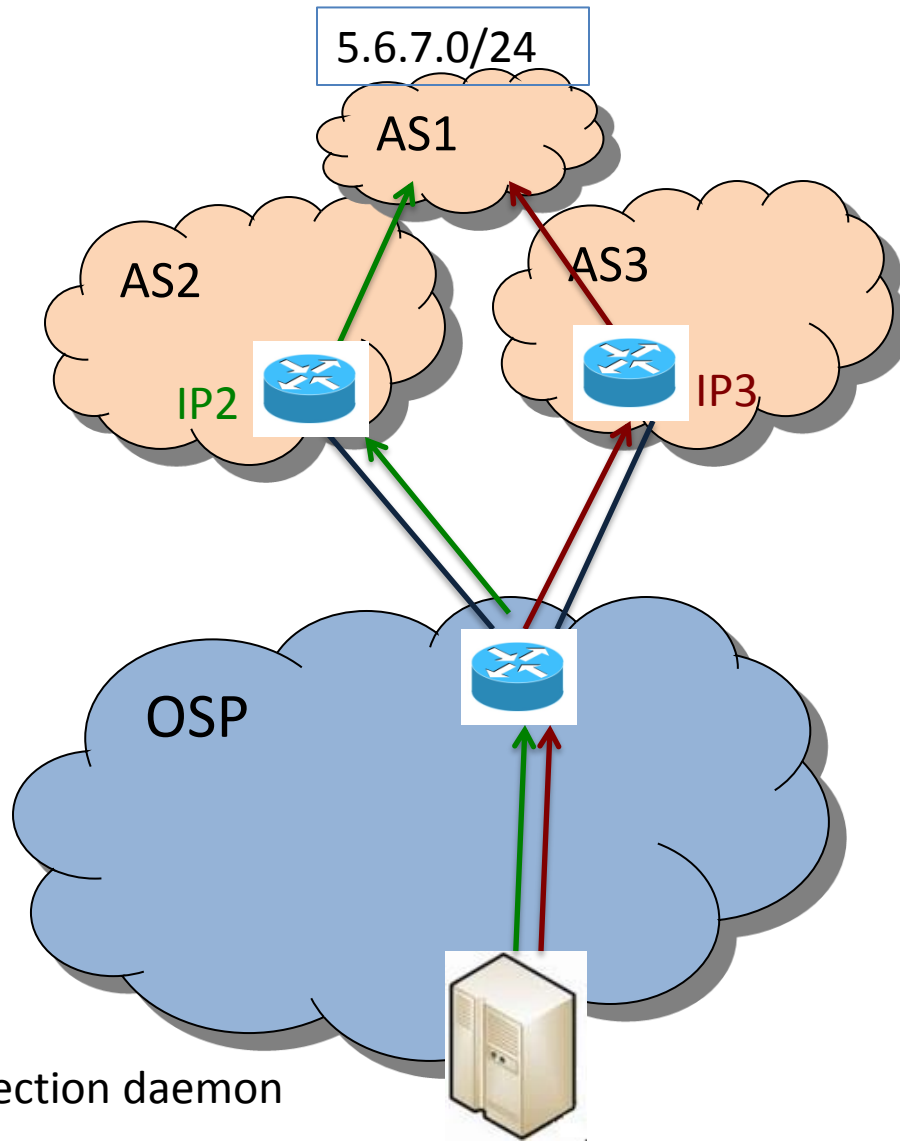


- **INPUT:** user prefixes, DCs, external links
- **OUTPUT:** TE strategy, user prefix \rightarrow (DC, external link)
- **CONSTRAINTS:** link capacity, route availability

Performance & cost measures

- Use RTT as the performance measure
 - Many latency-sensitive apps: search, email, maps
 - Apps are chatty: $N \times \text{RTT}$ quickly gets to 100+ms
- Transit cost: $F(v) = \text{price} \times v$
 - Ignore internal traffic cost

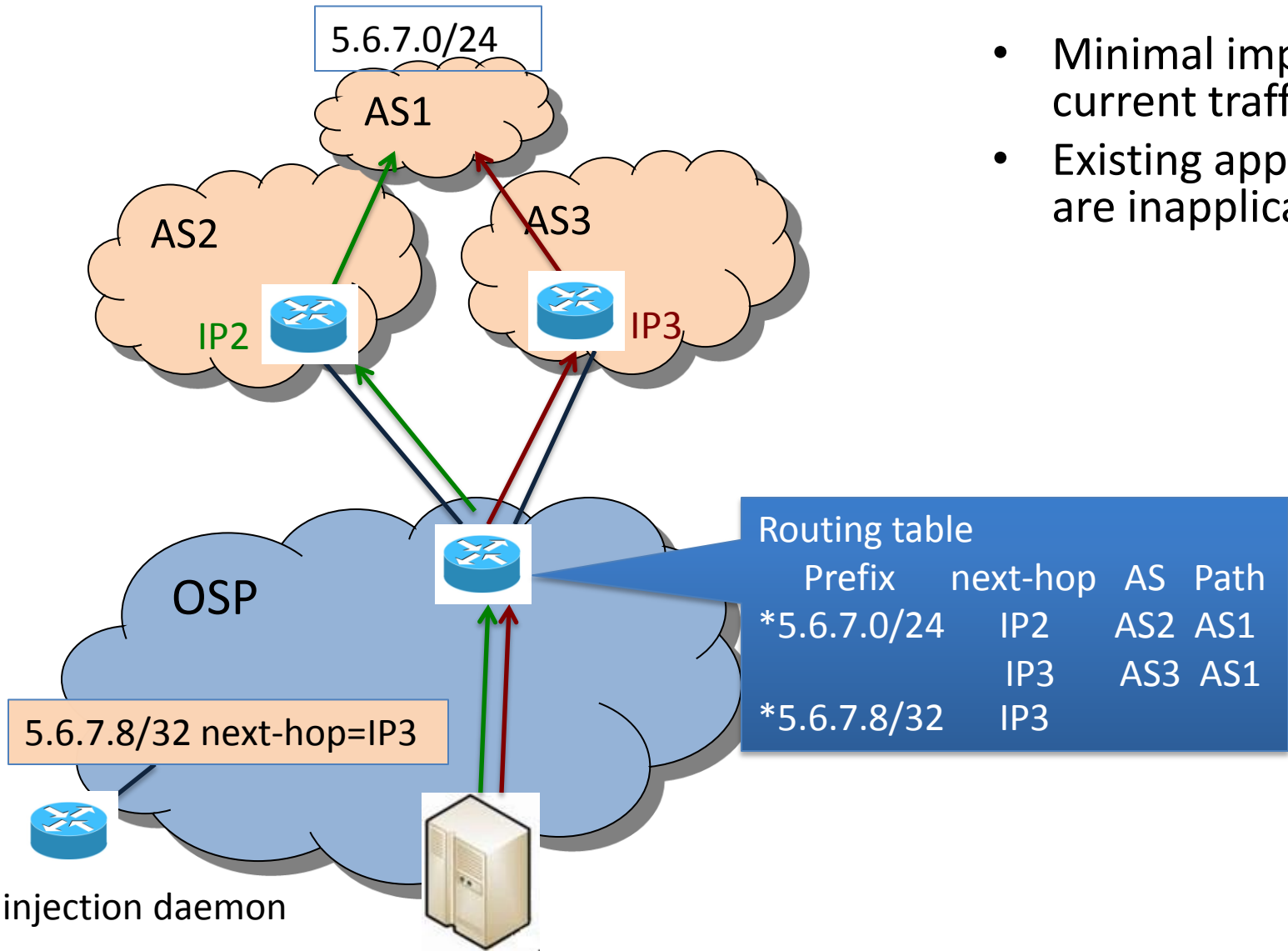
Measuring alternative paths with route injection



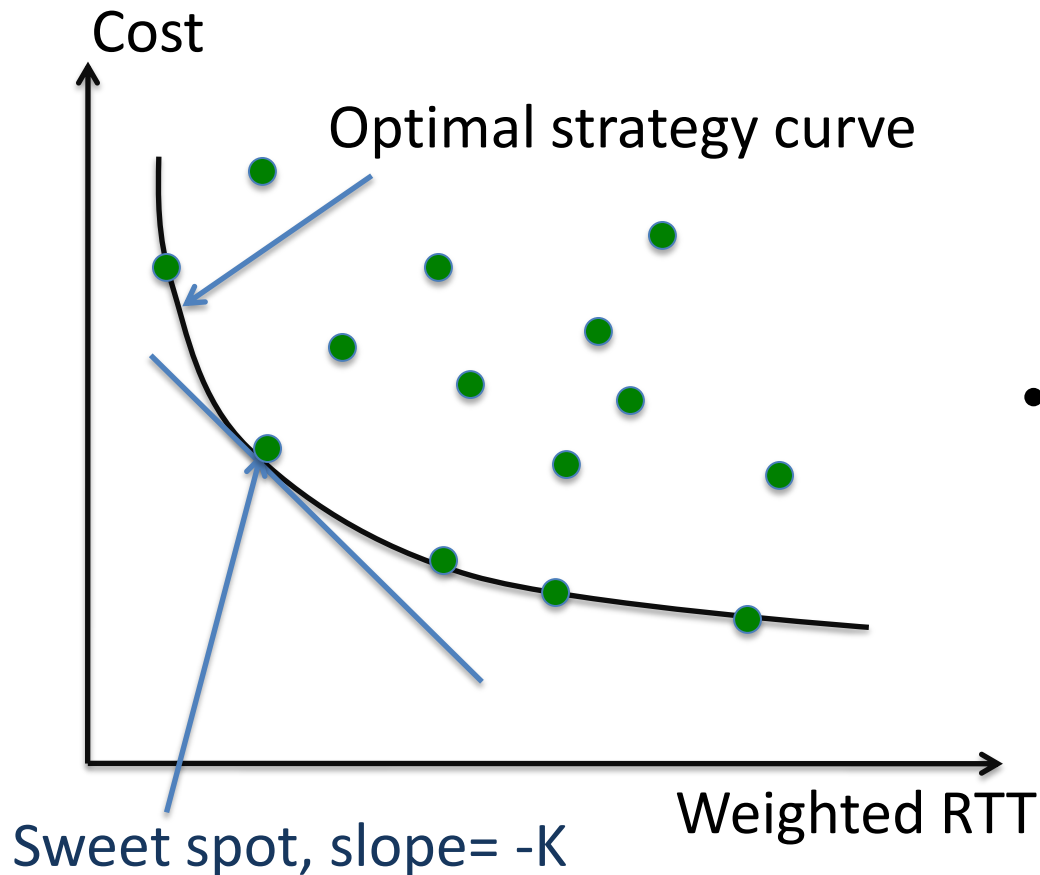
- Minimal impact on current traffic
- Existing approaches are inapplicable

Measuring alternative paths with route injection

- Minimal impact on current traffic
- Existing approaches are inapplicable



Selecting desirable strategy

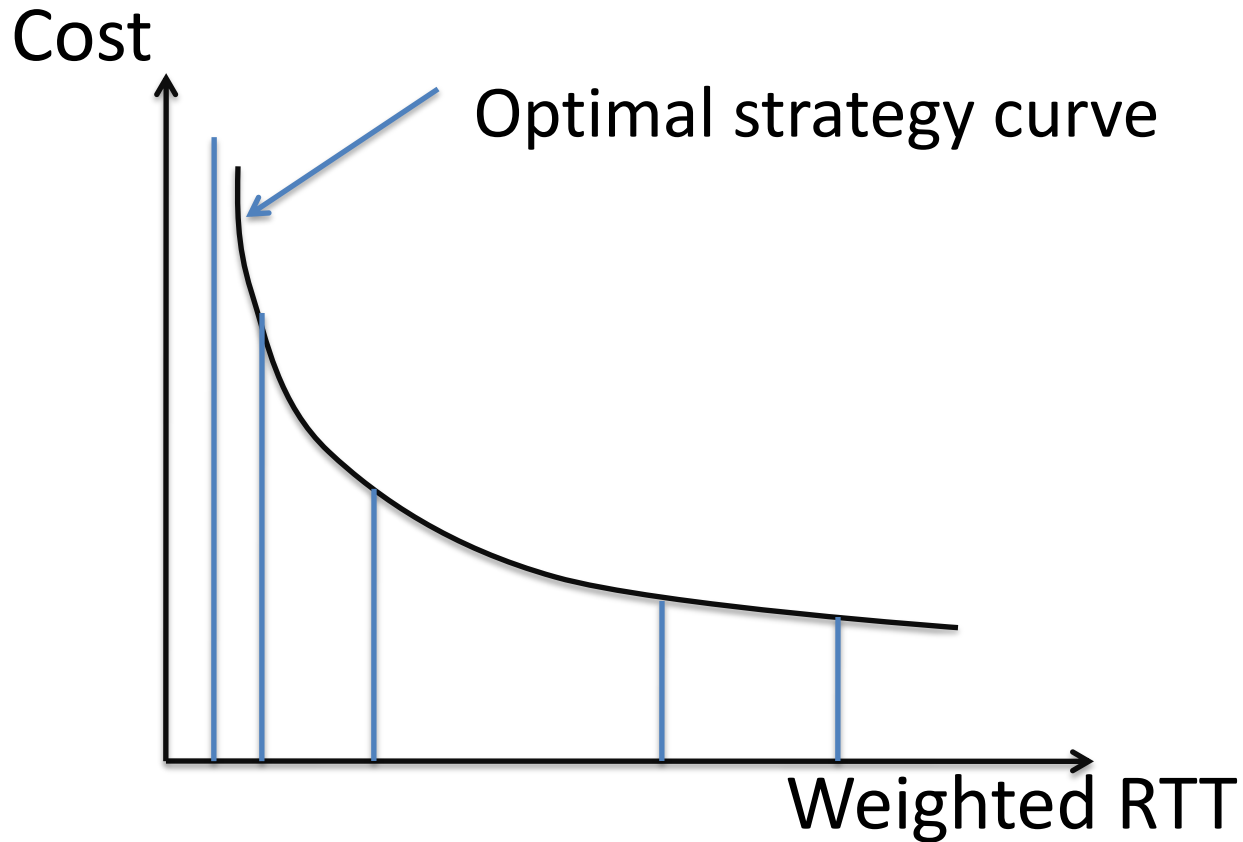


- M^N strategies for N prefixes and M alternative paths/prefix
 - Only consider optimal strategies
- Finding “sweet spot” based on desirable cost-performance tradeoff
 - K extra cost for unit latency decrease

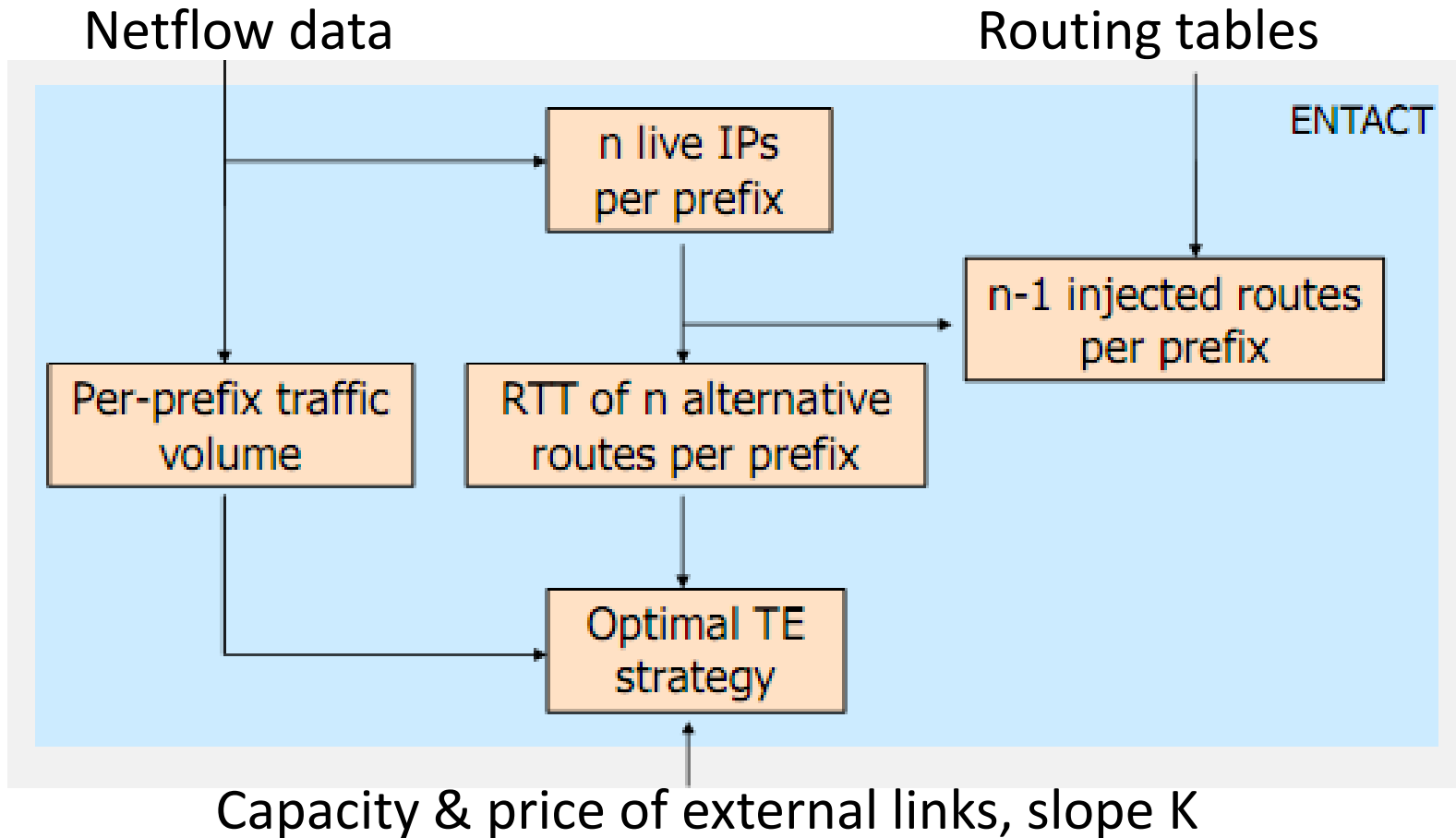
Computing optimal strategy

- P95 cost optimization is complex
 - Optimize short-term cost online
 - Evaluate using P95 cost
- An ILP problem
 - STEP1: Find a fractional solution
 - STEP2: Convert to an integer solution

Finding optimal strategy curve



Entact architecture



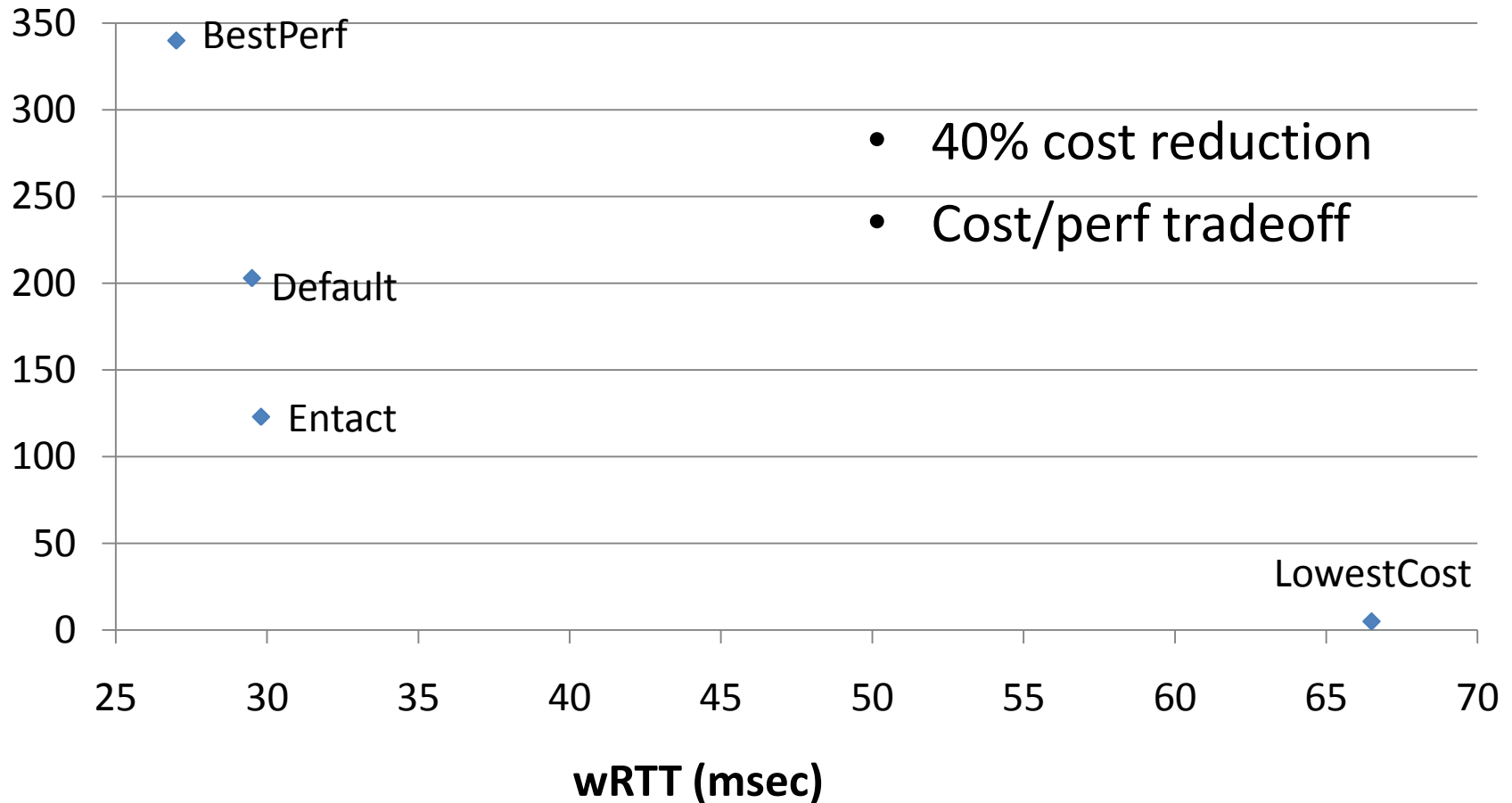
Experimental setup



- MSN: one of the largest OSP networks
 - 11 DCs, 1,000+ external links
- Assumptions in evaluation
 - Traffic and performance do not change with TE strategies
- 6K destination prefixes from 2,791 ASes
 - High-volume, single-location, representative

Results

Cost (per unit traffic)



Where does cost reduction come from?

Path chosen by Entact	Prefixes (%)	wRTT difference (msec)	Short-term cost difference
Same	88.2	0	0
Cheaper & shorter	1.7	-8	-309
Cheaper & longer	5.5	+12	-560
Pricier & shorter	4.6	-15	+42
Pricier & longer	0.1	0	0

- Entact makes “intelligent” performance-cost tradeoff
- Automation is crucial for handling complexity & dynamics

Overhead

- Route injection
 - 30k routes, 51sec, 4.84MB in RIB, 4.64MB in FIB
- Traffic shift
- Computation time
 - STEP1: $O(n^{3.5})$
 - STEP2: $O(n^2 \log(n))$
 - 20K prefix ~ 9 sec; 300K prefix ~ 171 sec
- Bandwidth
 - $30K \times 2 \times 2 \times 5 \times 80\text{bytes} / 3600\text{sec} = 0.1\text{Mbps}$

Conclusions

- TE automation is crucial for large OSP network
 - Multiple DCs
 - Many external links
 - Dependencies between prefixes
- Entact – first online TE scheme for OSP network
 - 40% cost reduction w/o performance degradation
 - Low operational overhead

Discussion

- The cost concerned in the paper doesn't cover energy cost on data centers. Should this be part of the optimization object?
- Can OSPs do anything to reduce the user request ingoing latency besides the outgoing one?
- Is the computation complexity too high? If so, can you think of any way to decrease it?
- They probe the same number of alternative paths to one prefix, no matter how many IPs in that prefix. Is this a fair way to implement Entact

Thank you!