Pretium:
Dynamic Pricing and Traffic Engineering for Timely Inter-Datacenter Transfers

Ishai Menache

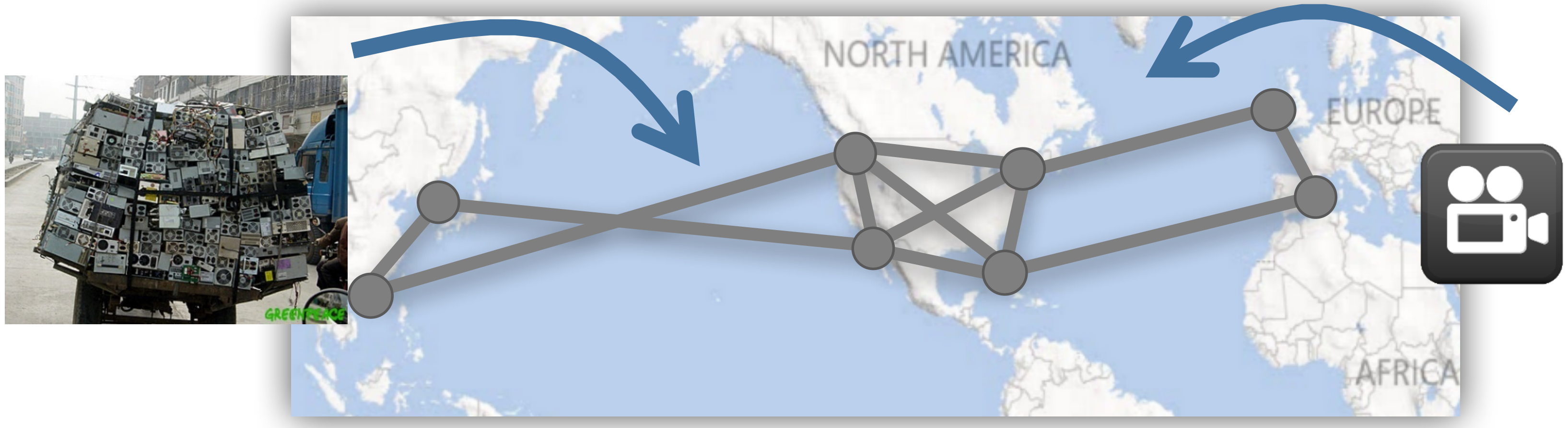
Virajith Jalaparti, Ivan Bliznets,
Brendan Lucier and Srikanth Kandula

Microsoft*

Sigcomm 2016

*Disclamer: Currently not part of Microsoft technology

Inter-datacenter Traffic Engineering (TE)



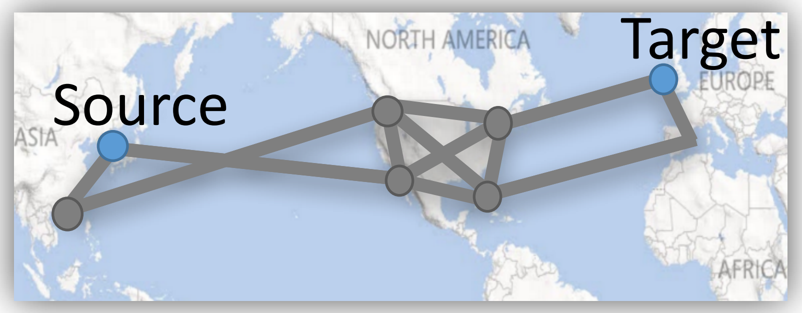
Allocate bandwidth between:

- Rate requests – Interactive apps, video streaming
- Large Transfers – business data, subject to deadlines
- High-priority traffic – Low latency requirements

… while keeping costs low (provisioning and usage)

Existing TE schemes are game-able

- Users, who offer input to TE, can specify:
  - {source, destination} of request
  - {begin-time, deadline}
  - Demand (bytes or rate)
  - Value or priority
- Recent WAN TE prior work: SWAN [Sigcomm’13], B4 [Sigcomm’13], Tempus [Sigcomm’14], Amoeba (Eurosys’15)



Gaming TE = false inputs that offer advantage

- Inflate value/priority
- Report stricter deadline

Challenge

Elicit truthful requirements
while
keeping TE usable

Today’s pricing schemes do not solve TE gaming

Network pricing, today, is largely unrelated to traffic engineering

- Either fixed $/GB wide-area or $/bandwidth at vNIC
  - E.g. $0.02/GB in-region
  - E.g. Lease VMs w/ guaranteed 250Mbps in/out

This hurts both users and providers

- Providers cannot steer traffic to lightly loaded {paths, time-periods}
- Users cannot pay more for better service (e.g., deadline guarantees)

Survey of Microsoft WAN customers

- 81% willing to delay transfers if price is lower
- Can accept dynamic pricing **if** guarantee & price are fixed when transfer starts

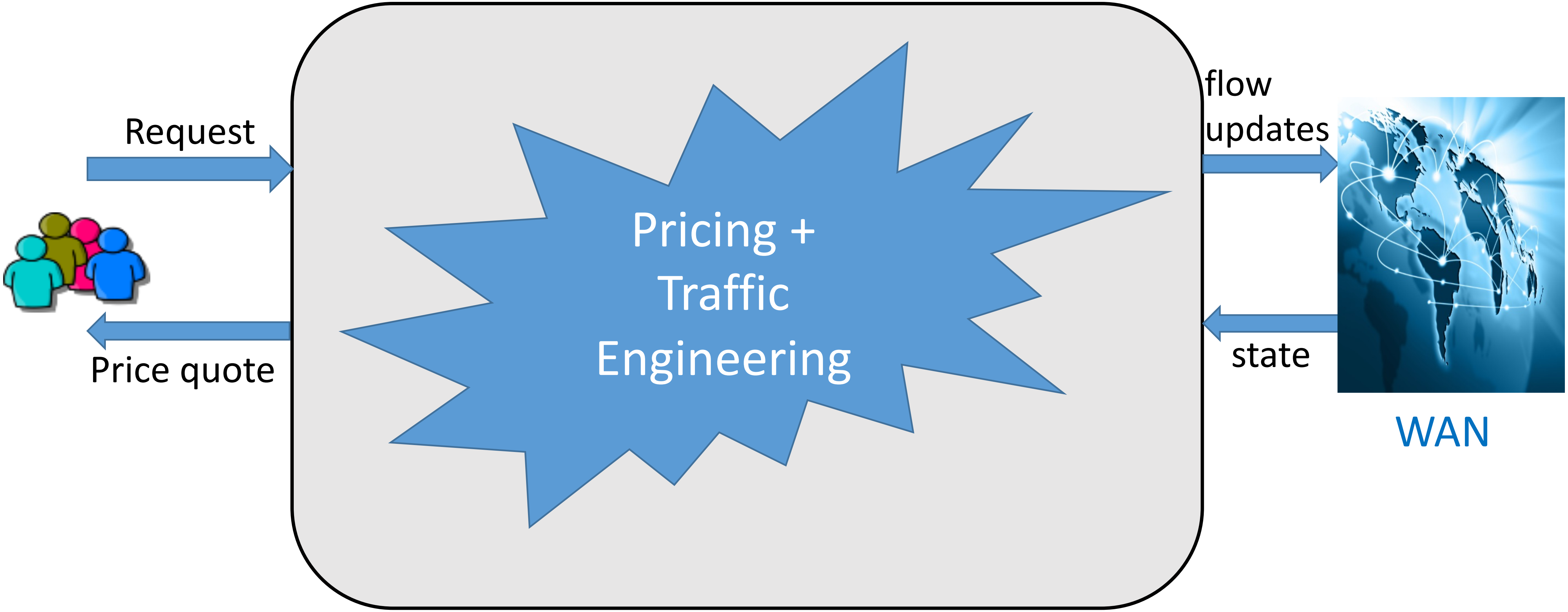
Our goals

A pricing + TE framework that

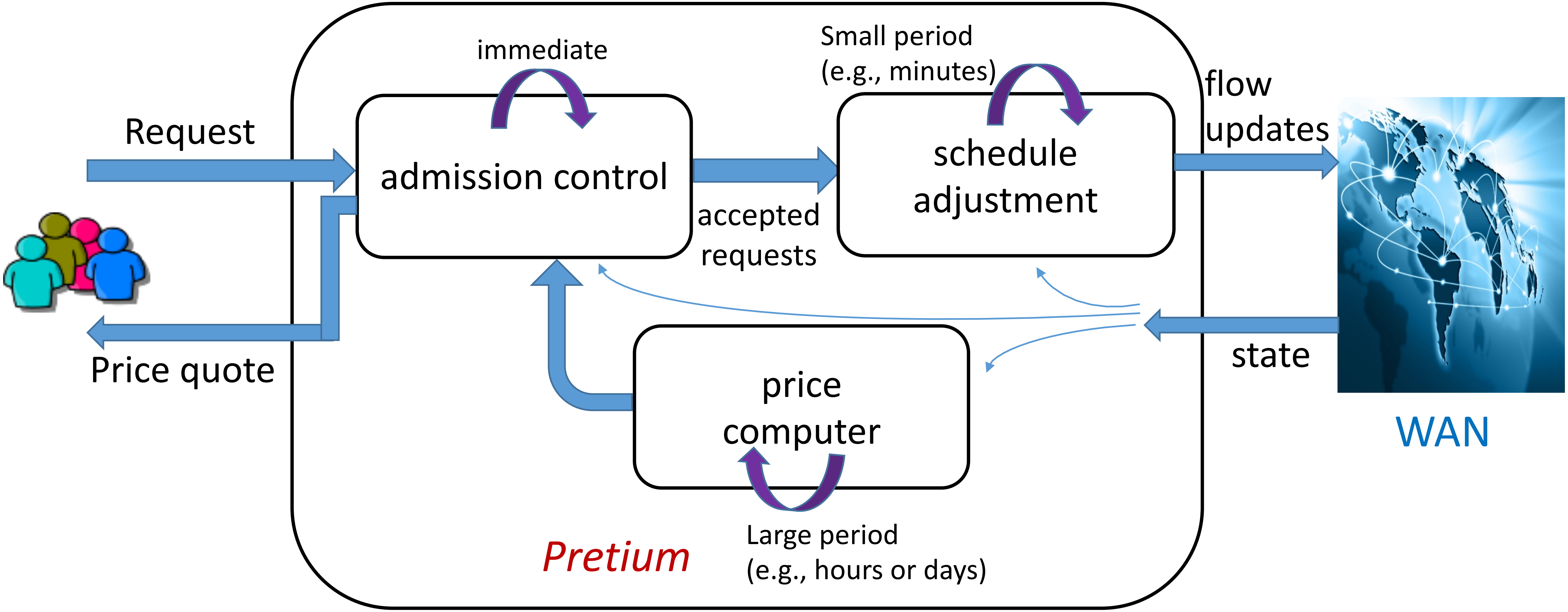
a) pushes users towards being truthful
b) facilitates offering QoS
c) maximizes network efficiency given costs
    - E.g., Welfare: (Total value) minus (operating costs)

- All must be done online, i.e., with imperfect knowledge of future
- Complex costs

Pretium – Dynamic Pricing and TE



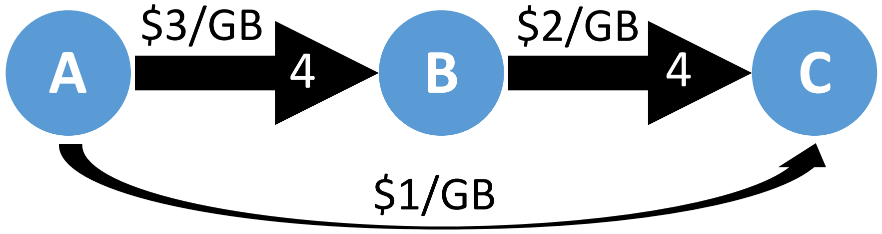
Pretium architecture



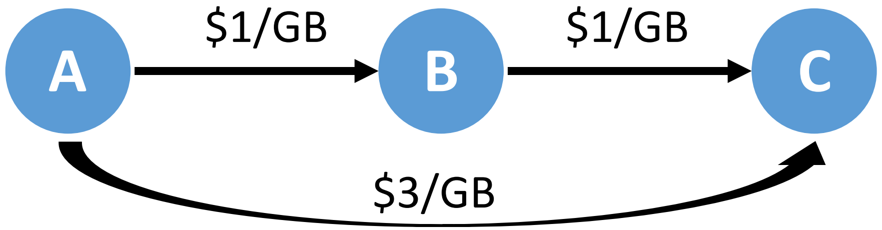
Pricing model

Maintain internal prices per {link, future time-step}

t=1:



t=2:

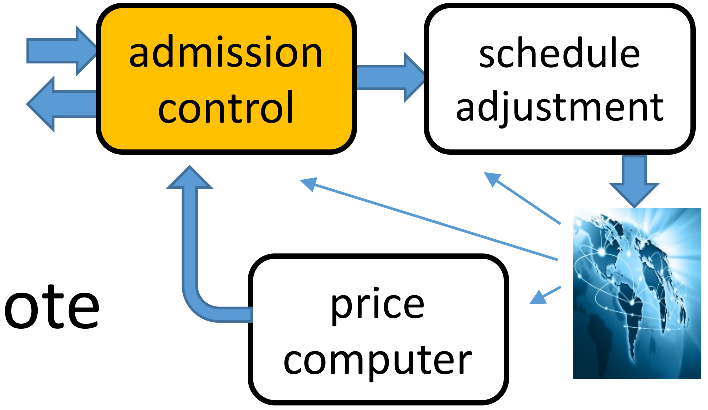


...

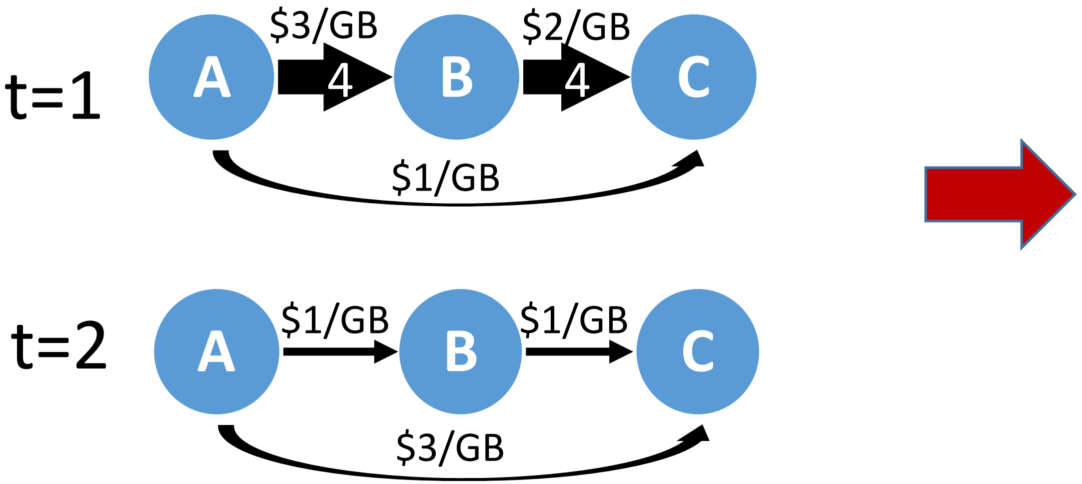
Request: Route 2GB from A to C by t=2

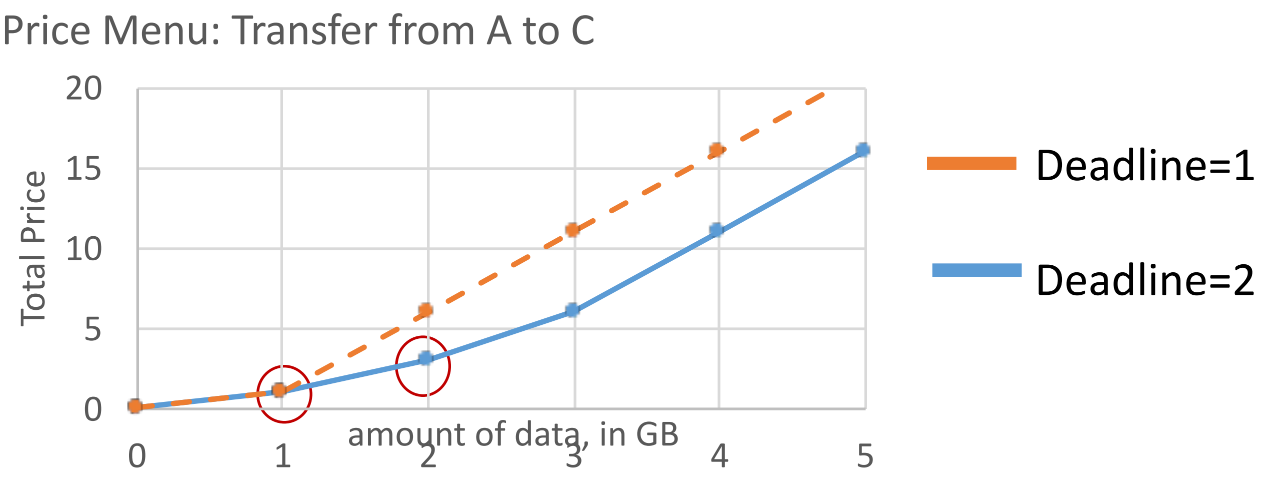
Price quote: $3

Admission Control

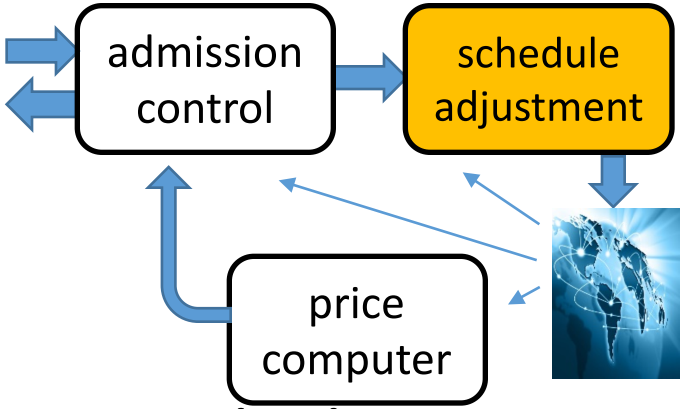


- Interface: User submits request, receives a price quote
  - Presented as a menu of (QoS, price) contracts
  - Pricing indirectly controls admission





Request: Route 2GB from A to C by t=2



Schedule adjustment

Late-binding: transfer is guaranteed at admission, some capacity is reserved into the future, but actual schedule is computed just-in-time

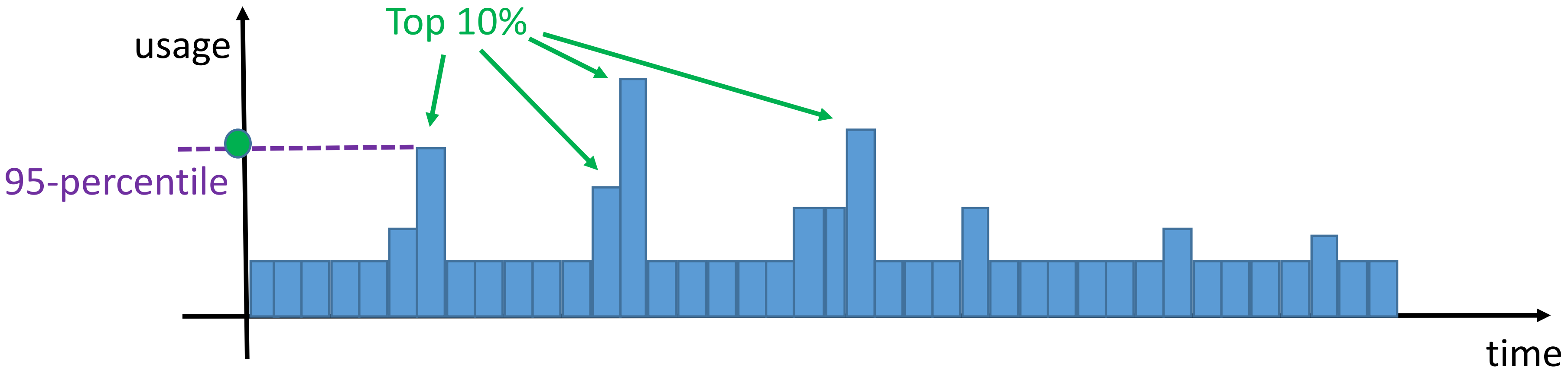
Optimization:

Max	[value*- costs]
s.t.	[satisfy transfer guarantees]
	[respect capacity constraints]

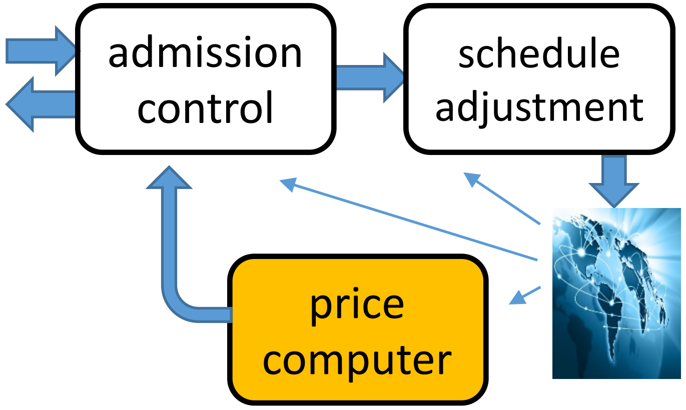
1. Why Max value?
2. Value*: price-per-byte as proxy for value-per-byte
3. Capacity constraints: set aside capacity for high-priority requests

Handling complex costs

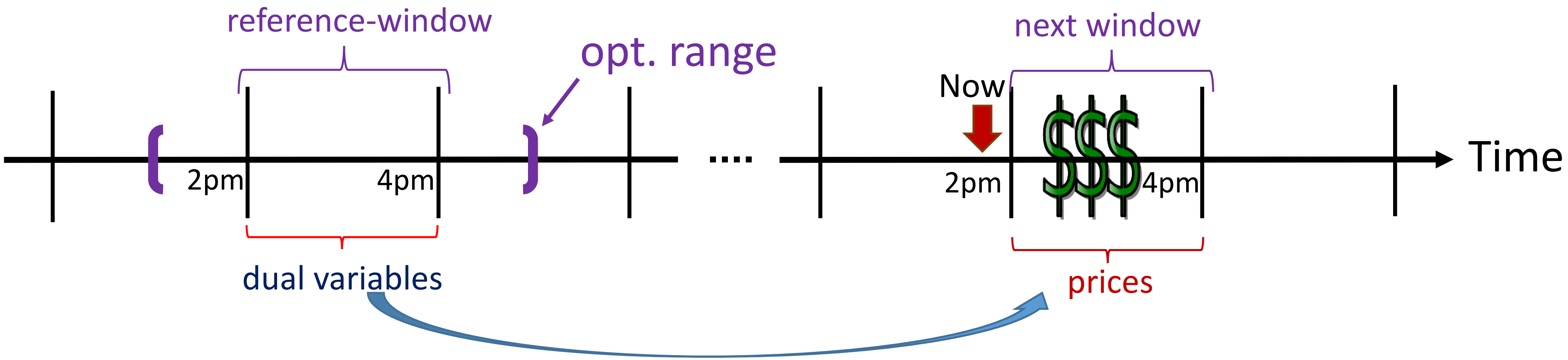
- Recall our objective: Max [value*- **costs**]
- Costs can be non-linear (e.g. 95th percentile usage)
  - Solution: approximate by average top 10% usage
  - Also, can be encoded into linear program (LP) using sorting networks



Price computation



- Update link prices on slow time scale
- Computing optimal prices requires demand forecasting
  - demands are periodic but also some spikes…
- Approach
  - solve offline optimization centered on a reference-window of past requests
  - propose dual variables as prices for next time-window



  - Online adjustments: E.g., increase calculated price in case of link congestion

Incentive Compatibility

Customers will maximize their expected utility by truthfully reporting the parameters of their request

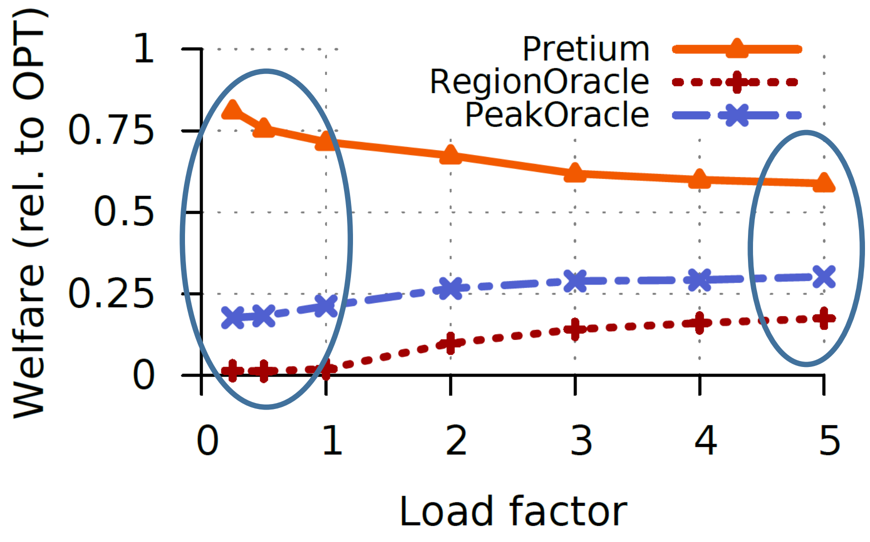
- Formal guarantees require additional technical assumptions
- Even if assumptions do not hold, users do not gain much by misreporting their parameters

Evaluation

- Traffic trace from production inter-DC WAN
  - Network: ~100 nodes, >200 edges
  - Netflow data collected at 5-min intervals
  - Request value-per-byte drawn from random distributions (normal, pareto etc.)
    - value is linear in # bytes transferred

- Compared Pretium to various baselines
  - Offline optimal (OPT)
  - Optimal region-based pricing (RegionOracle)
    - Divide network into regions corresponding to US, Europe, Asia etc
  - Optimal peak/off-peak pricing (PeakOracle)
    - Divide 24hr period into peak and off-peak hours

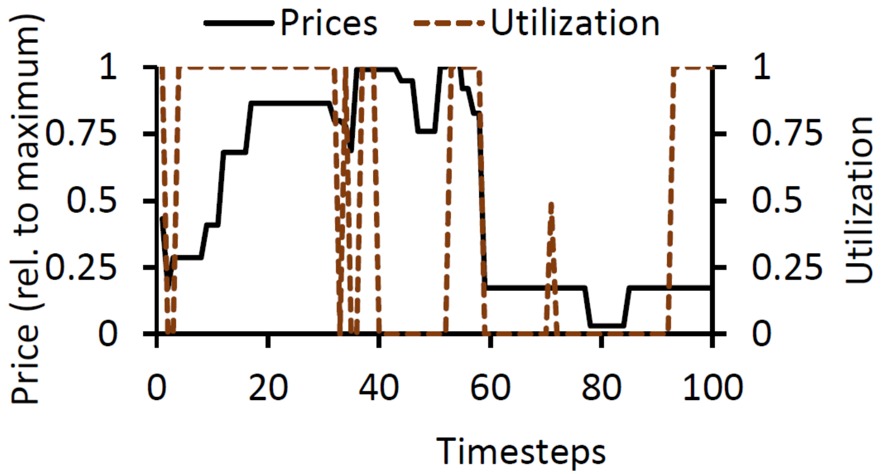
Benefits in welfare



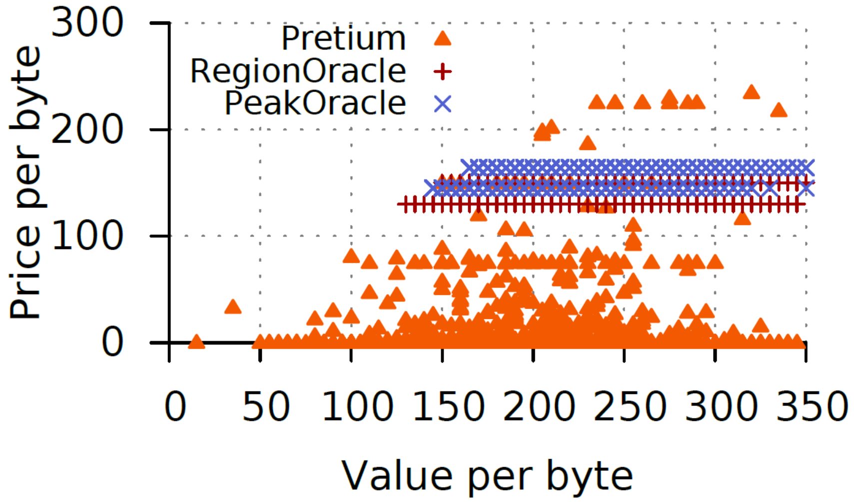
- At low load:
  - RegionOracle, PeakOracle: 1-18% welfare
    - Cannot distinguish low and high-value requests
  - Pretium: ~80% welfare
- At high load:
  - RegionOracle, PeakOracle: 10-30% welfare
    - Better welfare due to more high-value requests
  - Pretium: ~58% welfare
    - Congestion effects…

Why Pretium performs well?

Varying prices based on utilization



Varying prices based on values



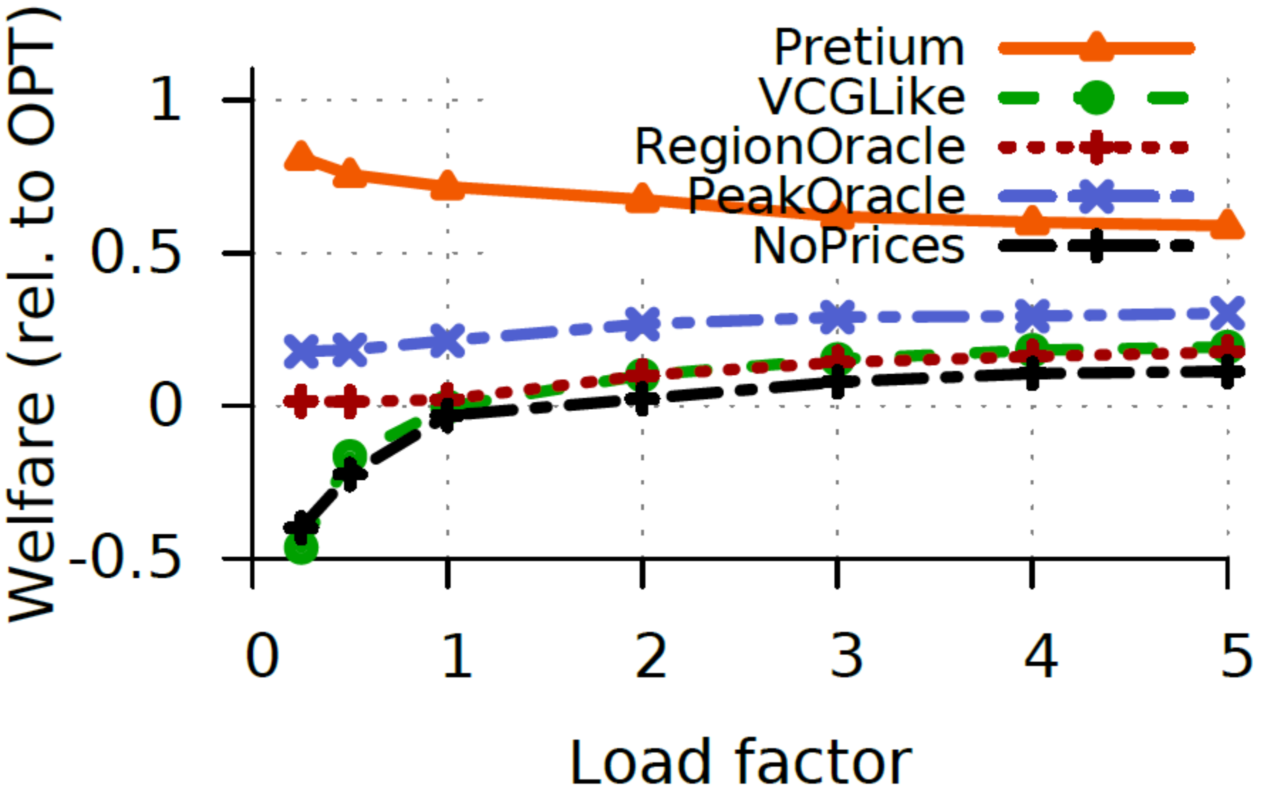
Other results: Pretium reduces peak utilization, break-down of benefits, etc.

Conclusion

- Takeaway: Combine dynamic pricing and traffic engineering
  - Immediate quotes to users with a price (~truthful and supports QoS)
  - Using prices, TE repeatedly solves a linear approximation of the desired goal
  - Periodic (slower time-scale) price adjustment
- Simulations show welfare gains of 30-60% relative to static pricing
- Future work:
  - Explore demand forecasting techniques
  - Investigate non-linear utilities (see BwE [Sigcomm'15])
  - Maximize revenue

Backup slides

Evaluation


Pretium
VCGLike
RegionOracle
PeakOracle
NoPrices
Welfare (rel. to OPT)
1
0.5
0
-0.5
0
1
2
3
4
5
Load factor