Celerity: Towards Low-Delay Multi-Party Conferencing

Minghua Chen

Joint work with
Xiangwen Chen, Baochun Li, Yao Zhao, Yunnan Wu, and Jin Li

The Chinese University of Hong Kong

香港中文大學
The Chinese University of Hong Kong

Multi-party Video Conferencing Are Becoming Popular

Source: Cisco VNI Mobile, 2010

Network Transmission Is the Key Challenge

- How to deliver one party's stream to other parties?
  - High throughput, low delay, no/low loss

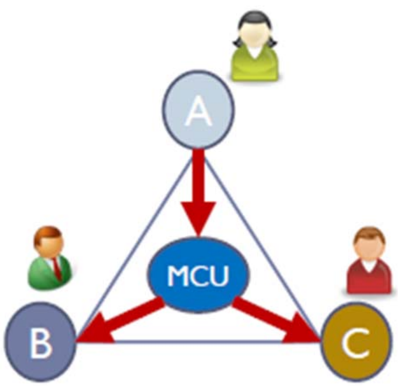
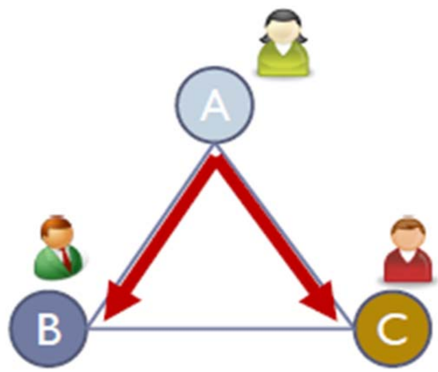
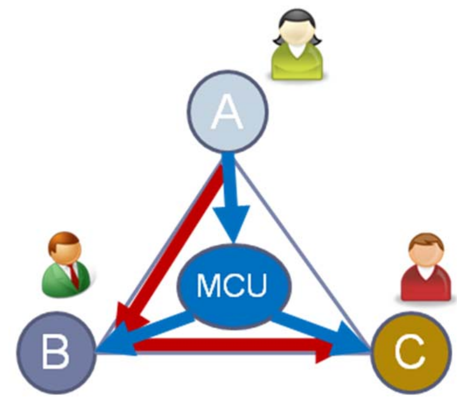
- Challenges:

Real-time conference requires bounded delay

Unknown network topologies

Unpredictable network dynamics

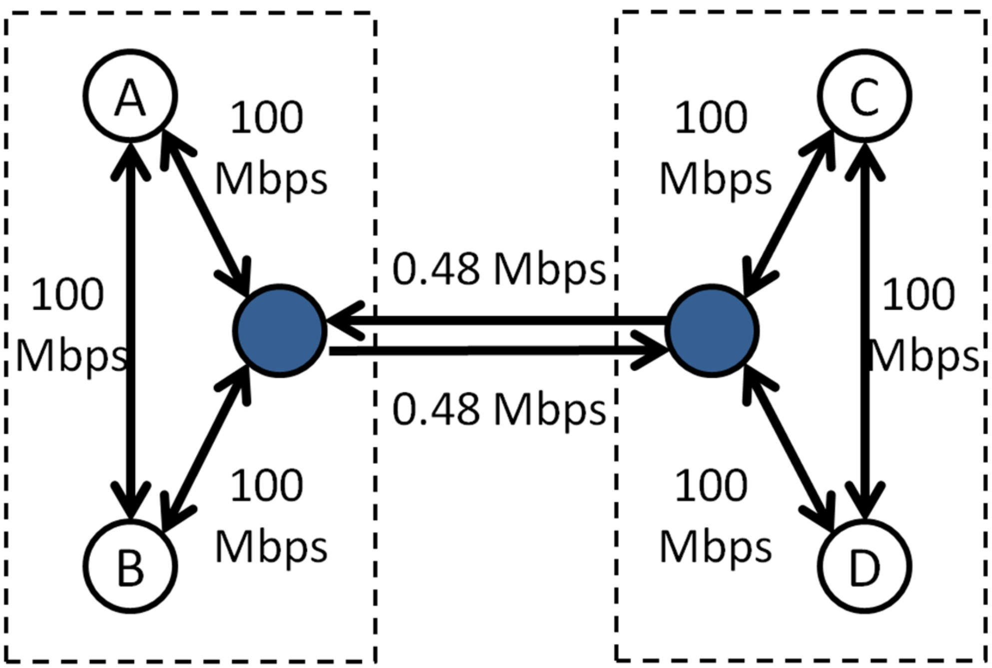
Existing Approaches: Explore only a Limited Design Space

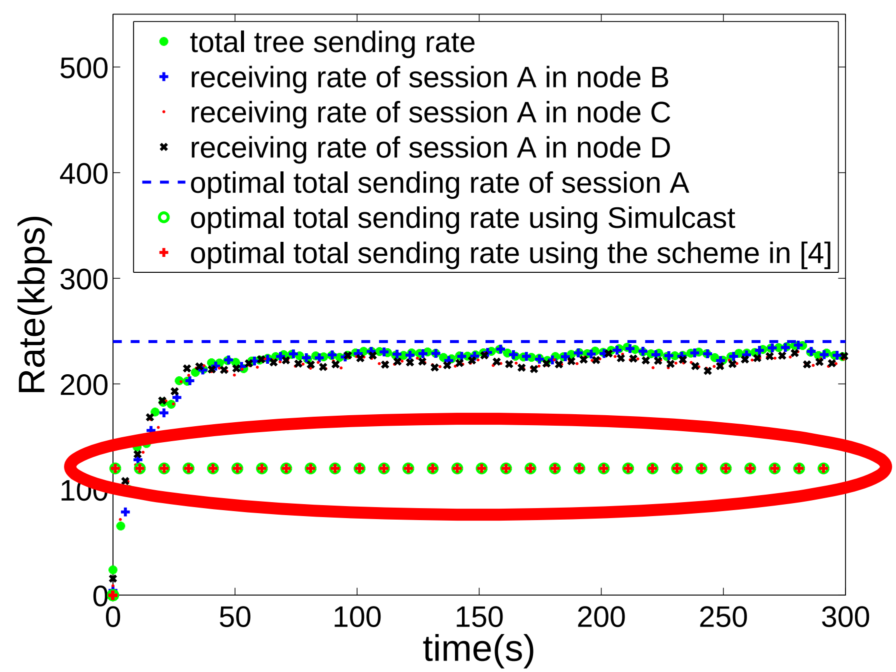
Server –based	**Simulcast**	**Mutualcast**
		
P2P connections are wasted	Slowest peer suffers bad experience	Only optimal when uplinks are the only bottlenecks

[Li et al. 05, Chen et al. 08, Liang et al. 11]

Existing Approaches: Suboptimal Performance

- 4-party conferencing over a "branch-office" topology
  - Simulcast and Mutualcast achieves only half of the optimal

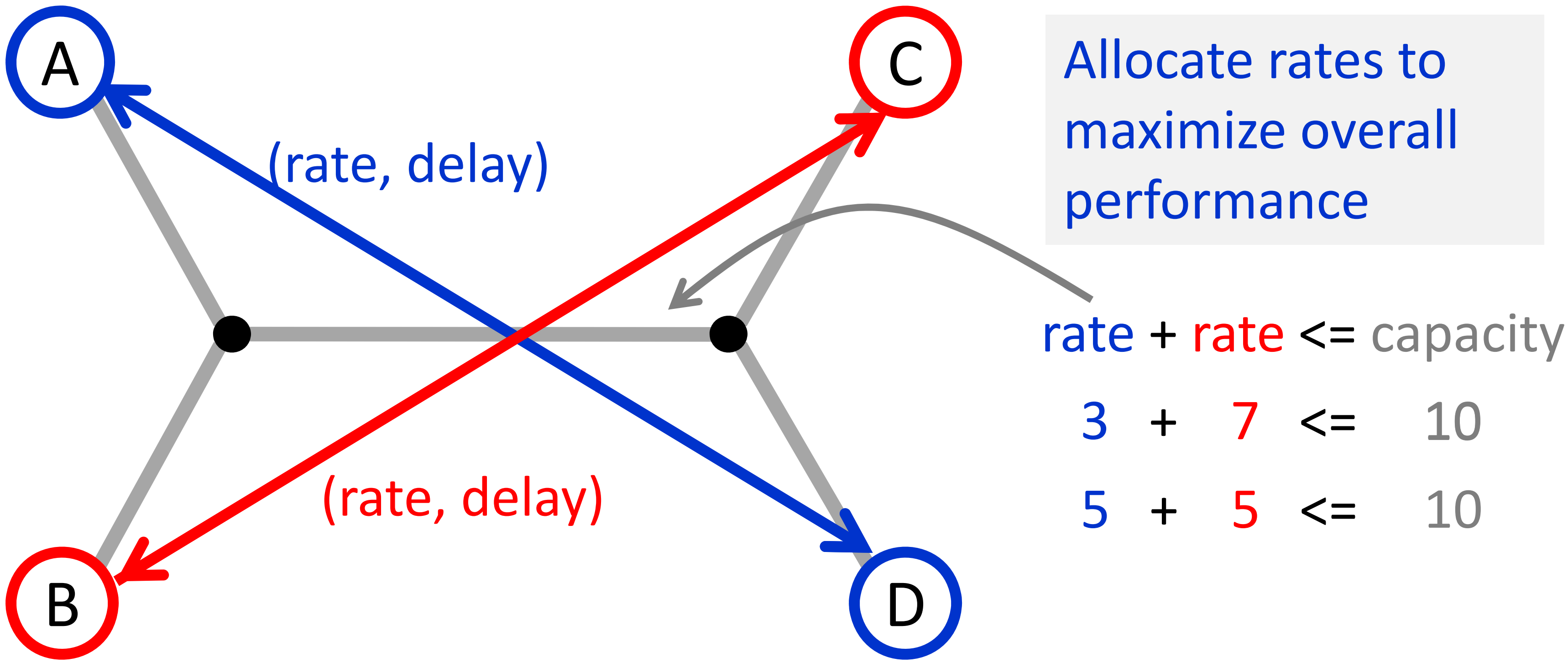




Our Contributions

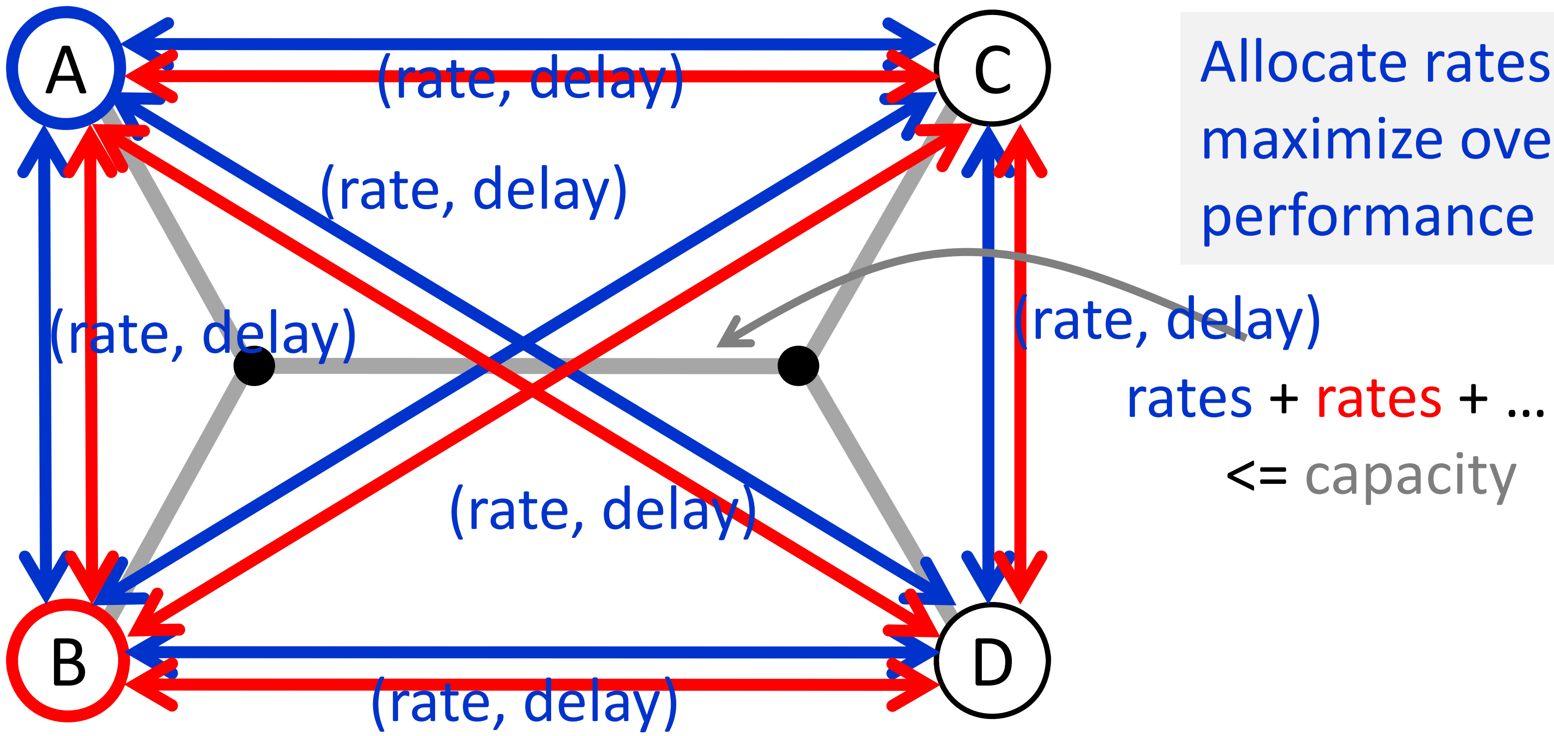
Existing Solutions	Our Theory-Inspired Solution: Celerity
Server-based Expensive in maintaining servers Not scalable, no delay guarantee Poor performance	Adapt to arbitrary network topologies Bounded end-to-end delay
Simulcast Not scalable, no delay guarantee Poorer performance	High throughput
Mutualcast No delay guarantee Optimal if uplinks are the only bottlenecks	Adapt to network dynamic via a distributed control

Two-Party Conferencing: Packing Multiple One-to-One Sessions



- Every source unicasts on a directed overlay link
  - Each overlay link is a TCP/UDP connection, with a “rate” and a “delay”
- Every underlay physical link is shared by one or multiple overlay links

Multi-Party Conferencing as Packing Multiple One-to-Many Sessions



Allocate rates to maximize overall performance

- Every peer broadcasts on an overlay complete graph
  - Each overlay link is a TCP/UDP connection, with a “rate” and a “delay”
- Every underlay physical link is shared by one or multiple overlay links

New Overlay-Link Rate Based Formulation

$$\max_{\boldsymbol{c} \geq 0} \quad \sum_{m=1}^{M} U_m \left(R_m(\boldsymbol{c}_m, D) \right)$$

$$\text{s.t.} \quad \boldsymbol{a}_l^T \left(\boldsymbol{c}_1 + \ldots + \boldsymbol{c}_M \right) \leq C_l, \quad \forall l \in \mathcal{L}$$

Sum of videos' PSNRs

Routing vector of overlay traffics

Underlay' link capacity constraints

Underlay link capacity

– c_m: overlay-link rate vector for session m

– R_m (c_m, D): delay-bounded throughput for session m

New Overlay-Link Rate Based Formulation

$$\max_{\boldsymbol{c}\geq 0} \quad \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{c}_m, D)\right)$$

$$\text{s.t.} \quad \boldsymbol{a}_l^T\left(\boldsymbol{c}_1 + \ldots + \boldsymbol{c}_M\right) \leq C_l, \quad \forall l \in \mathcal{L}$$

– **Design problem**: Given D, *distributedly* find R and c to maximize the sum PSNR

No need to know routing vectors **a** and capacities **C**

What's New?

- Allow bottleneck to be anywhere in the network
  - Go beyond a common assumption that peer uplinks are the only capacity bottlenecks

- Allow systematically exploring design space beyond existing solutions
  - Go beyond using one-to-one networking components as primitives (Skype, Simulcast…)

Architectural Insights

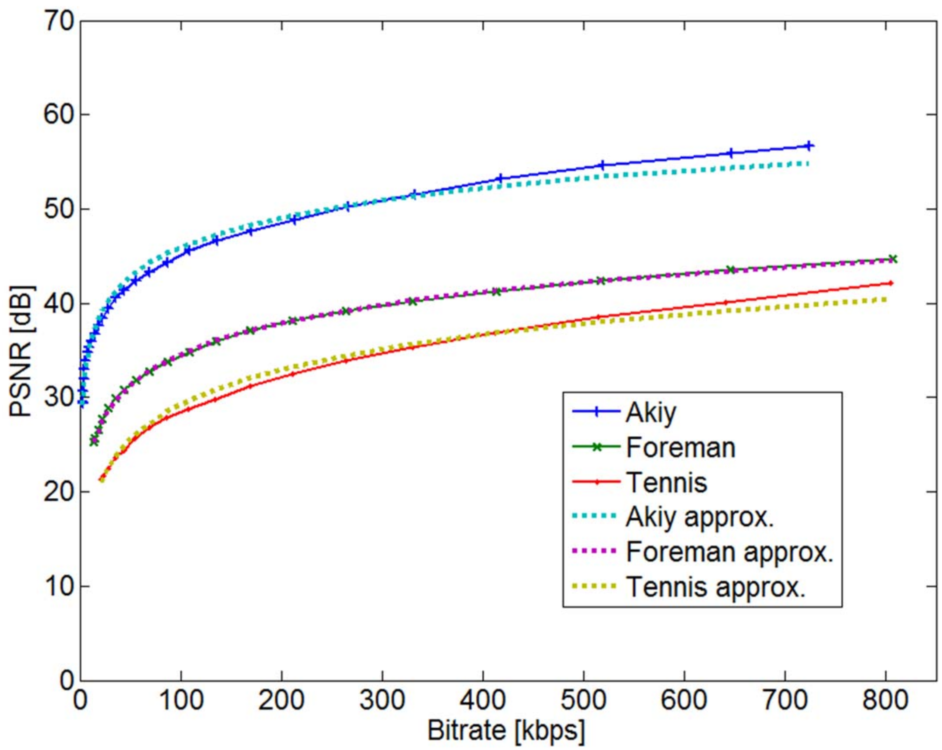
How to model PSNR as a utility function?

How to achieve high delay-bounded throughput R_m (c_m, D), given D and c_1, …, c_M?

How to obtain/optimize c_1, …, c_M?

Modeling PSRN by a Log Utility Function

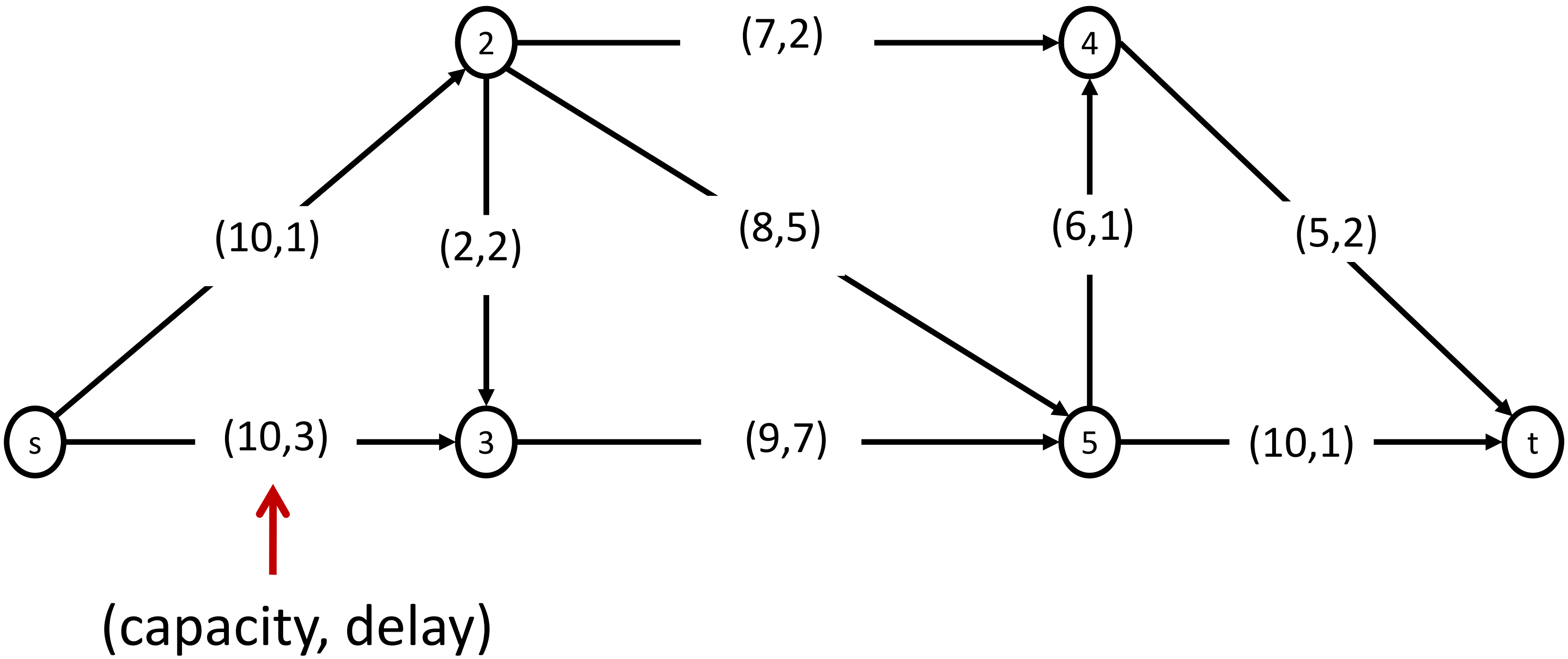
- PSNR of a video stream coded at a rate R can be approximated by β log R + z [Chen et al. 08]



Achieving Delay-Bounded Capacity Is Hard

- Given $c_1, \ldots, c_M$, achieving the optimal delay-bounded throughput is NP-complete
  - Even determining the achievability of a unicast rate is NP-complete (length-constrained max-flow) [R92]

Max-Flow Subject to Delay Bound



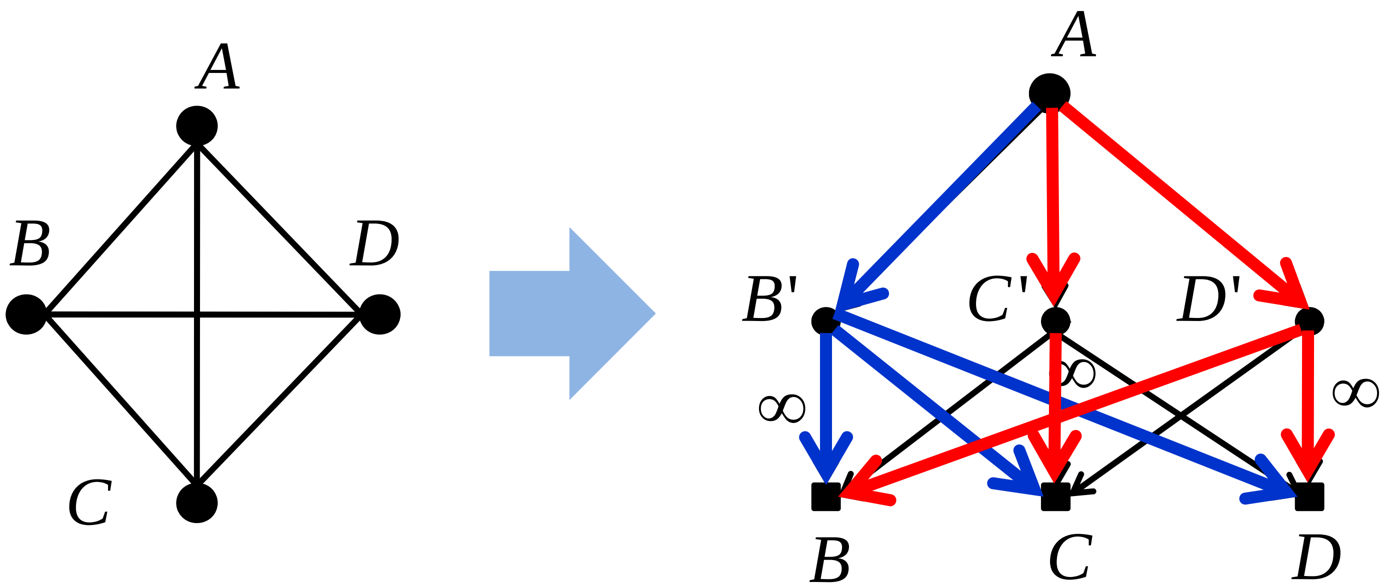
Can node s delivers data to node t at rate 14, subject to an end-to-end delay bound of 14?

Achieving Delay-Bounded Capacity Is Hard

- Given $c_1, \ldots, c_M$, achieving the optimal delay-bounded throughput is NP-complete
  - Even determining the achievability of a unicast rate is NP-complete (length-constrained max-flow) [R92]
- For unicast (i.e., 2-party conferencing) scenario
  - Polynomial-time algorithm to achieve (1-\epsilon) to the optimal of routing [KL02, NDSL-TR-11]
- Broadcast and multicast scenarios: largely open
- Role of network coding: open

Achieving High Delay-Bounded Throughput

- We take a practical approach to pack 2-hop delay-bounded spanning trees
  - Propose a greedy polynomial-time algorithm
  - Achieve min-cut over a delay-bounded sub-graph



Distributed Algorithm to Optimize $c_1, \ldots, c_M$

$$\max_{\boldsymbol{c}\geq 0} \quad \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{c}_m, D)\right)$$

$$\text{s.t.} \quad \boldsymbol{a}_l^T \boldsymbol{y} \leq C_l, \quad \forall l \in \mathcal{L}; \boldsymbol{y} = \boldsymbol{c}_1 + \ldots + \boldsymbol{c}_M$$

Shadow price of using underlay link l

$$\mathcal{G}\left(\boldsymbol{c}, \boldsymbol{p}\right) \triangleq \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{c}_m)\right) - \sum_{l\in\mathcal{L}} \int_0^{\boldsymbol{a}_l^T \boldsymbol{y}} q_l(z)\, dz - \sum_{l\in\mathcal{L}} p_l\left(\boldsymbol{a}_l^T \boldsymbol{y} - C_l\right)$$

Penalty of violating underlay link l's capacity constraint

- A non-strictly concave optimization problem
- Saddle points of $\mathcal{G}\left(\boldsymbol{c}, \boldsymbol{p}\right)$ is the optimal solution

A Loss-Delay Based Primal-Dual Algorithm

Rate of session m on overlay-link e

Queuing delay of overlay-link e

$$\dot{c}_{m,e} = \alpha \left[U'_m(R_m) \frac{\partial R_m}{\partial c_{m,e}} - \sum_{l \in e} p_l - \beta \sum_{l \in e} q_l \right]^+_{c_{m,e}}$$

Incentive to increase the overlay-link rate

Packet loss rate of overlay-link e

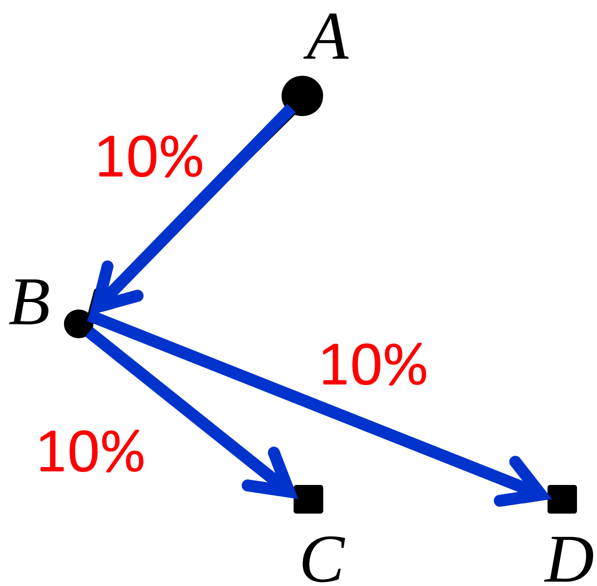
- All sufficient statistics can be obtained decentralized-ly
- Allow Celerity to adapt to unknown network topologies and unpredictable dynamics

Convergence Property of the Algorithm

$$\dot{c}_{m,e} = \alpha \left[U'_m(R_m) \frac{\partial R_m}{\partial c_{m,e}} - \sum_{l \in e} p_l - \beta \sum_{l \in e} q_l \right]^+_{c_{m,e}}$$

- For non-strictly concave problem, convergence of Primal-dual algorithm is an open problem [V05, Chen et al. 08]
- **Theorem**: the time-average Lagrange function value converge to the saddle point (optimal solution) within a gap of $\max\left(\alpha, \max_{l \in \mathcal{L}} C_l^{-1}\right) \frac{\Delta^2}{2}$.
- Allow different step sizes, beyond the result in [NO09]
  - Critical for p_l to be interpreted as queuing delay
- Exact convergence is still open

Network Coding for Speedy Loss-Recovery



- All links have unit capacity with packet loss rate (%)
- Routing only
  - Broadcast rate = 0.81
- Routing + retransmission
  - Broadcast rate = 0.9
  - Incur retransmission delay
- ✔ Network coding
  - Broadcast rate = 0.9
  - No retransmission delay

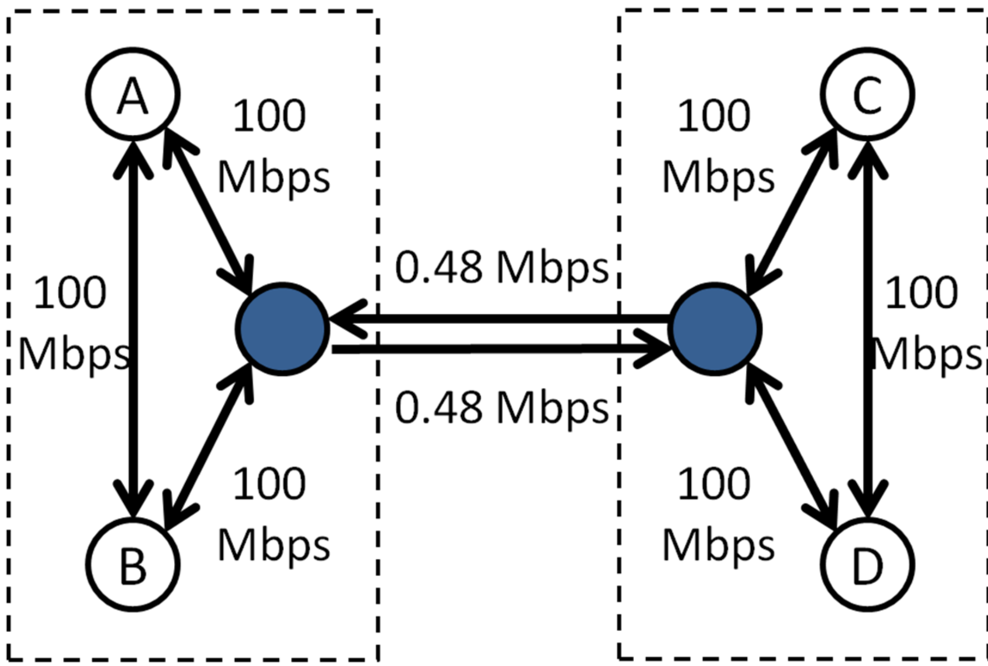
Big Picture

- □ Multi-party conferencing as packing multiple one-to-many sessions
- □ New overlay-link based formulation
- □ Theory-inspired solution design with practical concerns taken into account
- □ Algorithm design with performance guarantee

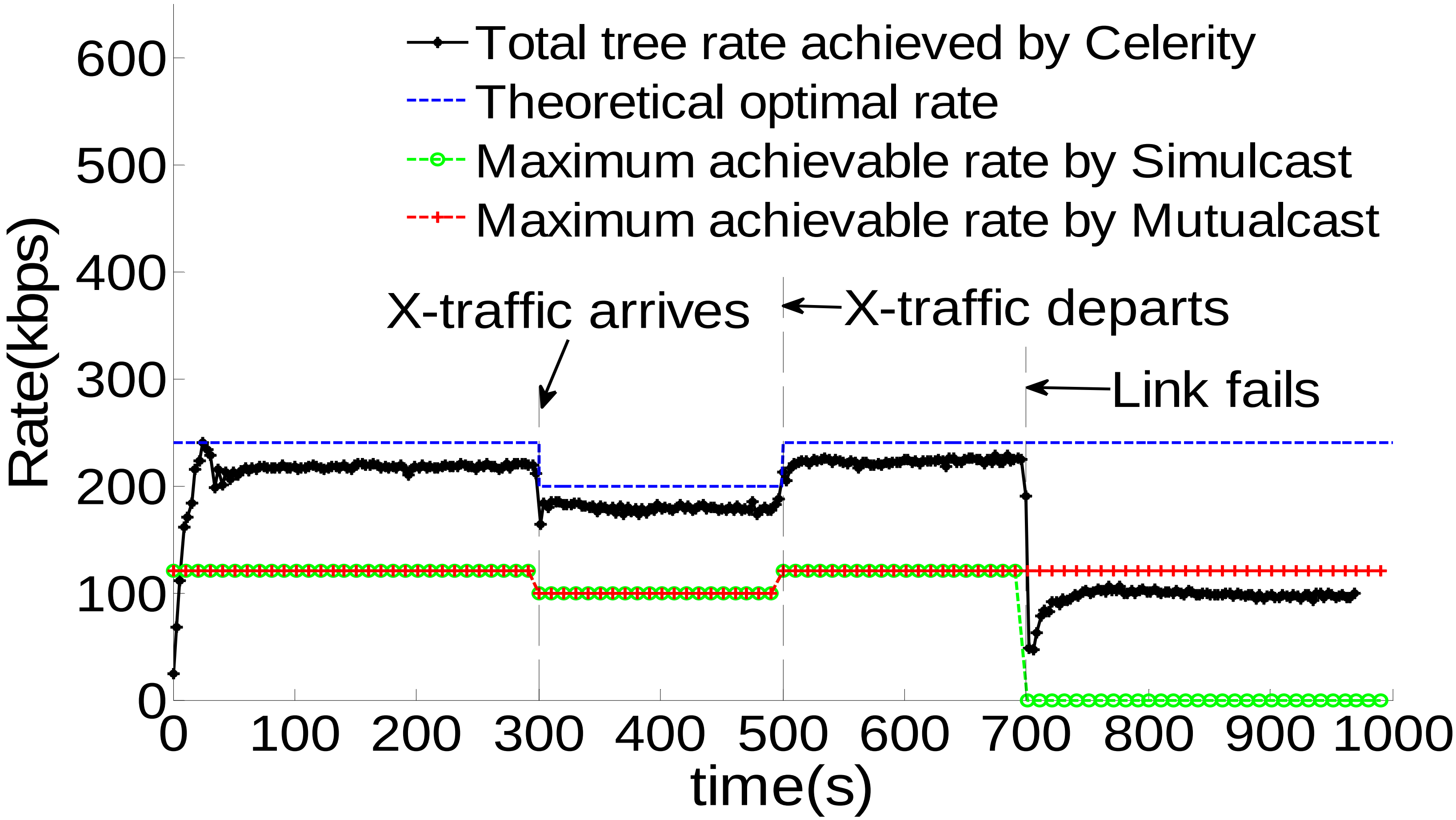
- □ How does the solution perform in practice?

Local Experiment

- Implement proto-types of Celerity, Simulcast, and Mutualcast in C++ and run on Win7 PCs
- Purposes
  - To see whether the system works as expected
  - To stress-test the system

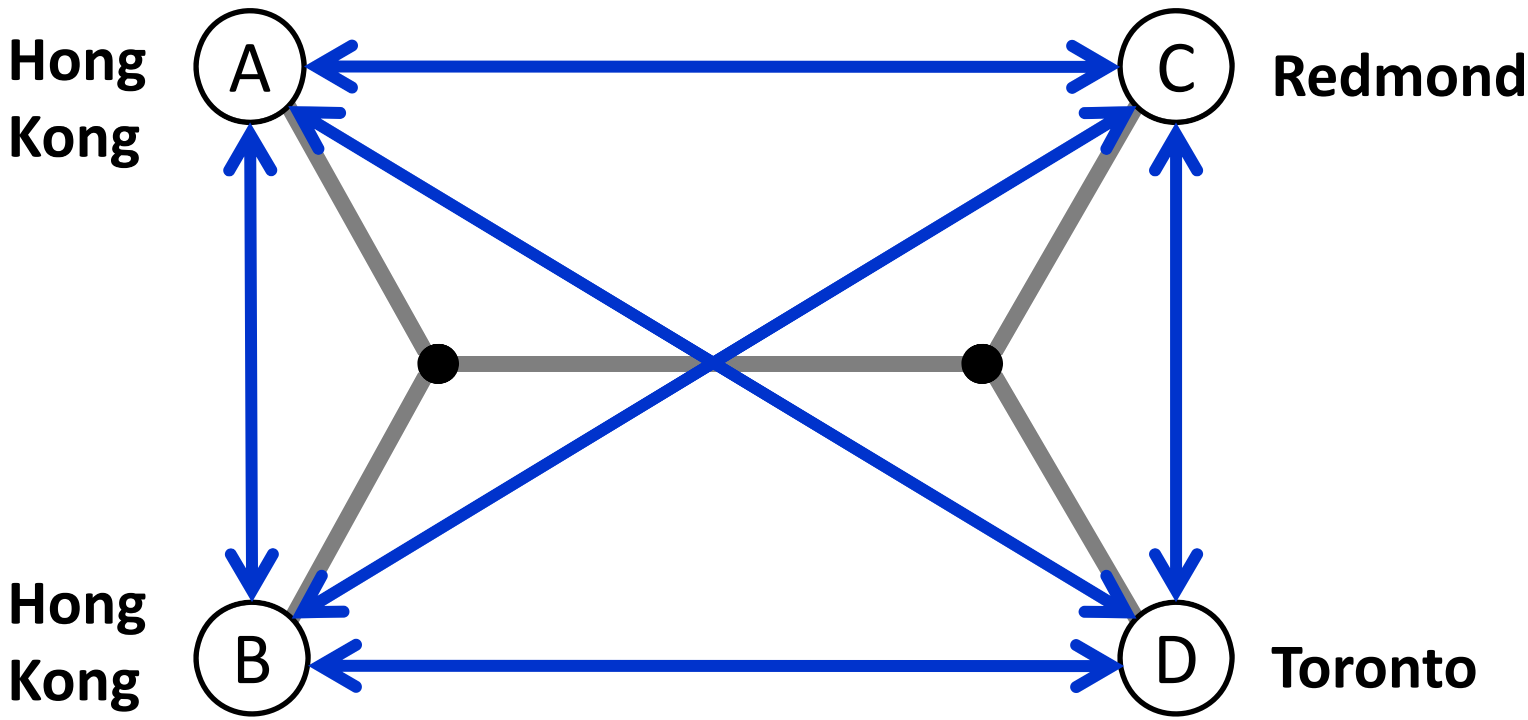


Local Experiment Results: Rate Performance of Node A

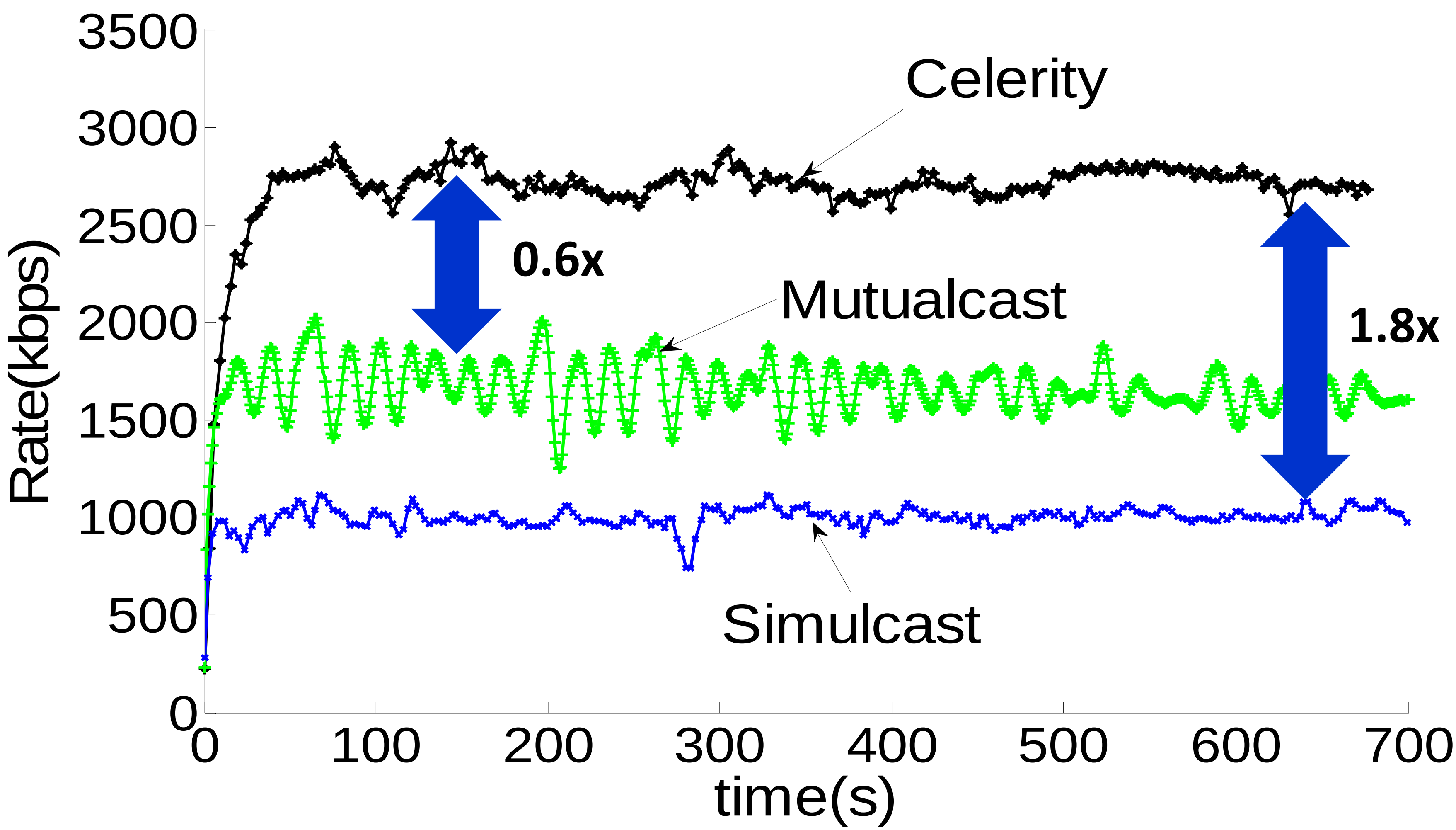


Internet Experiment

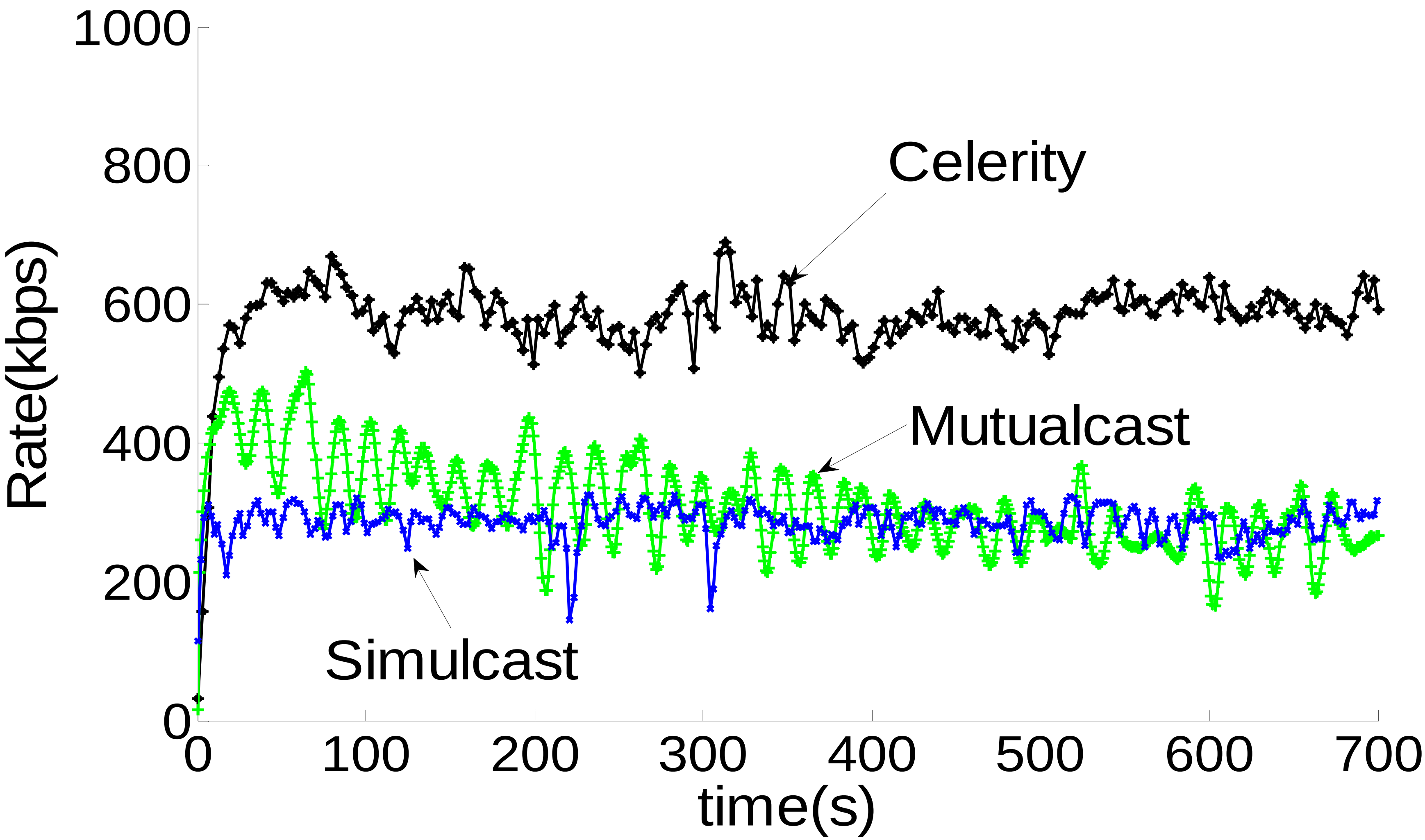
- 4-party conferencing across 3 countries and two continents



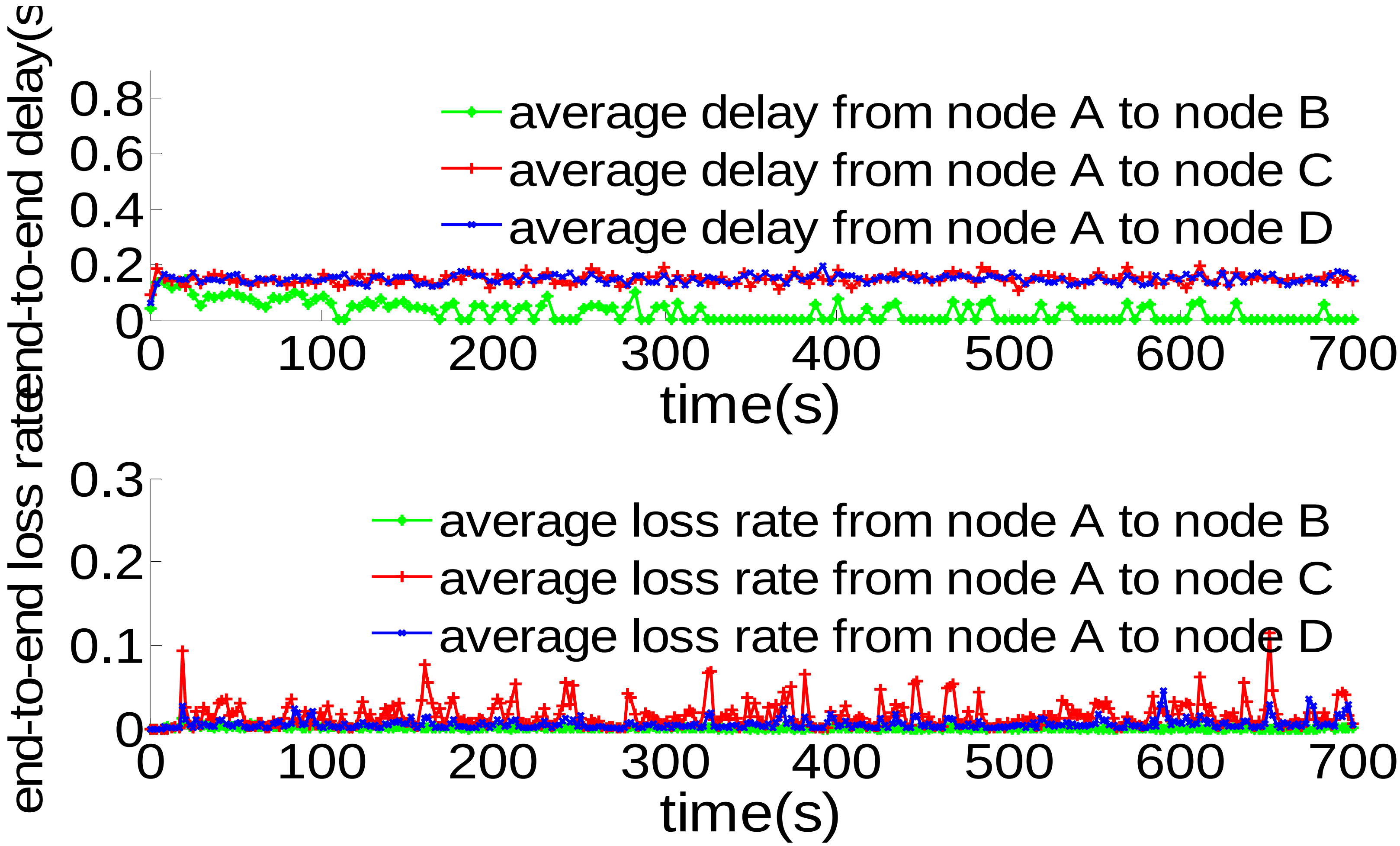
Celerity Outperform Existing Solutions



Rates of Session A

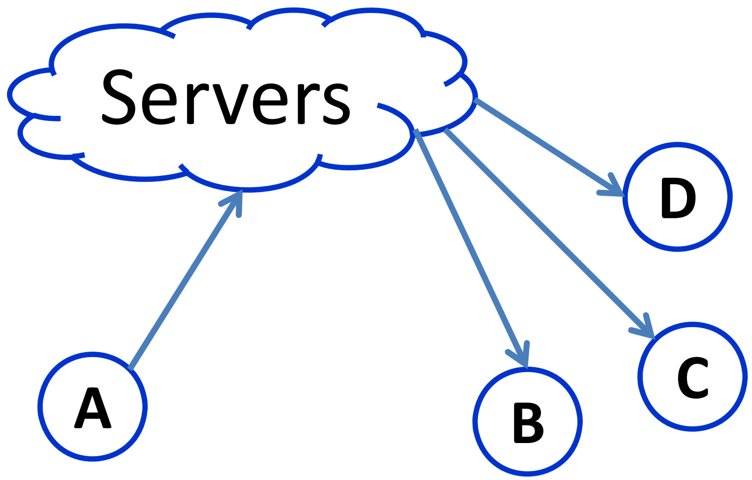


Queuing Delay and Loss Rate from Node A to Other Nodes

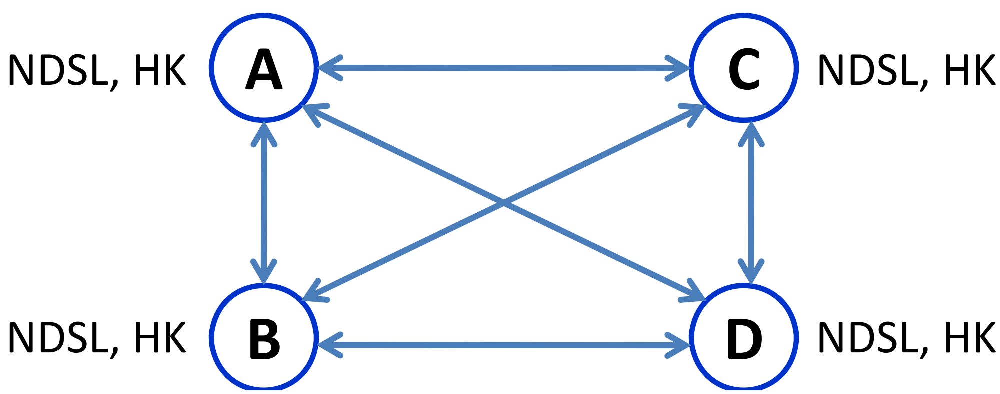


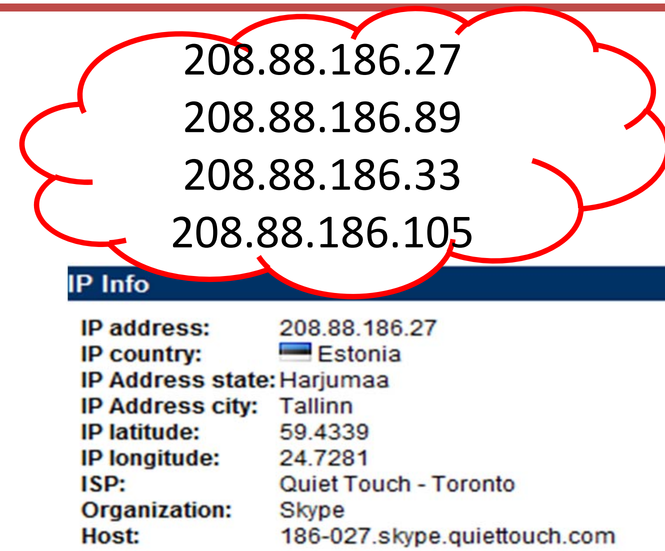
Compare Celerity with Skype

Skype is a server-based solution

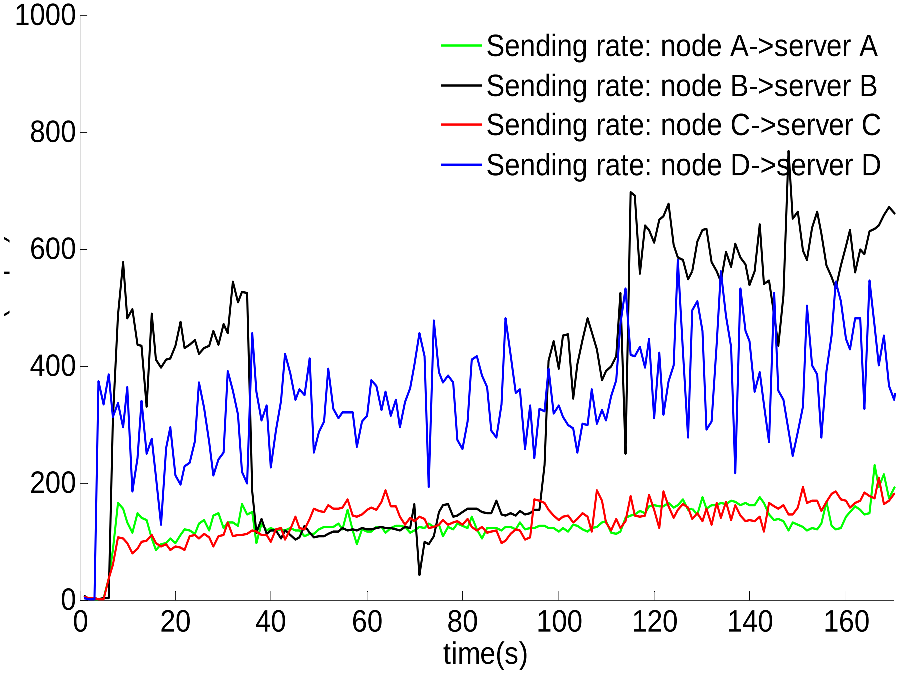


Experiment Setting for Comparison





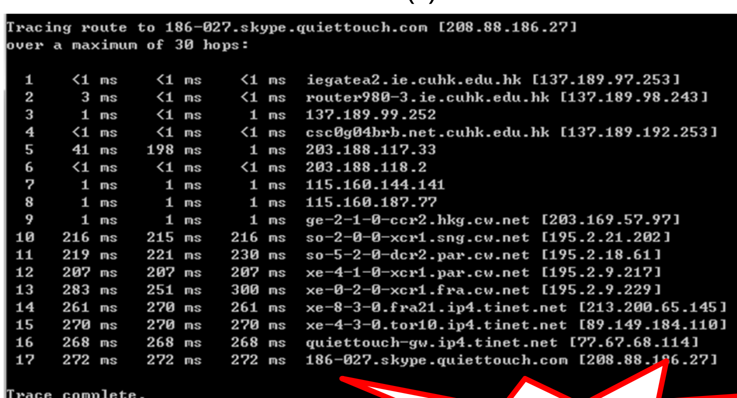
Skype's Results



Low throughput

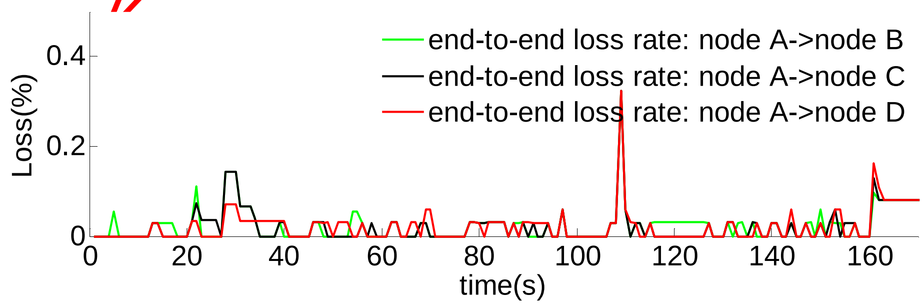
Long delay

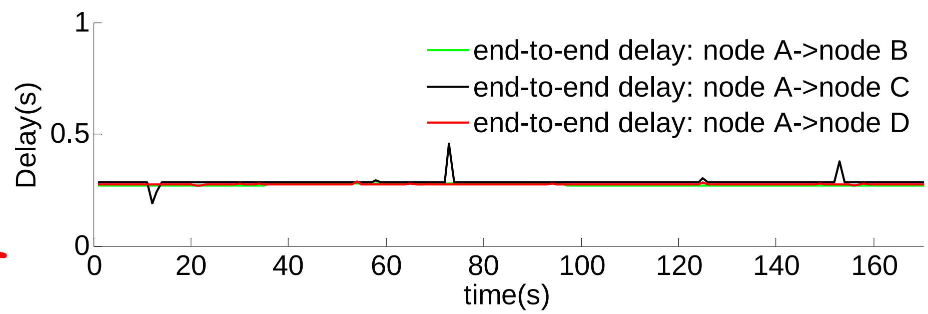
High loss rate



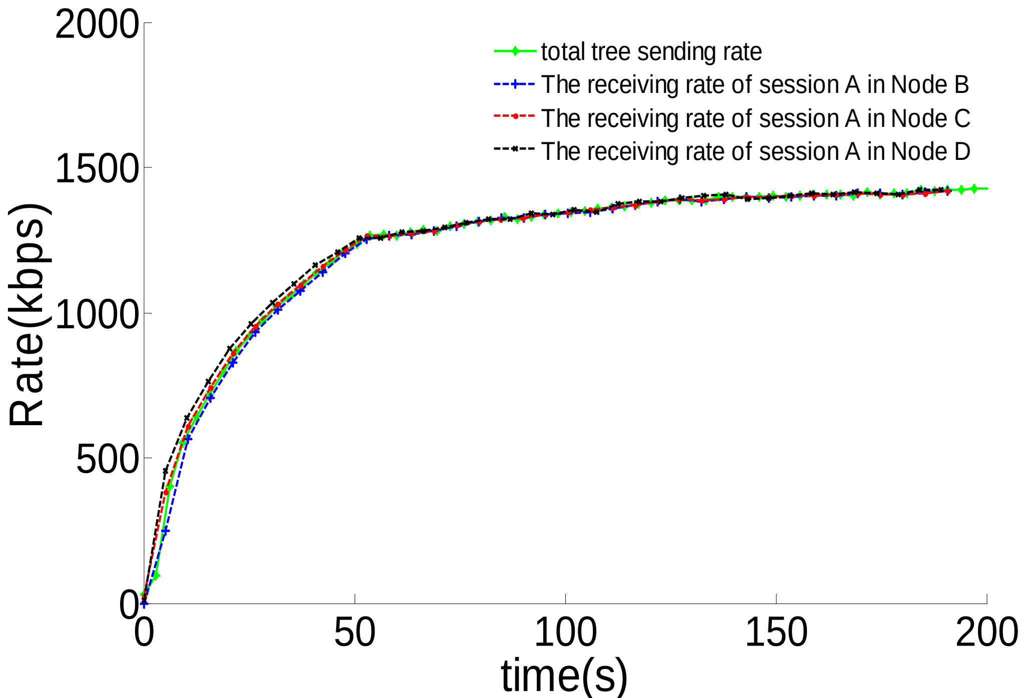
```
Tracing route to 186-027.skype.quiettouch.com [208.88.186.27]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  iegatea2.ie.cuhk.edu.hk [137.189.97.253]
  2     3 ms    <1 ms    <1 ms  router980-3.ie.cuhk.edu.hk [137.189.98.243]
  3     1 ms    <1 ms     1 ms  137.189.99.252
  4    <1 ms    <1 ms    <1 ms  csc0g04brb.net.cuhk.edu.hk [137.189.192.253]
  5    41 ms   198 ms     1 ms  203.188.117.33
  6    <1 ms    <1 ms    <1 ms  203.188.118.2
  7     1 ms     1 ms     1 ms  115.160.144.141
  8     1 ms     1 ms     1 ms  115.160.187.77
  9     1 ms     1 ms     1 ms  ge-2-1-0-ccr2.hkg.cw.net [203.169.57.97]
 10   216 ms   215 ms   216 ms  so-2-0-0-xcr1.sng.cw.net [195.2.21.202]
 11   219 ms   221 ms   230 ms  so-5-2-0-dcr2.par.cw.net [195.2.18.61]
 12   207 ms   207 ms   207 ms  xe-4-1-0-xcr1.par.cw.net [195.2.9.217]
 13   283 ms   251 ms   300 ms  xe-0-2-0-xcr1.fra.cw.net [195.2.9.229]
 14   261 ms   270 ms   261 ms  xe-8-3-0.fra21.ip4.tinet.net [213.200.65.145]
 15   270 ms   270 ms   270 ms  xe-4-3-0.tor10.ip4.tinet.net [89.149.184.110]
 16   268 ms   268 ms   268 ms  quiettouch-gw.ip4.tinet.net [77.67.68.114]
 17   272 ms   272 ms   272 ms  186-027.skype.quiettouch.com [208.88.186.27]

Trace complete.
```

17 hops

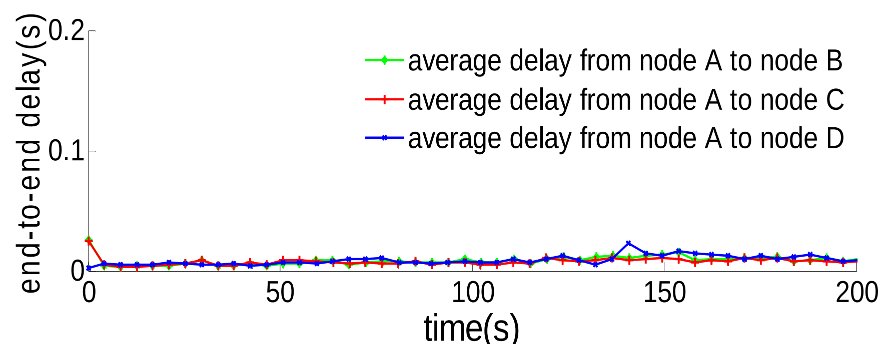


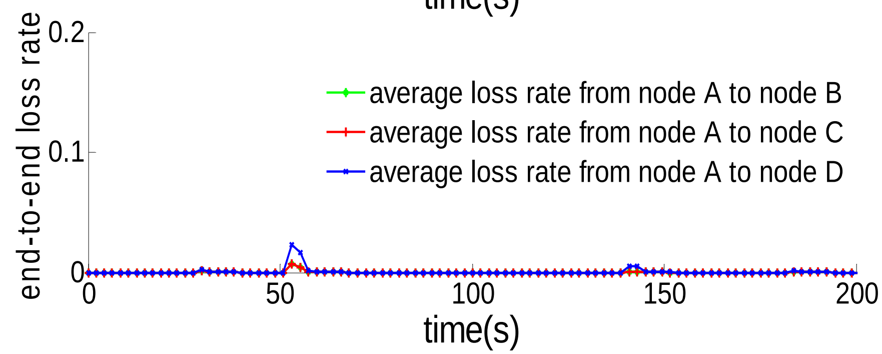


Celerity's Result



Higher throughput

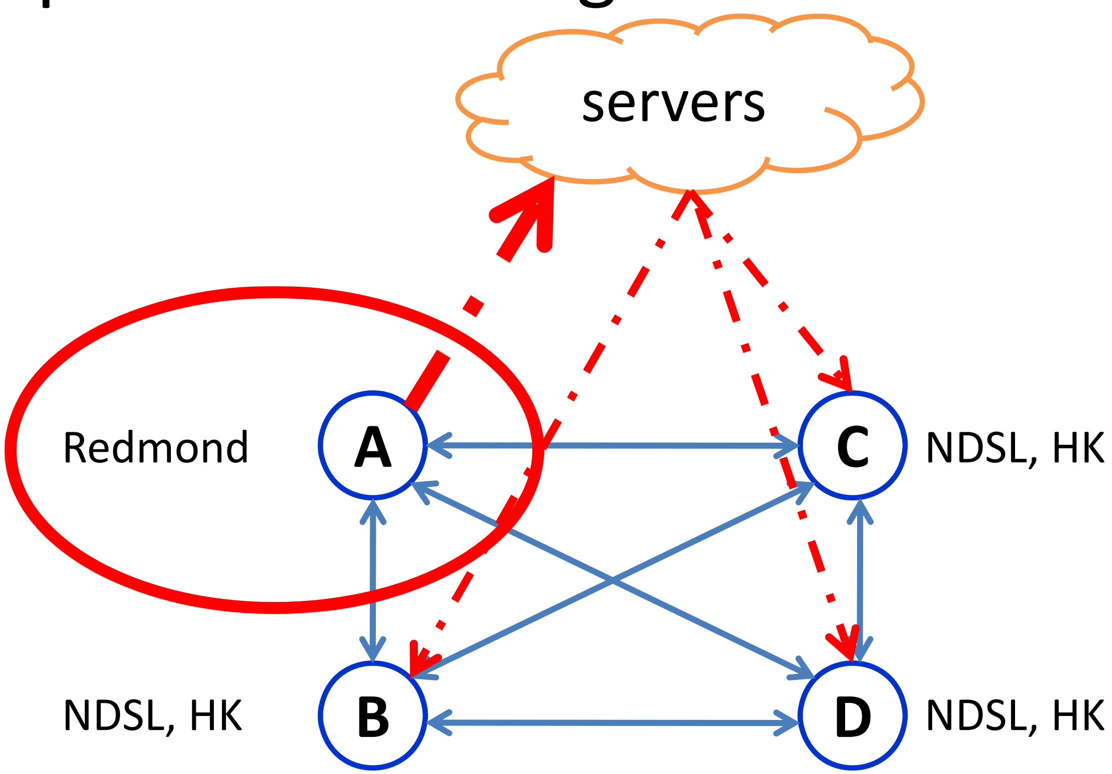




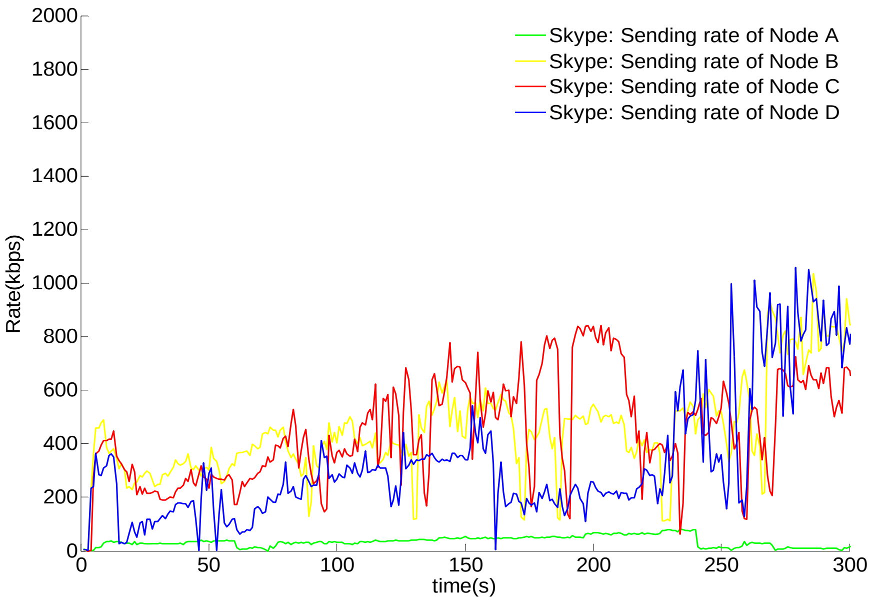
Low Loss and Delay

Compare Celerity with Skype (II)

- Experiment Setting

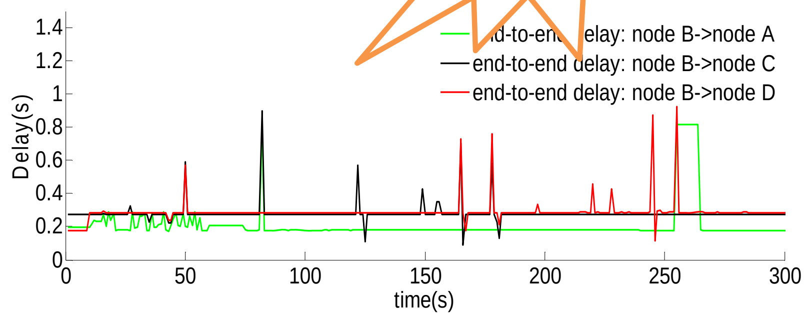


Skype's Results

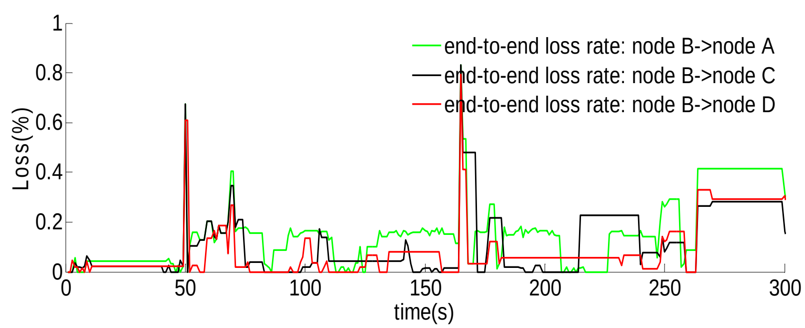


Low throughput

Large oscillation

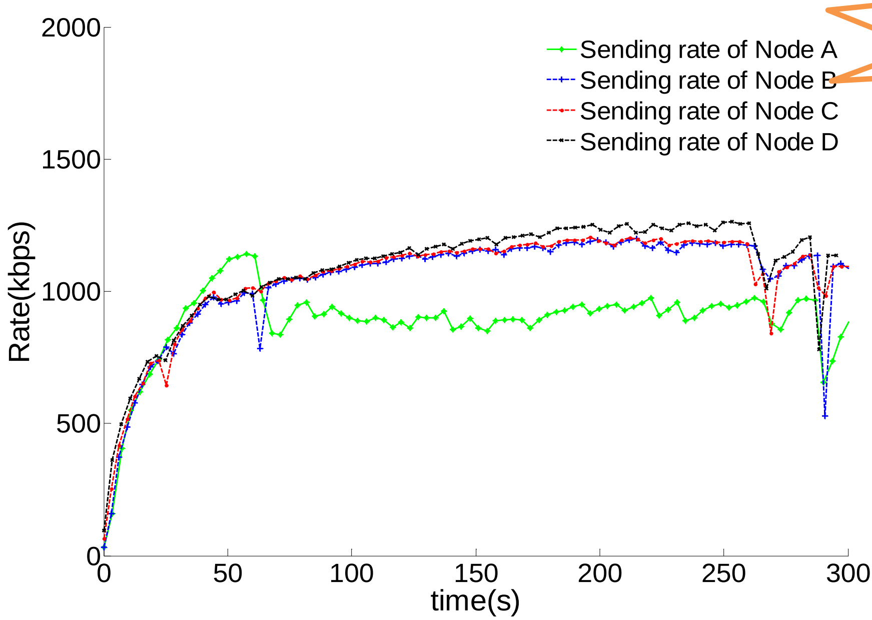


High loss rate and delay

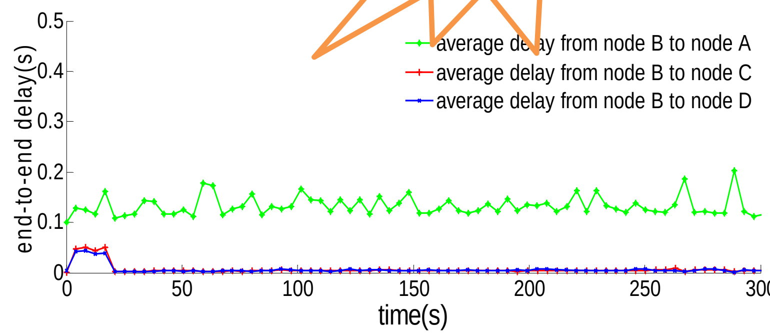


Celerity's Results

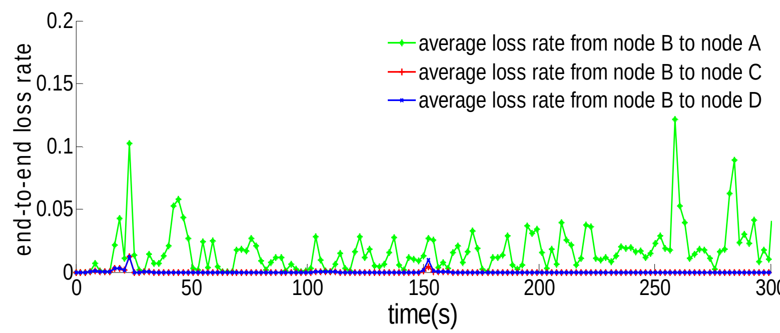
High throughput



Small oscillation



Low loss rate and delay

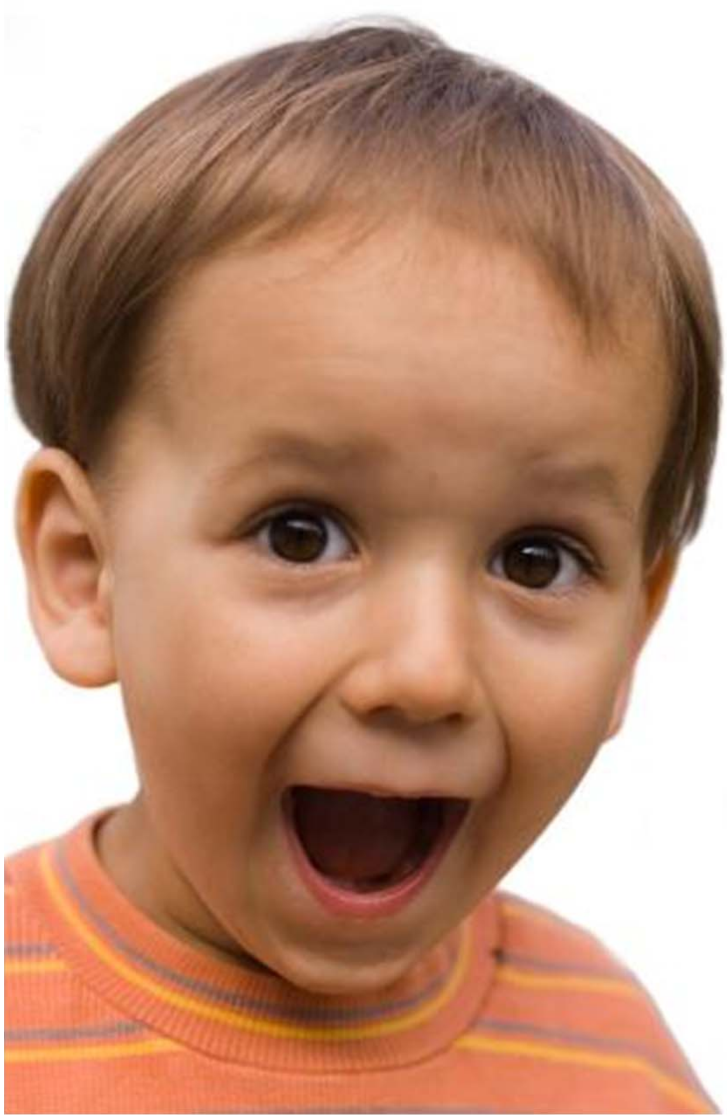


It Is a Beginning Rather than an End

- We reconsider the design space of multi-party conferencing, and present our theory-inspired solution Celerity
  - Guarantee bounded delay
  - Achieve high throughput
  - Adapt to arbitrary network topology and dynamics

- Internet experiments show significant performance gain over existing solutions

- **On-going**: build a real system; study open problems



Thank you

Minghua Chen (minghua@ie.cuhk.edu.hk)

http://www.ie.cuhk.edu.hk/~mhchen