Elasticity Detection:

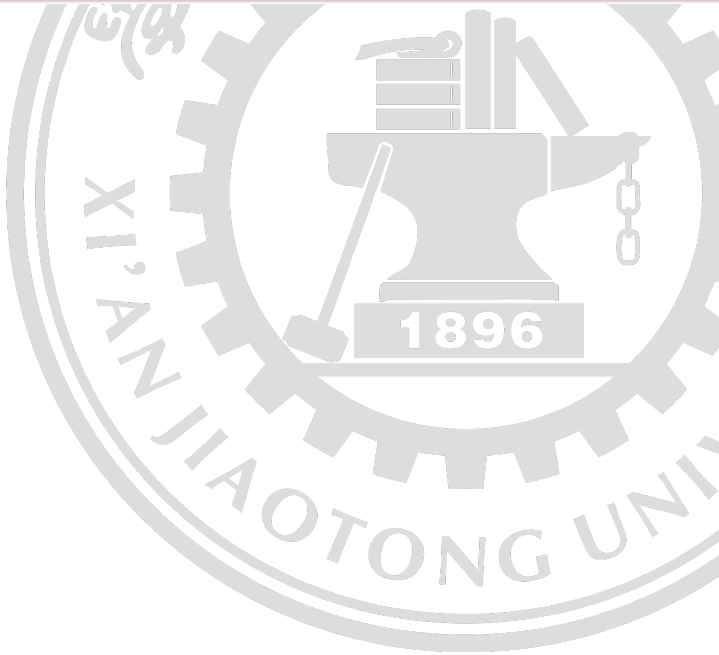
A Building Block for Internet Congestion Control[4]

King-Siong Si

Institute of Multimedia Knowledge Fusion and Engineering,
Xi'an JiaoTong University

sjsinx@stu.xjtu.edu.cn

January 24, 2024



To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay
  - Examples: Cubic, NewReno, etc.

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay
  - Examples: Cubic, NewReno, etc.
  - Compete well for throughput.

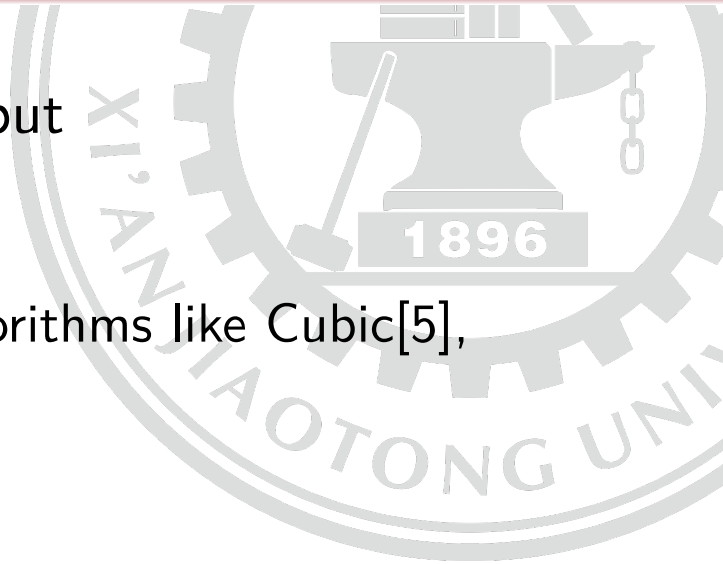
To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay
  - Examples: Cubic, NewReno, etc.
  - Compete well for throughput.
  - High queueing delay even when cross-traffic is not aggressive

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay
  - Examples: Cubic, NewReno, etc.
  - Compete well for throughput.
  - High queueing delay even when cross-traffic is not aggressive
- Information about cross traffic can improve CC.

To Be, Or Not to Be?



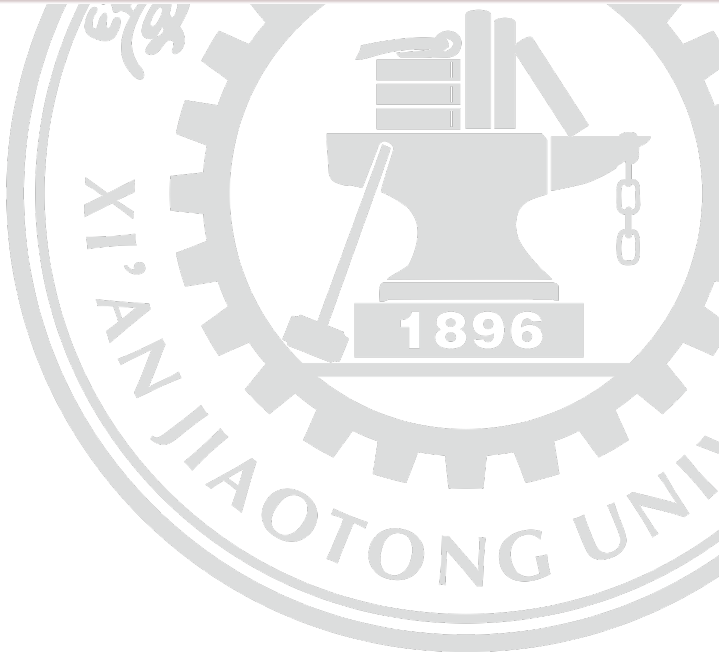
- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay
  - Examples: Cubic, NewReno, etc.
  - Compete well for throughput.
  - High queueing delay even when cross-traffic is not aggressive
- Information about cross traffic can improve CC.
  - Adjust CC behavior based on the nature of cross traffic.

To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay
  - Examples: Cubic, NewReno, etc.
  - Compete well for throughput.
  - High queueing delay even when cross-traffic is not aggressive
- Information about cross traffic can improve CC.
  - Adjust CC behavior based on the nature of cross traffic.
  - Use Cubic when cross traffic is competing.

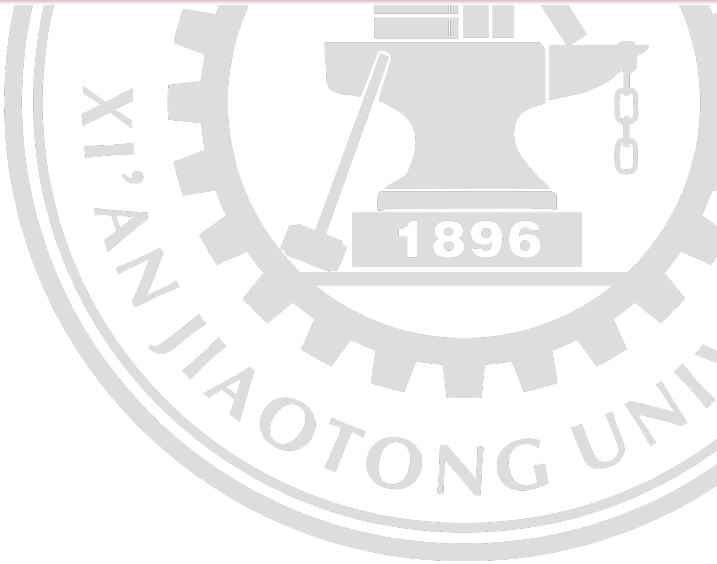
To Be, Or Not to Be?

- Delay-Based Algorithms: Low Throughput
  - Examples: Vegas[2], Copa[1], etc.
  - Achieve lower delays.
  - Poor throughput against deployed algorithms like Cubic[5], NewReno[6], BBR[3], etc.
- Deployed Algorithms: High Delay
  - Examples: Cubic, NewReno, etc.
  - Compete well for throughput.
  - High queueing delay even when cross-traffic is not aggressive
- Information about cross traffic can improve CC.
  - Adjust CC behavior based on the nature of cross traffic.
  - Use Cubic when cross traffic is competing.
  - Use delay-based CC algorithms otherwise.



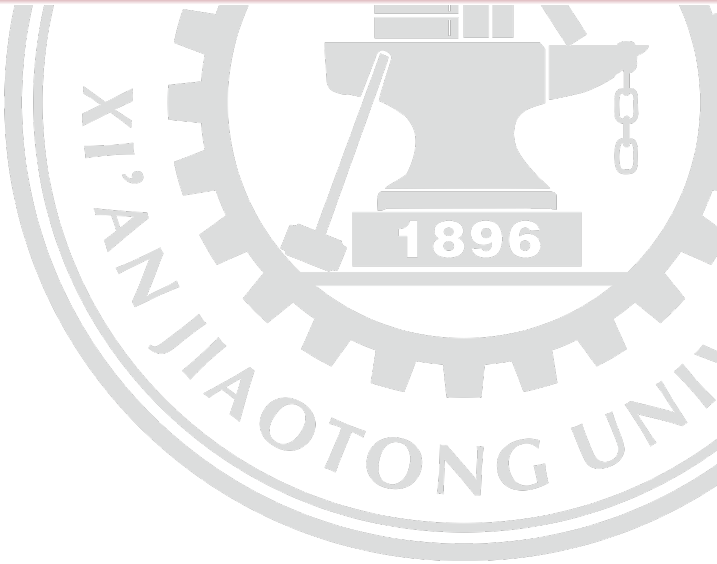
Qeustions

- Question 1: What kind of information?



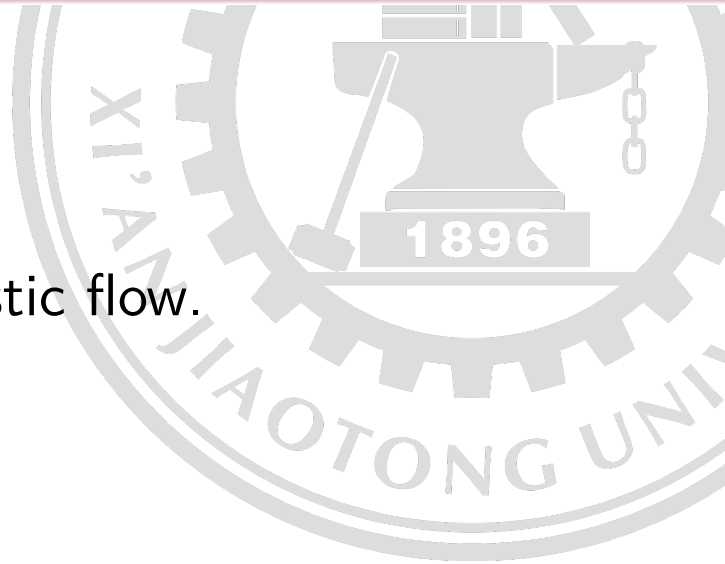
Qeustions

- Question 1: What kind of information?
- Elasticity



Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.



Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.

Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.

Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.
- If cross traffic is inelastic then,

Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.
- If cross traffic is inelastic then,
  - Use delay-controlling mode, like Vegas, Copa, etc.

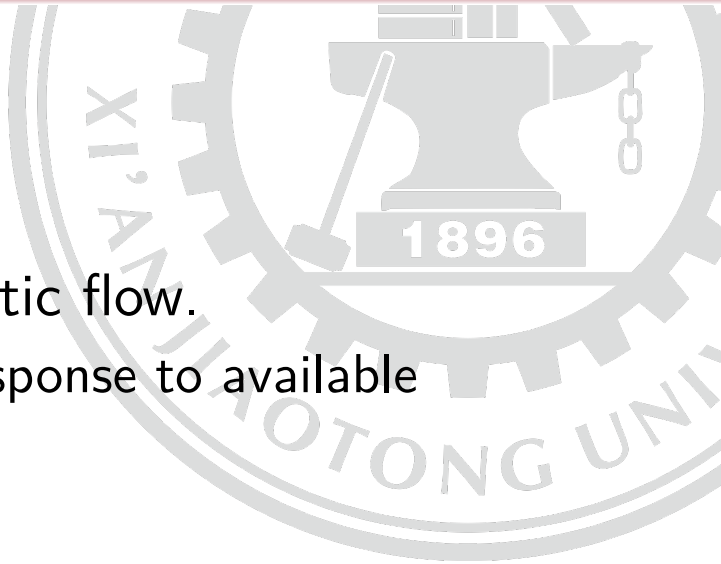
Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.
- If cross traffic is inelastic then,
  - Use delay-controlling mode, like Vegas, Copa, etc.
- Otherwise,

Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.
- If cross traffic is inelastic then,
  - Use delay-controlling mode, like Vegas, Copa, etc.
- Otherwise,
  - Use TCP-competitive mode, like Cubic, BBR, etc.

Qeustions



- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.
- If cross traffic is inelastic then,
  - Use delay-controlling mode, like Vegas, Copa, etc.
- Otherwise,
  - Use TCP-competitive mode, like Cubic, BBR, etc.
- Switch between two modes to achieve

Qeustions

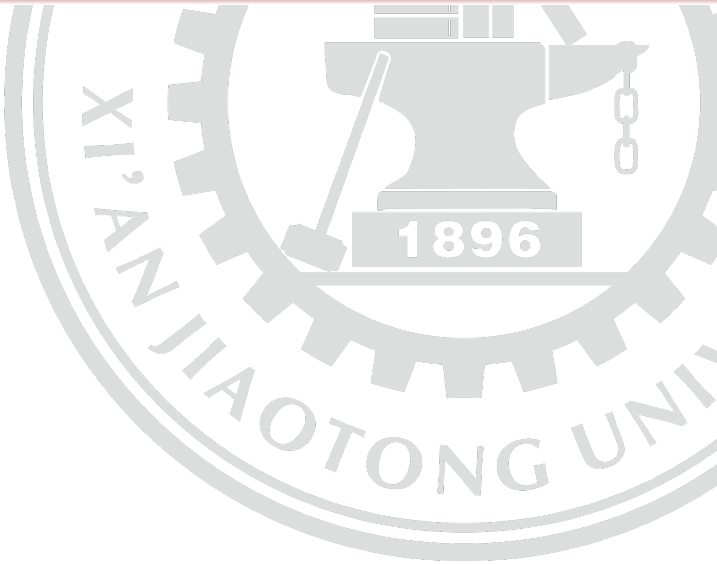
- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.
- If cross traffic is inelastic then,
  - Use delay-controlling mode, like Vegas, Copa, etc.
- Otherwise,
  - Use TCP-competitive mode, like Cubic, BBR, etc.
- Switch between two modes to achieve
  - Low delays

Qeustions

- Question 1: What kind of information?
- Elasticity
- Elastic cross traffic: If contains *any* elastic flow.
  - Elastic flow: Adjust sending rate in response to available bandwidth.
  - Inelastic flow: Otherwise.
- If cross traffic is inelastic then,
  - Use delay-controlling mode, like Vegas, Copa, etc.
- Otherwise,
  - Use TCP-competitive mode, like Cubic, BBR, etc.
- Switch between two modes to achieve
  - Low delays
  - Same throughtput as the deployed TCP-competitive CC algorithms

Qeustions

- Question 2: How to detect elasticity?



Qeustions

- Question 2: How to detect elasticity?
- Strawman: Compare total queueing delay and "self-inflicted" delay.

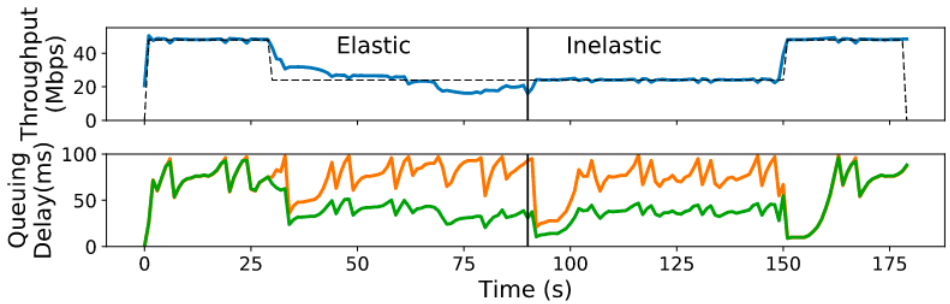


Figure 1: Instantaneous delay measurements do not reveal elasticity.[1]

[1] From Figure 2 in [4].

Qeustions

- Question 2: How to detect elasticity?
- Strawman: Compare total queueing delay and "self-inflicted" delay.

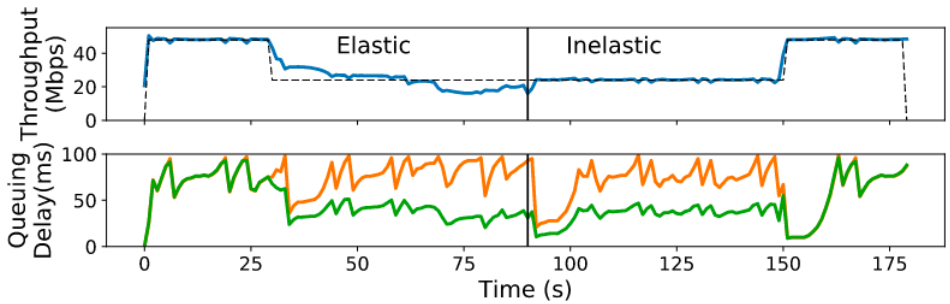


Figure 1: Instantaneous delay measurements do not reveal elasticity.[1]

- Nimbus

[1] From Figure 2 in [4].

Qeustions

- Question 2: How to detect elasticity?
- Strawman: Compare total queueing delay and "self-inflicted" delay.

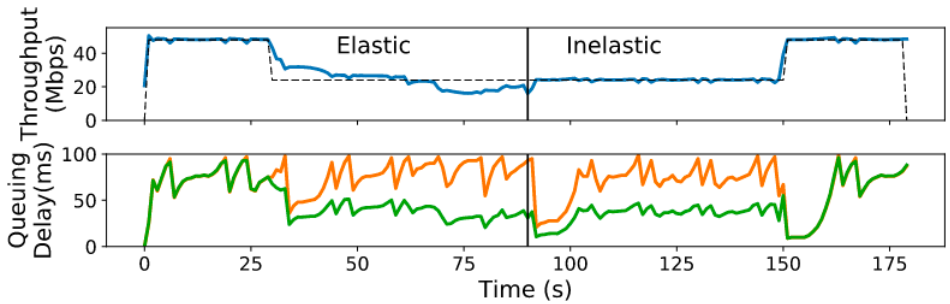


Figure 1: Instantaneous delay measurements do not reveal elasticity.[1]

- Nimbus
  - Modulate sending rate to create variations in available bandwidth.

[1] From Figure 2 in [4].

Qeustions

- Question 2: How to detect elasticity?
- Strawman: Compare total queueing delay and "self-inflicted" delay.

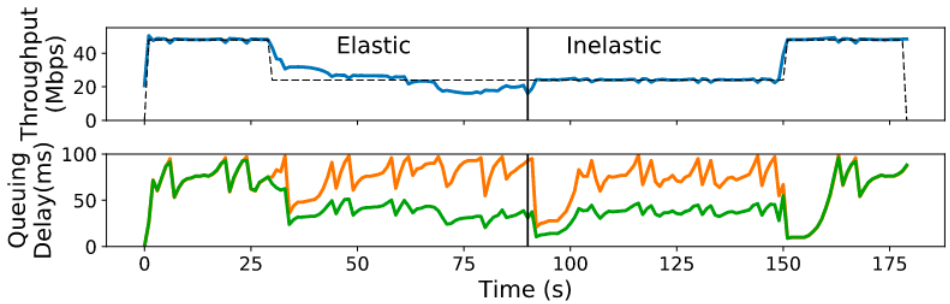


Figure 1: Instantaneous delay measurements do not reveal elasticity.[1]

- Nimbus
  - Modulate sending rate to create variations in available bandwidth.
  - Monitor how the cross traffic responds to these variations.

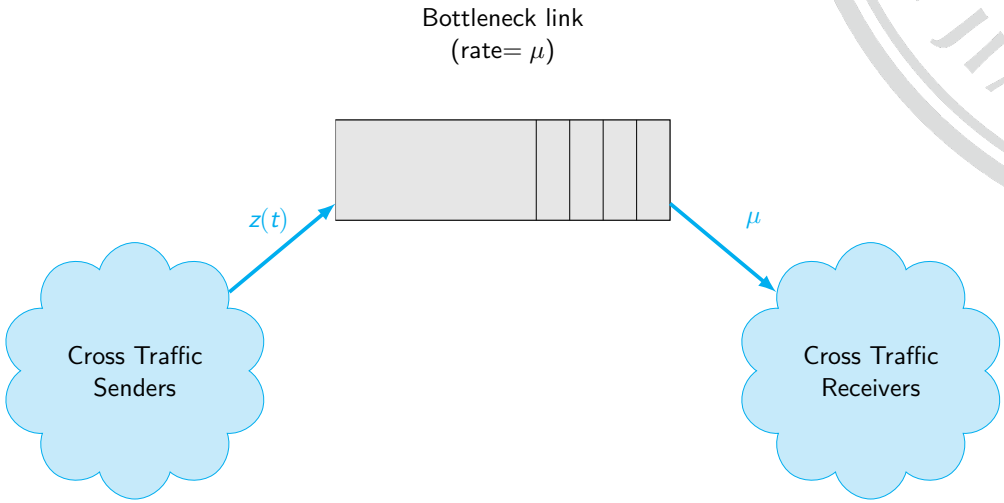
[1] From Figure 2 in [4].

Steps

- Step 1: Measure rate of the cross traffic.

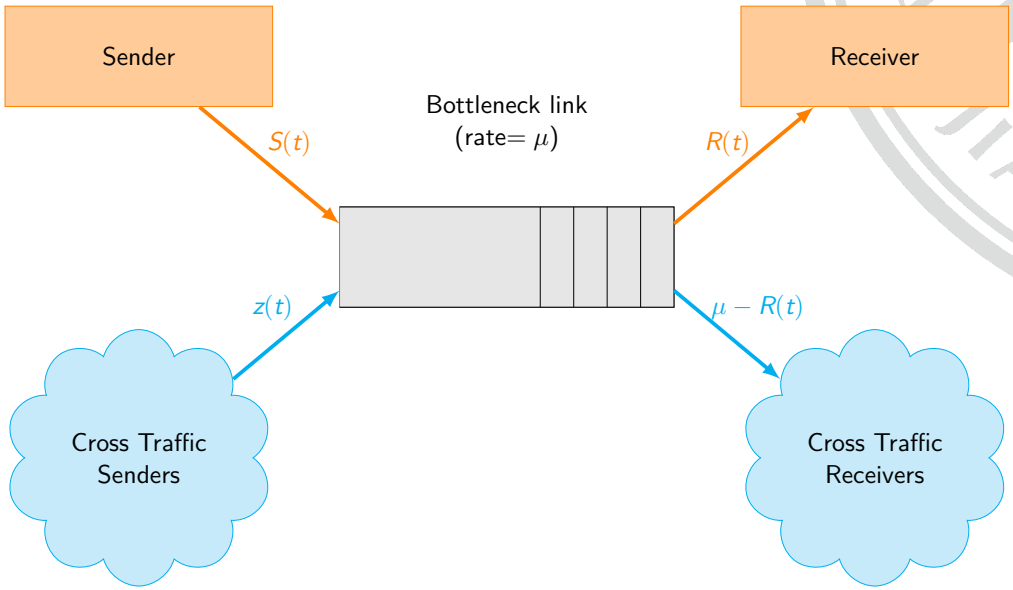
Steps

- Step 1: Measure rate of the cross traffic.



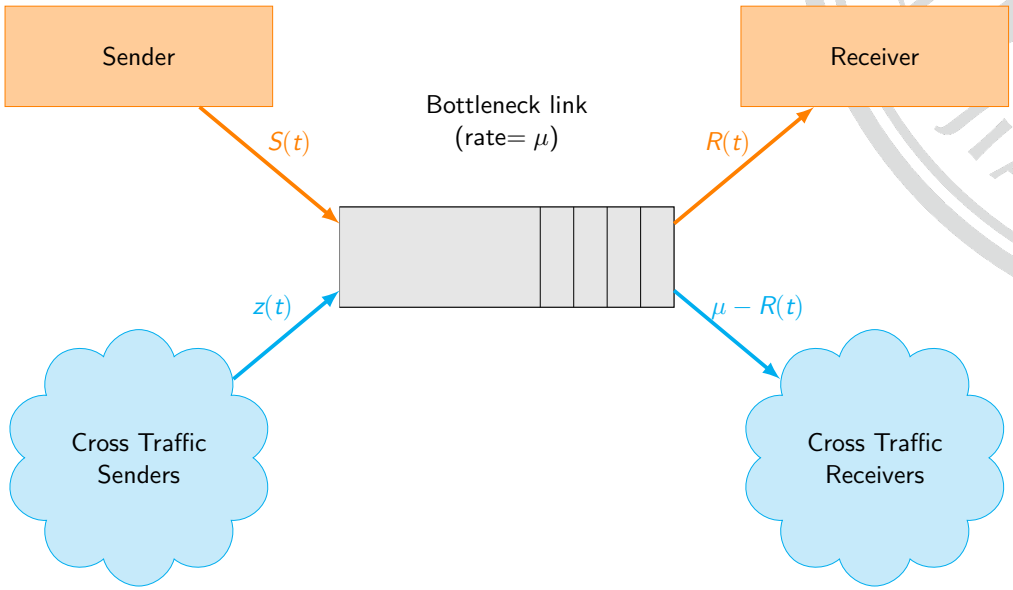
Steps

- Step 1: Measure rate of the cross traffic.



Steps

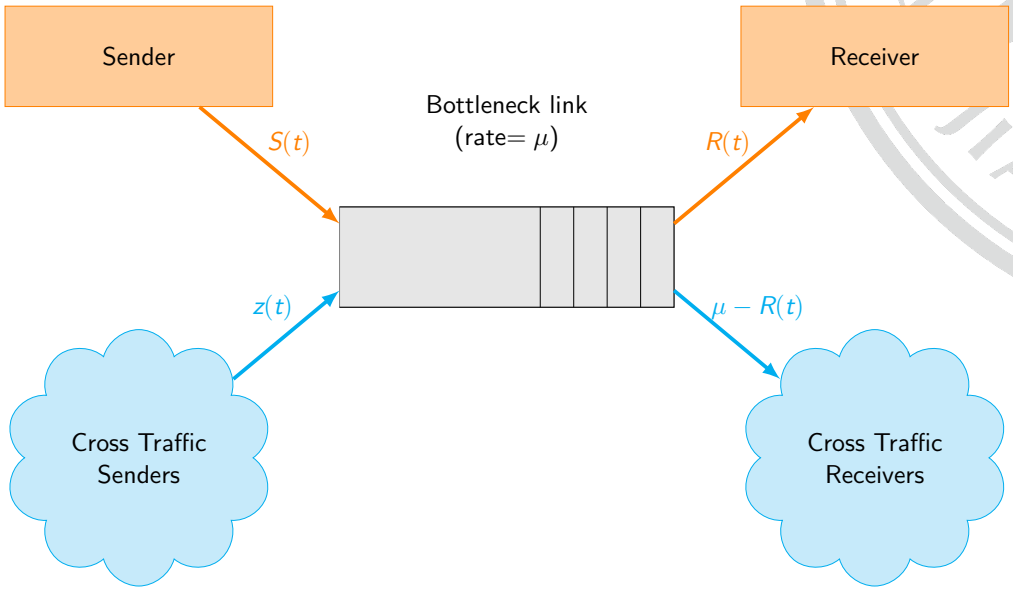
- Step 1: Measure rate of the cross traffic.



- Input: $\frac{S(t)}{S(t)+z(t)}$,

Steps

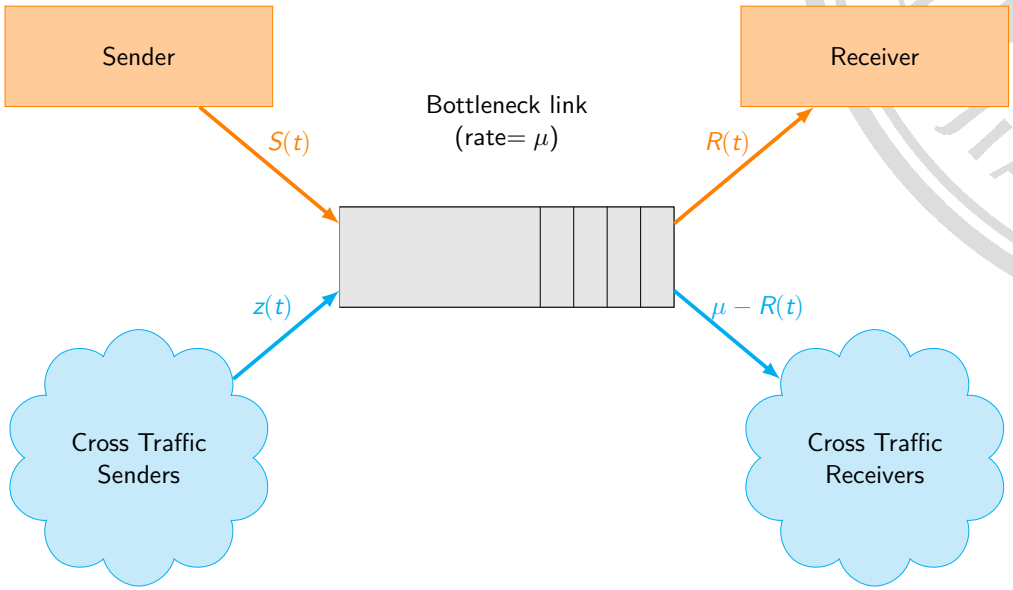
- Step 1: Measure rate of the cross traffic.



- Input: $\frac{S(t)}{S(t)+z(t)}$, Output: $\frac{R(t)}{\mu}$,

Steps

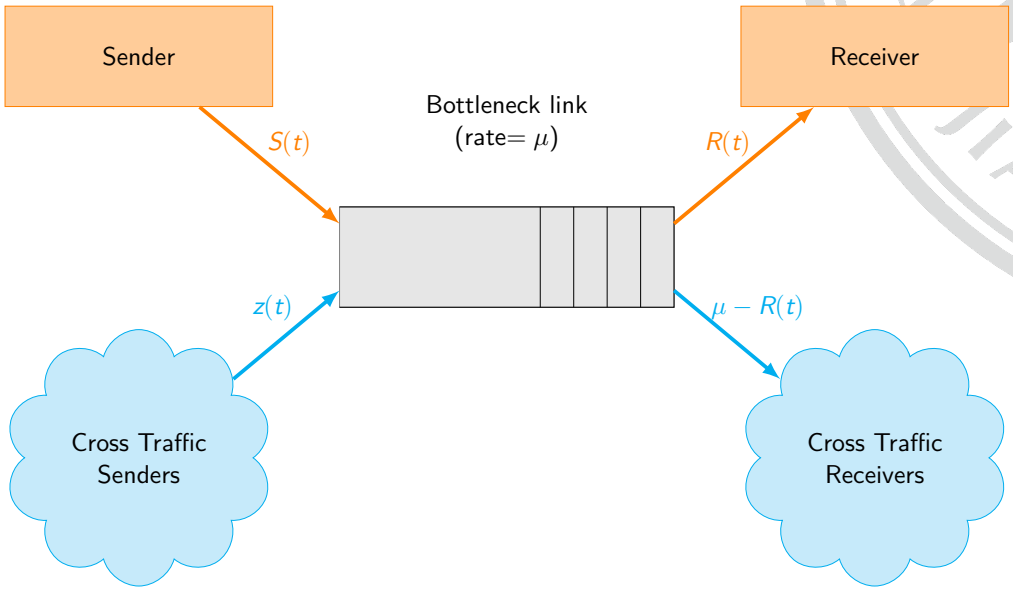
- Step 1: Measure rate of the cross traffic.



- Input: $\frac{S(t)}{S(t)+z(t)}$, Output: $\frac{R(t)}{\mu}$, $\Rightarrow \frac{S(t)}{S(t)+z(t)} = \frac{R(t)}{\mu}$

Steps

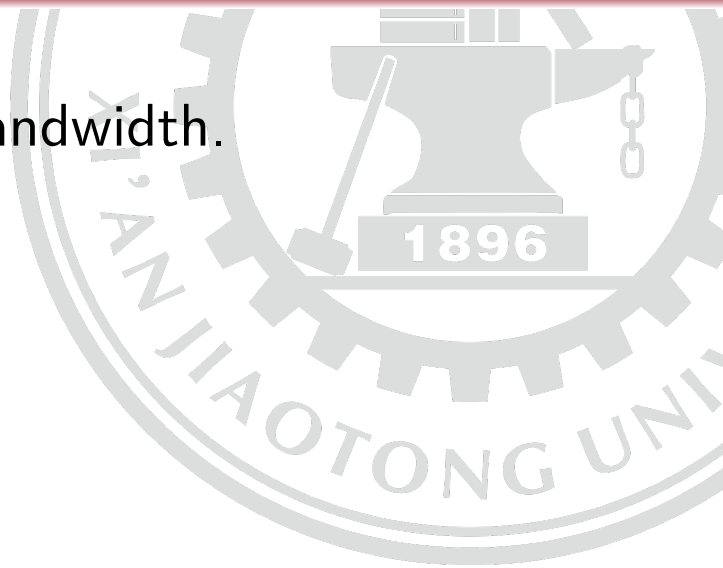
- Step 1: Measure rate of the cross traffic.



- Input: $\frac{S(t)}{S(t)+z(t)}$, Output: $\frac{R(t)}{\mu}$, $\Rightarrow \frac{S(t)}{S(t)+z(t)} = \frac{R(t)}{\mu}$
- $\Rightarrow \hat{z}(t) = \mu\frac{S(t)}{R(t)} - S(t)$

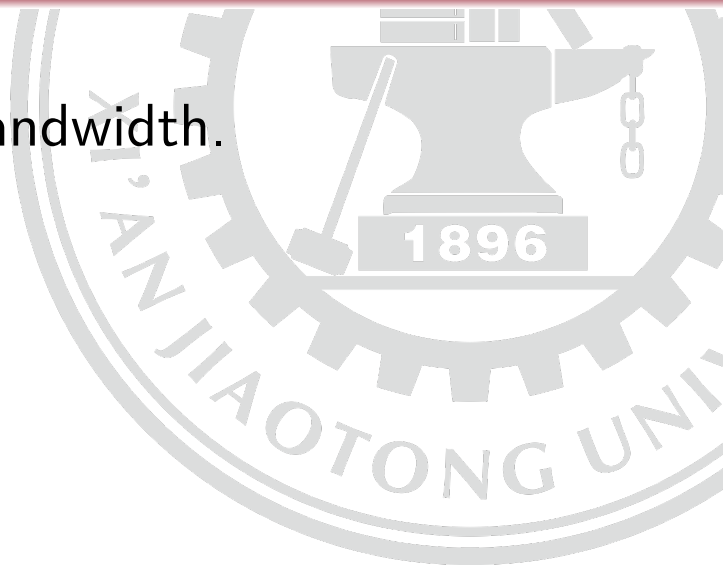
Steps

- Step 2: Create variations in available bandwidth.



Steps

- Step 2: Create variations in available bandwidth.
- Send packets in pulses.



Steps

- Step 2: Create variations in available bandwidth.
- Send packets in pulses.
- $S(t) = S_{expected}(t) + A\sin(2\pi ft)$.

Steps

- Step 2: Create variations in available bandwidth.
- Send packets in pulses.
- $S(t) = S_{expected}(t) + A\sin(2\pi ft)$.

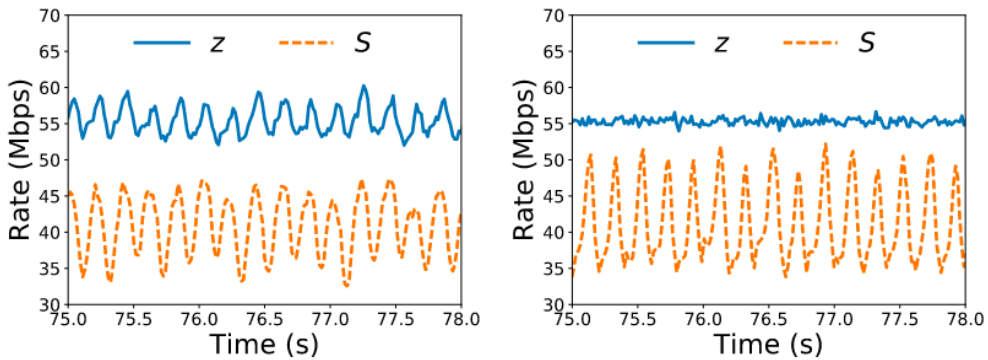
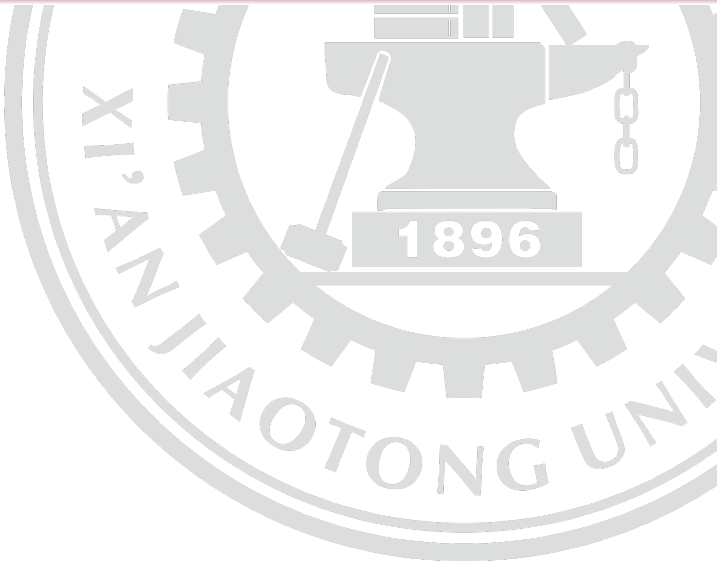


Figure 2: Cross traffic's reaction to pulses.[1]

[1] From Figure 3 in [4].

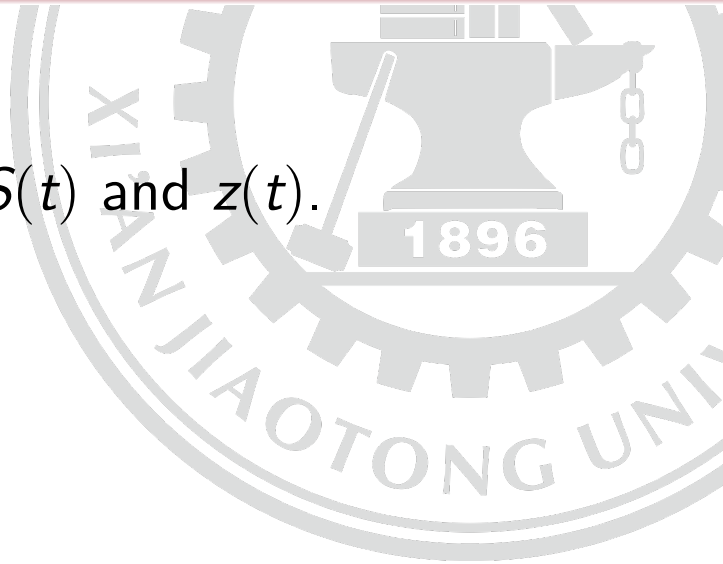
Steps

- Step 3: Classify cross traffic elasticity.



Steps

- Step 3: Classify cross traffic elasticity.
- Measure the *cross-correlation* between $S(t)$ and $z(t)$.

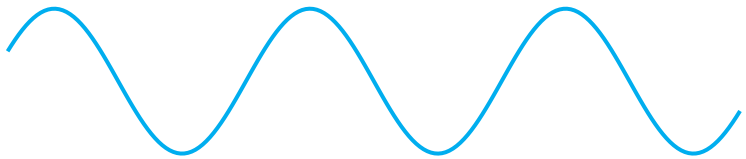


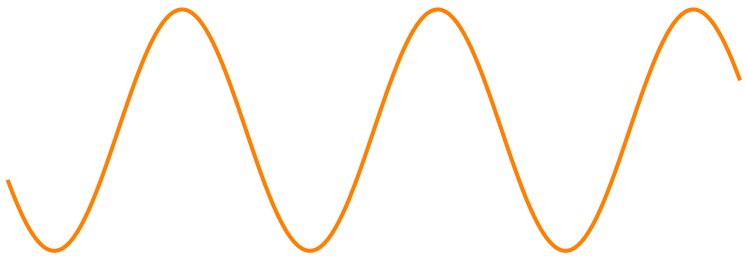
Steps

- Step 3: Classify cross traffic elasticity.
- Measure the *cross-correlation* between $S(t)$ and $z(t)$. (×)
- Align $S(t)$ and $z(t)$ is challenging.

Steps

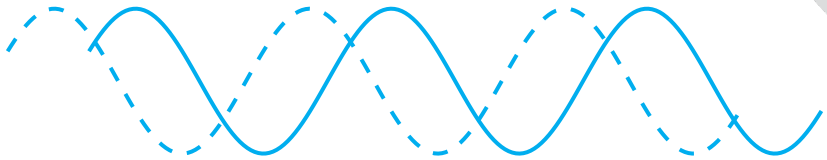
- Step 3: Classify cross traffic elasticity.
- Measure the *cross-correlation* between $S(t)$ and $z(t)$. ($\times$)
- Align $S(t)$ and $z(t)$ is challenging.

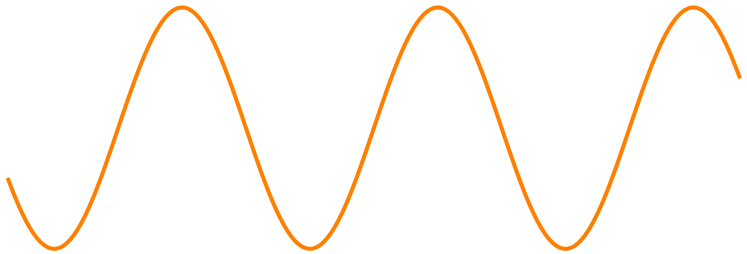




Steps

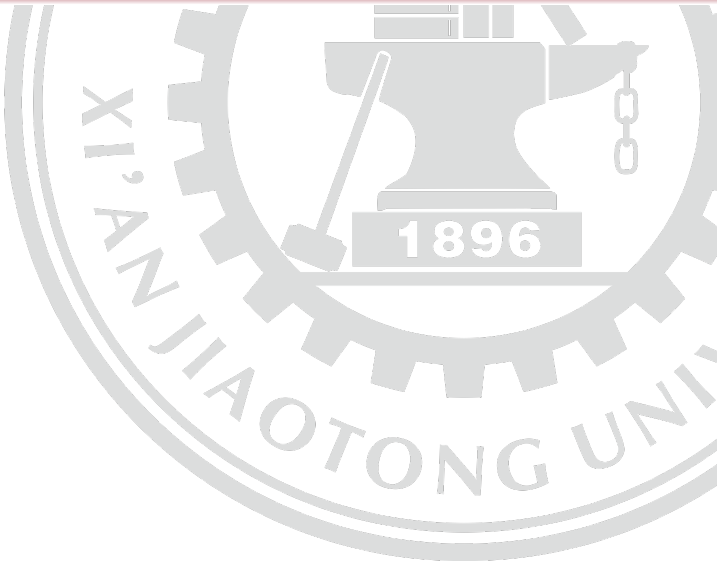
- Step 3: Classify cross traffic elasticity.
- Measure the *cross-correlation* between $S(t)$ and $z(t)$. ($\times$)
- Align $S(t)$ and $z(t)$ is challenging.





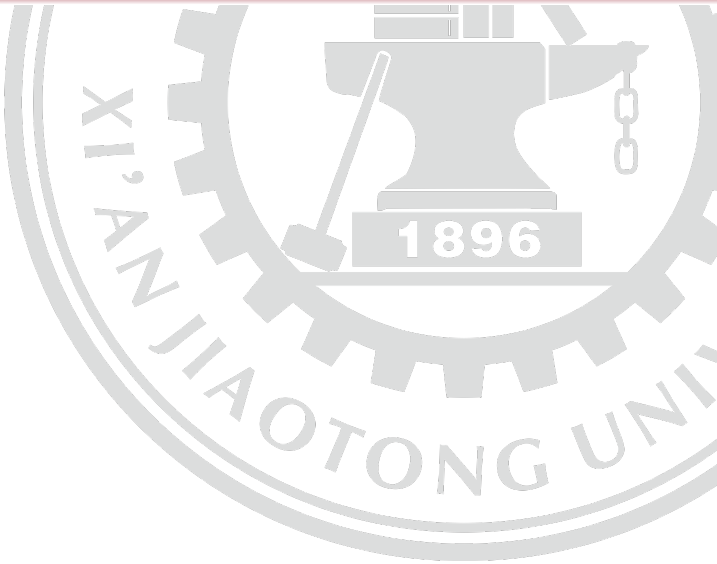
Steps

- Step 3: Classify cross traffic elasticity.



Steps

- Step 3: Classify cross traffic elasticity.
- From time to frequency domain



Steps

- Step 3: Classify cross traffic elasticity.
- From time to frequency domain

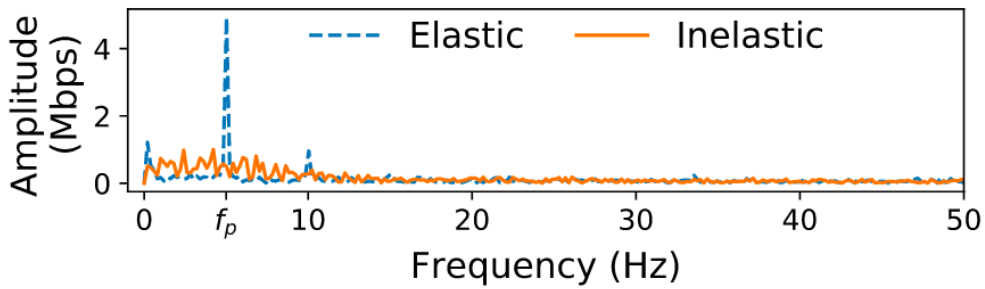


Figure 3: Cross traffic FFT for elastic and inelastic traffic.[1]

[1] From Figure 4 in [4].

Steps

- Step 3: Classify cross traffic elasticity.
- From time to frequency domain

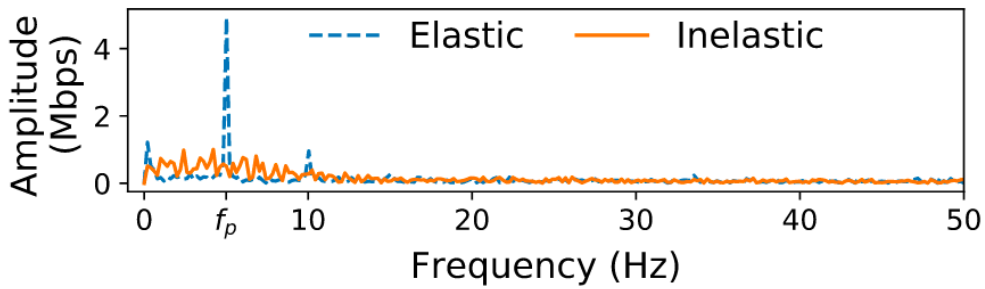


Figure 3: Cross traffic FFT for elastic and inelastic traffic.[1]

- Elastic cross traffic exhibits a pronounced peak at f_p compared to the neighboring frequencies.

[1] From Figure 4 in [4].

Steps

- Step 3: Classify cross traffic elasticity.
- From time to frequency domain

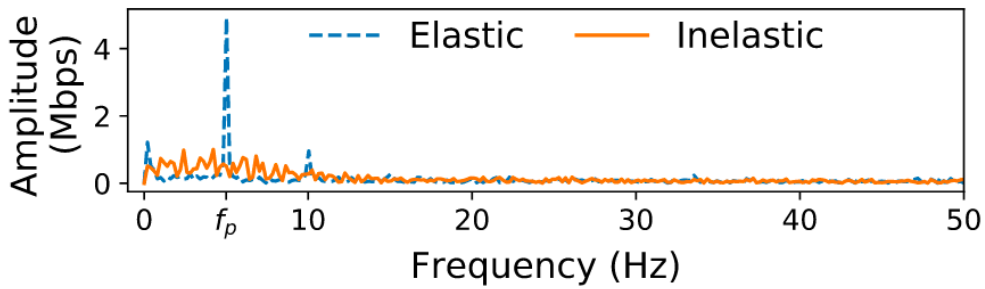
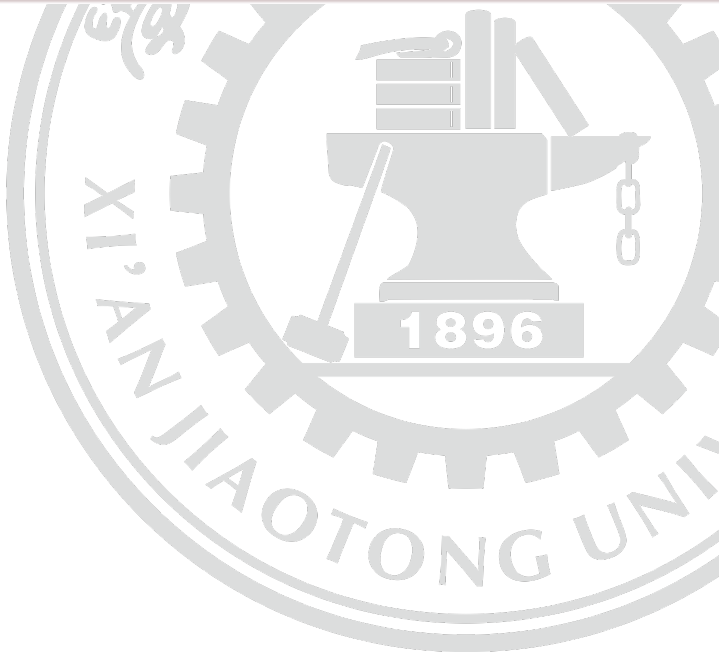


Figure 3: Cross traffic FFT for elastic and inelastic traffic.[1]

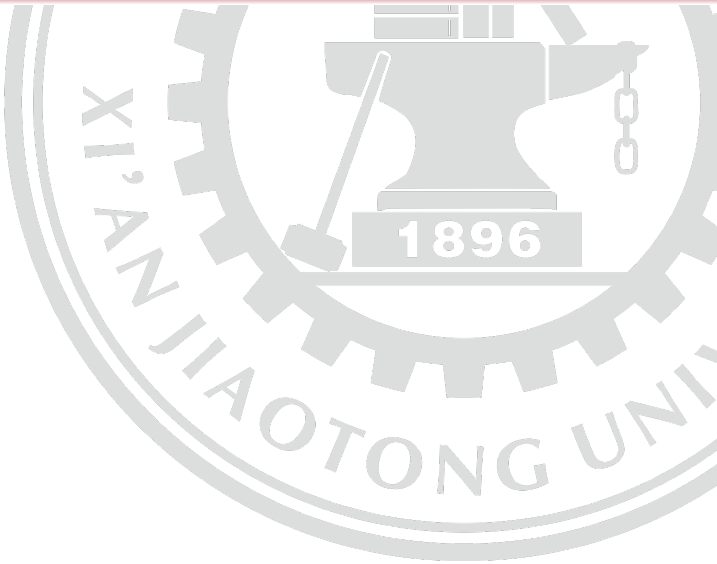
- Elastic cross traffic exhibits a pronounced peak at f_p compared to the neighboring frequencies.
- No need to know RTT of cross traffic.

[1] From Figure 4 in [4].



Experiment

- Setup
  - Bottleneck rate: 96 Mbit/s,
  - Minimum RTT: 50 ms
  - Buffer: 2 BDP



Experiment

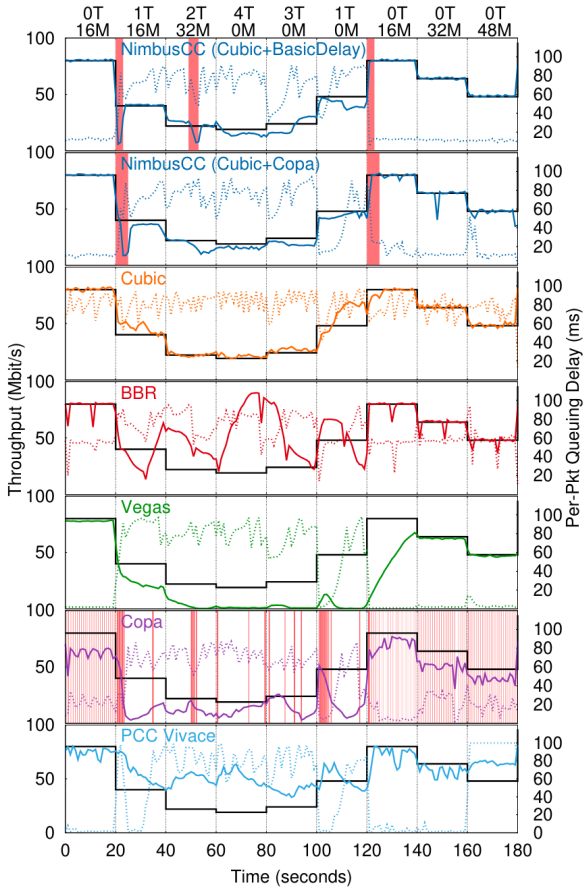
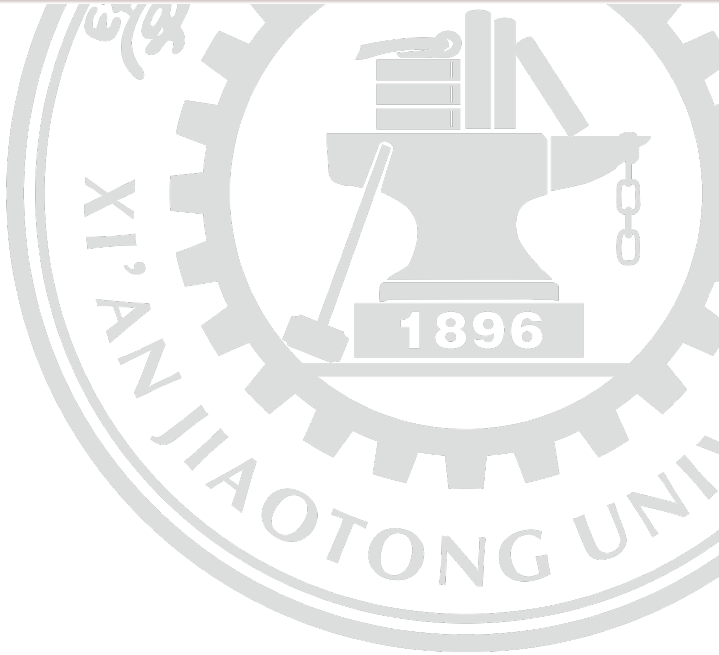


Figure 4: Performance of NimbusCC.[1]

[1] From Figure 7 in [4].



Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.
- The pulser: Use two different frequencies,

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.
- The pulser: Use two different frequencies,
  - f_{pc} in TCP-competitive mode, and

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.
- The pulser: Use two different frequencies,
  - f_{pc} in TCP-competitive mode, and
  - f_{pd} in delay-control mode.

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.
- The pulser: Use two different frequencies,
  - f_{pc} in TCP-competitive mode, and
  - f_{pd} in delay-control mode.
- A watcher: Infers whether the pulser is pulsing at frequency f_{pc} or frequency f_{pd} by

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.
- The pulser: Use two different frequencies,
  - f_{pc} in TCP-competitive mode, and
  - f_{pd} in delay-control mode.
- A watcher: Infers whether the pulser is pulsing at frequency f_{pc} or frequency f_{pd} by
  - computing the FFT of its receive rate, R, at these two frequencies.

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.
- The pulser: Use two different frequencies,
  - f_{pc} in TCP-competitive mode, and
  - f_{pd} in delay-control mode.
- A watcher: Infers whether the pulser is pulsing at frequency f_{pc} or frequency f_{pd} by
  - computing the FFT of its receive rate, R, at these two frequencies.
- Pulser election: A distributed and randomized election decides which flow is the pulser and which are watchers.

Multiple NimbusCC Flows

- NimbusCC fails if all the NimbusCC flows pulse at the same frequency(f_p).
- Set the pulser and the watchers.
- The pulser: Use two different frequencies,
  - f_{pc} in TCP-competitive mode, and
  - f_{pd} in delay-control mode.
- A watcher: Infers whether the pulser is pulsing at frequency f_{pc} or frequency f_{pd} by
  - computing the FFT of its receive rate, R, at these two frequencies.
- Pulser election: A distributed and randomized election decides which flow is the pulser and which are watchers.
  - Similar to CSMA

[1] Venkat Arun and Hari Balakrishnan.
Copa: Practical {Delay-Based} congestion control for the internet.
In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 329–342, 2018.

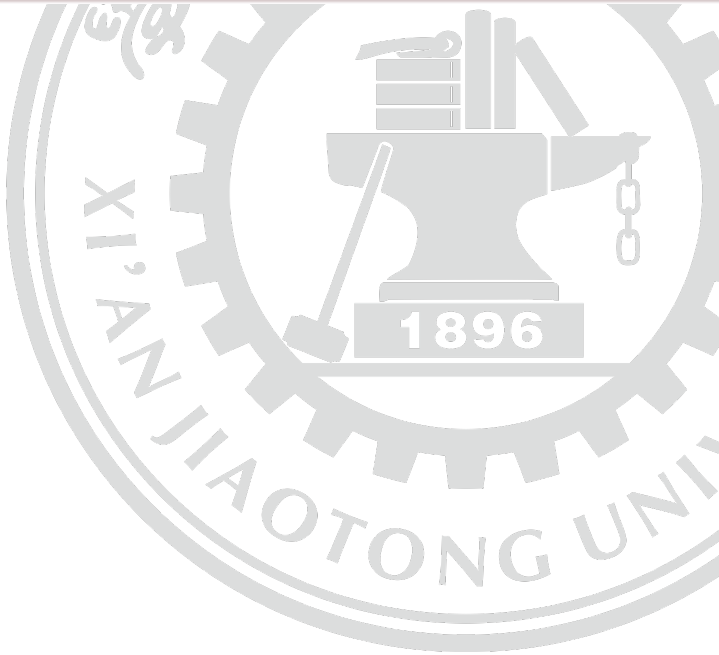
[2] Lawrence S Brakmo, Sean W O'malley, and Larry L Peterson.
Tcp vegas: New techniques for congestion detection and avoidance.
In *Proceedings of the conference on Communications architectures, protocols and applications*, pages 24–35, 1994.

[3] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson.
Bbr: Congestion-based congestion control.
Communications of the ACM, 60(2):58–66, 2017.

[4] Prateesh Goyal, Akshay Narayan, Frank Cangialosi, Srinivas Narayana, Mohammad Alizadeh, and Hari Balakrishnan.
Elasticity detection: A building block for internet congestion control.
In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 158–176, 2022.

[5] Sangtae Ha, Injong Rhee, and Lisong Xu.
Cubic: a new tcp-friendly high-speed tcp variant.
ACM SIGOPS operating systems review, 42(5):64–74, 2008.

[6] Janey C Hoe.
Improving the start-up behavior of a congestion control scheme for tcp.
ACM SIGCOMM Computer Communication Review, 26(4):270–280, 1996.



Thanks!