Adaptive Congestion Control for Unpredictable Cellular Networks

Yasir Zaki

New York University Abu Dhabi (NYUAD)

Verus design goals

1. Track fast channel changes
2. Balance throughput and delay
3. Provide fairness between competing flows

- Verus uses delay feedback
  - Changes only done at the end nodes
  - Proactively avoid congestion
  - Small signaling overhead

Verus design

- No channel prediction/modeling

- Build on TCP concepts:
  - Use slow start
  - Use Multiplicative Decrease (MD) on packet loss
  - Replace Additive Increase (AI) with a step based increase/decrease

Verus in a nutshell

- Learns the **delay profile** of the network
  - Reflects the relationship between delay and sending window
  - Represented as a curve and re-built every **1 second**

- Decide how many packets to send over **5 ms epochs**

- Enforces a delay estimate based on the **delay profile**
  - With a step-based increase/decrease

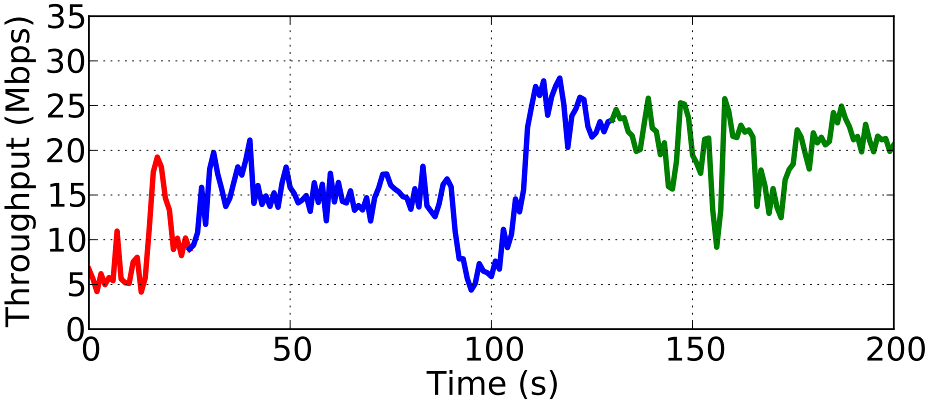
$$W(t+1) = \boxed{f(\underbrace{d(t) \pm \delta(t)}_{\text{Estimated delay } (D_{est})})}\ \text{Delay curve}$$

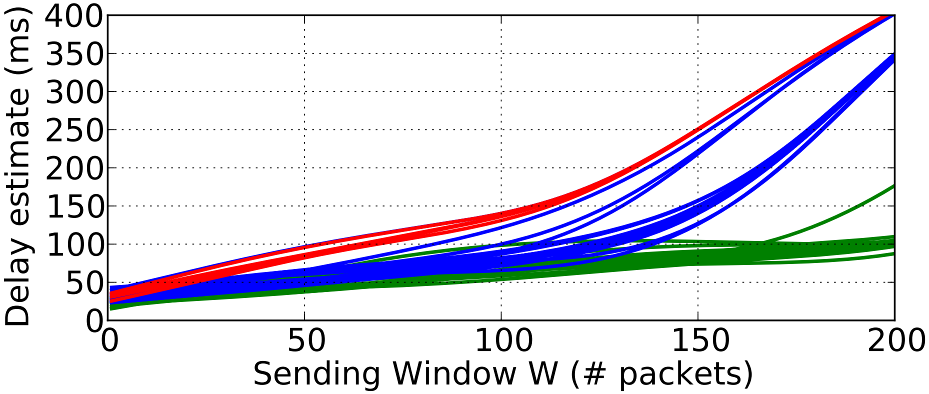
Delay curve concept

- A way to track network changes

- Reflects relationship between sending window and network delay

- Verus dynamically learns the network state
  - Through delay feedback (ACKs)





Tracking fast channel changes

Slow start:

- Every ACK: add a point (W, delay)

Build delay curve:

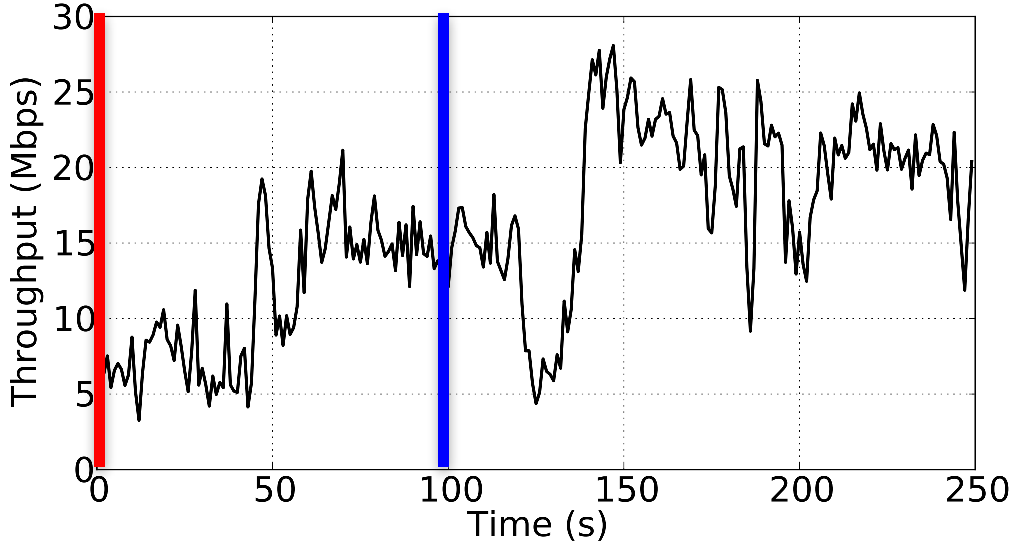
- Cubic spline interpolation

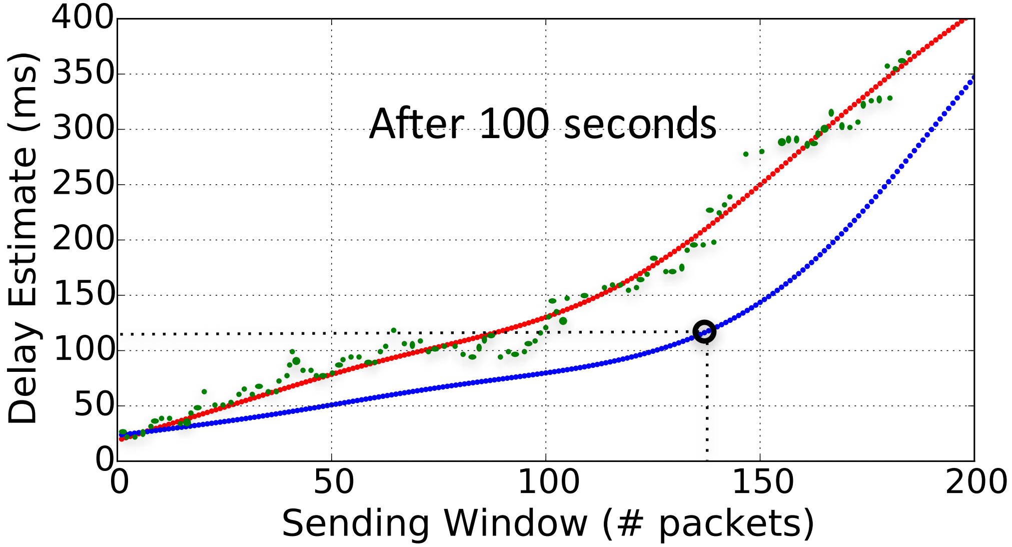
Verus control loop:

- every epoch 5 ms

Rebuild delay curve:

- every 1 second



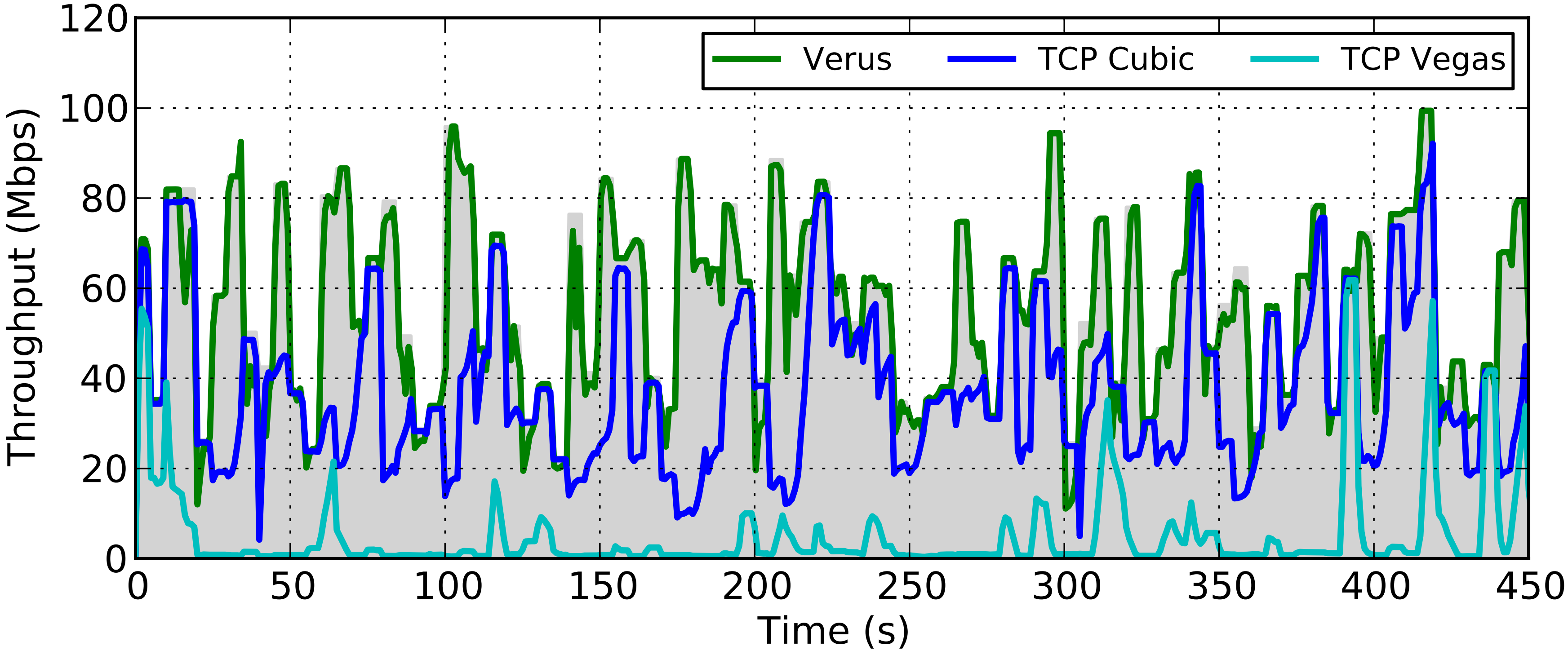


Tracking fast channel changes

Every 5 sec:

Link: 10-100 Mbps

Round trip time: 10-100 ms

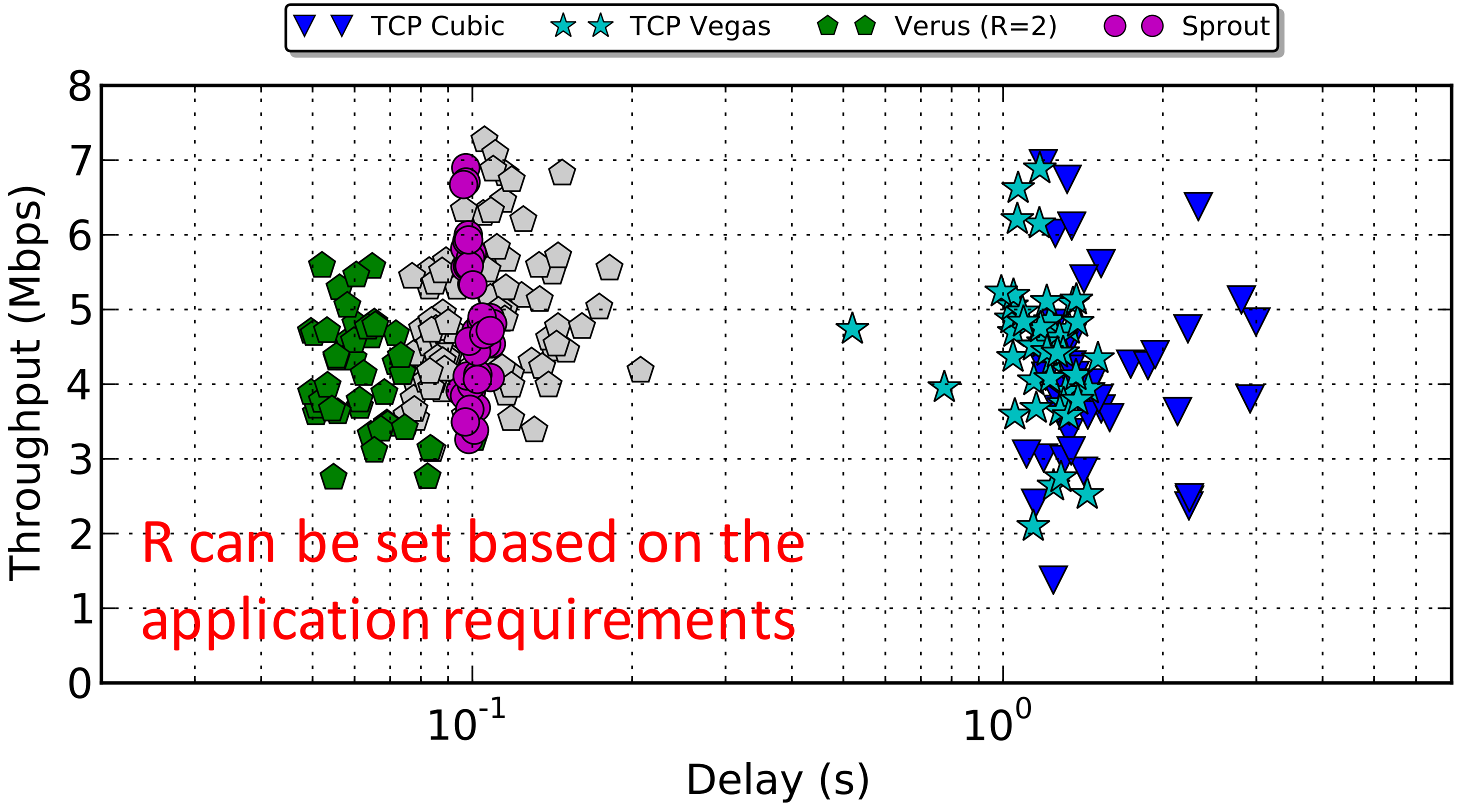


Trade-off between throughput & delay

Tuning parameter (R) defines the ratio between max and min network delay

Experiments over real LTE network:

- Stationary scenario
- 3 phones each running 3 flows
- Repeated 5 times each

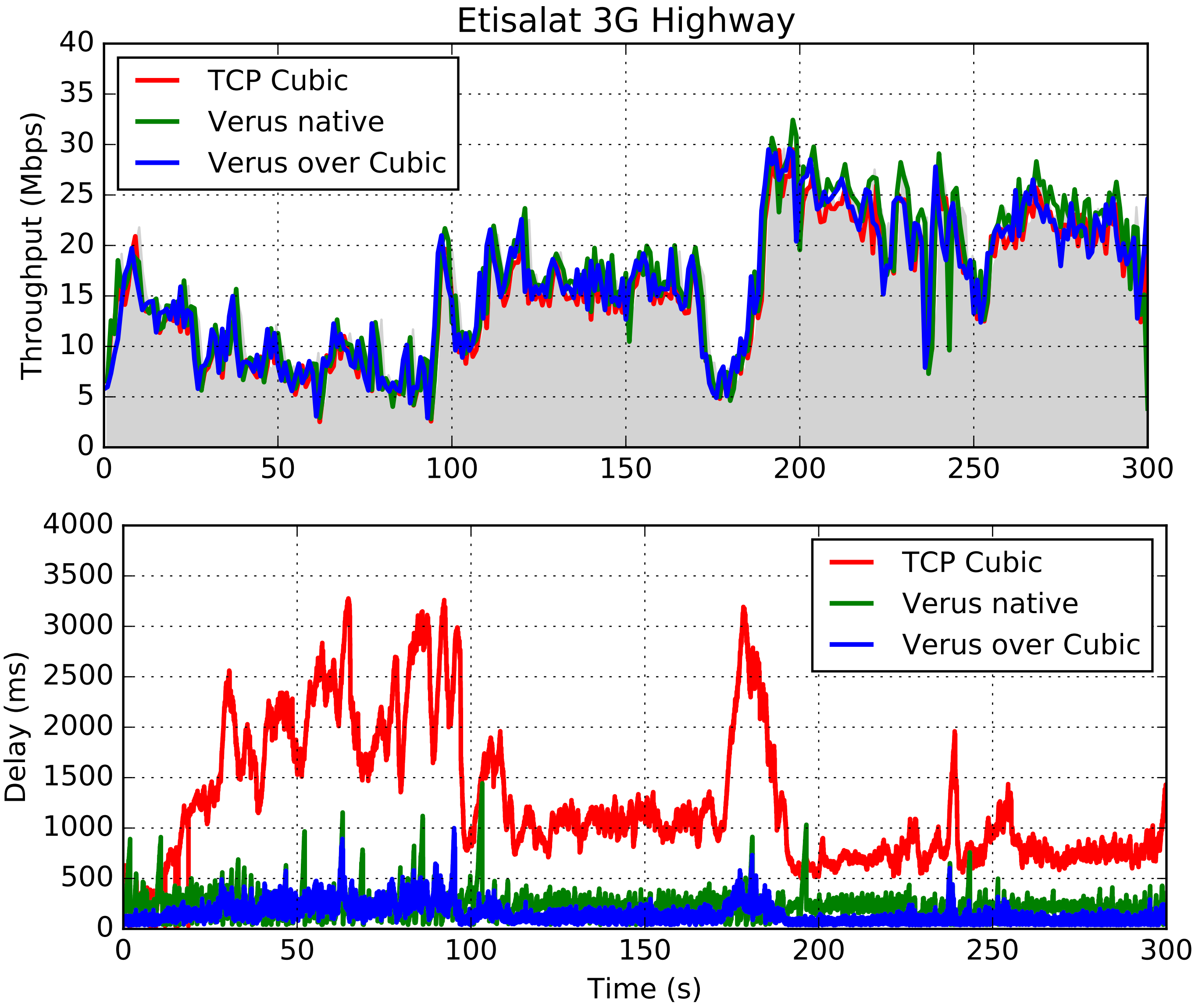


R can be set based on the application requirements

Verus implementations

- Native Verus over UDP
  - http://yzaki.github.io/verus/
- Verus Sockets (ongoing)
  - As CC module in UDT (UDP-based Data Transfer) http://udt.sourceforge.net/index.html
- As an adaptation layer over TCP Cubic
  - https://github.com/yzaki/verus/tree/verus_over_tcp
- As a CC module within Quic (Planned)

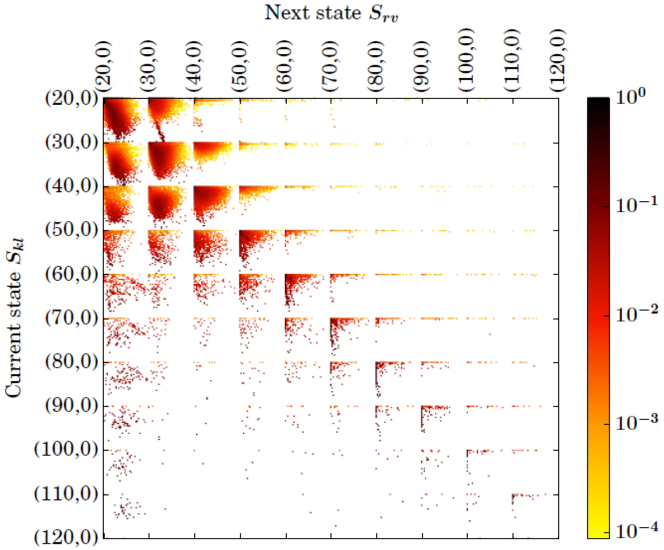
Verus over TCP Cubic



Verus modeling

- Delay based CC protocols are not well understood
  - A generic mathematical description
    - Simplify the understanding of these protocols
    - Prove convergence and stability

- Verus is modeled as a two-dimensional discrete-time Markov chain
  - Focus on highly fluctuating networks
  - Reflect properties of the protocol
  - Achieve similar performance


Next state S_{rv}
(20,0)
(30,0)
(40,0)
(50,0)
(60,0)
(70,0)
(80,0)
(90,0)
(100,0)
(110,0)
(120,0)
Current state S_{kl}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}