Understanding Bufferbloat in Cellular Networks

Haiqing Jiang, Zeyu Liu, Yaogong Wang, Kyunghan Lee, and Injong Rhee

Published in 2012 in the ACM SIGCOMM CellNet workshop on cellular networks

Presented by Vasilios Mitrokostas [my side comments in blue]
Graph images taken from paper

- **Introduction to bufferbloat**
- Authors' observations on bufferbloat in cellular networks
- Bufferbloat analysis
- Analysis of the involvement of TCP
- Existing and suggested solutions
- Some quick thoughts on this paper

Why study bufferbloat?

- In measuring TCP across four major US cellular networks, authors found performance degradation issues:
  - Increased delay
  - Low throughput
- One proposed major cause: bufferbloat
- The claim: these major carriers are "over-buffered"

Bufferbloat

- An issue where the buffering of packets actually increases delay, increases jitter, and decreases throughput
- The original intention of increased buffer size was to improve Internet performance
- If the size is too large, the interaction between the buffer and TCP congestion control degrades overall network performance

How bufferbloat causes issues

- Large packet buffers cause loss-based TCP congestion control algorithms to overestimate packets to queue
  - Leads to longer queuing delays
  - Results in packet delay variation (jitter)
- Essentially, packets are buffered when they instead should be dropped
- If this occurs on a bottlenecked link with a large packet buffer (e.g., on a newer router), packets will not be dropped until the buffer is full, causing TCP congestion avoidance to react slowly

Why would buffers be large?

- Large packet buffers help . . .
  - . . . deal with bursty traffic
  - . . . support user fairness
  - . . . promote channel variability
- Not as simple as merely reducing buffer sizes

- Introduction to bufferbloat
- **Authors' observations on bufferbloat in cellular networks**
- Bufferbloat analysis
- Analysis of the involvement of TCP
- Existing and suggested solutions
- Some quick thoughts on this paper

The authors' “untold story”

- Large buffers are causing issues
- Making them small isn't an elegant solution
- A trick employed by smartphone vendors today: set maximum TCP receive buffer size to a small value
  - Advertised window can't exceed this value
  - Sending window is the lesser of the congestion window and advertised window
  - As a result, this limitation keeps buffers from overfilling and mitigates end-to-end delay
- The problem: what's the right value?

The paper's goals

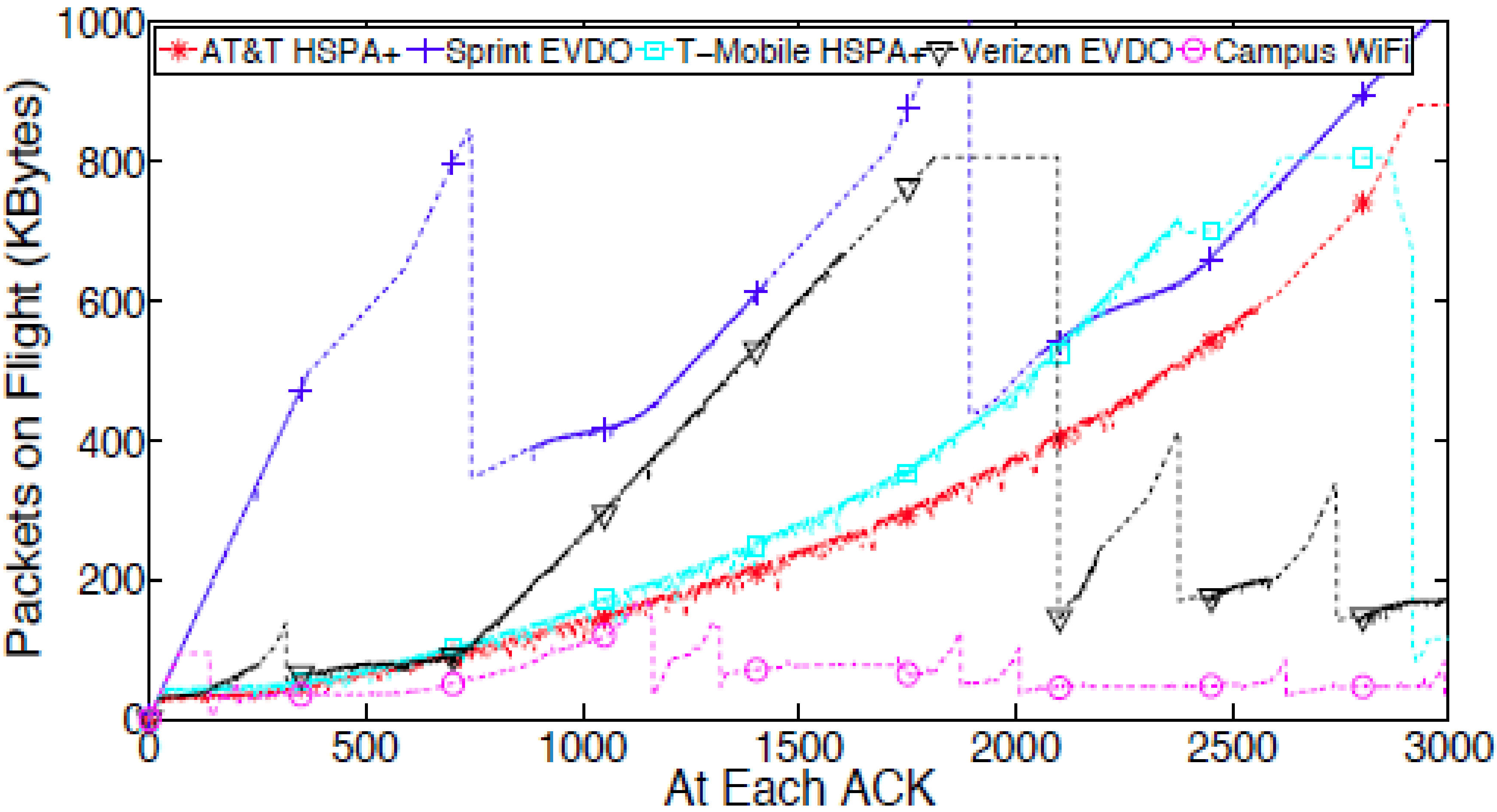
- Establish the prevalence of the bufferbloat problem in cellular networks
- Show that high-speed TCP aggravates the performance degradation of bufferbloated networks
- Discuss practical solutions

- Introduction to bufferbloat
- Authors' observations on bufferbloat in cellular networks
- **Bufferbloat analysis**
- Analysis of the involvement of TCP
- Existing and suggested solutions
- Some quick thoughts on this paper

Setting up the test

- Bulk-data transfer between laptop (receiver) and server (sender) over 3G networks; laptop access 3G mobile data across multiple US carriers
- Both sender and receiver use TCP CUBIC and Linux (Ubuntu 10.04)
  - Ubuntu, by default, sets maximum receive buffer size and maximum send buffer size to a large value
  - This way, flow is not limited by buffer size
- Detailed queue size is unknown, so the first test (the following chart) attempts to estimate this

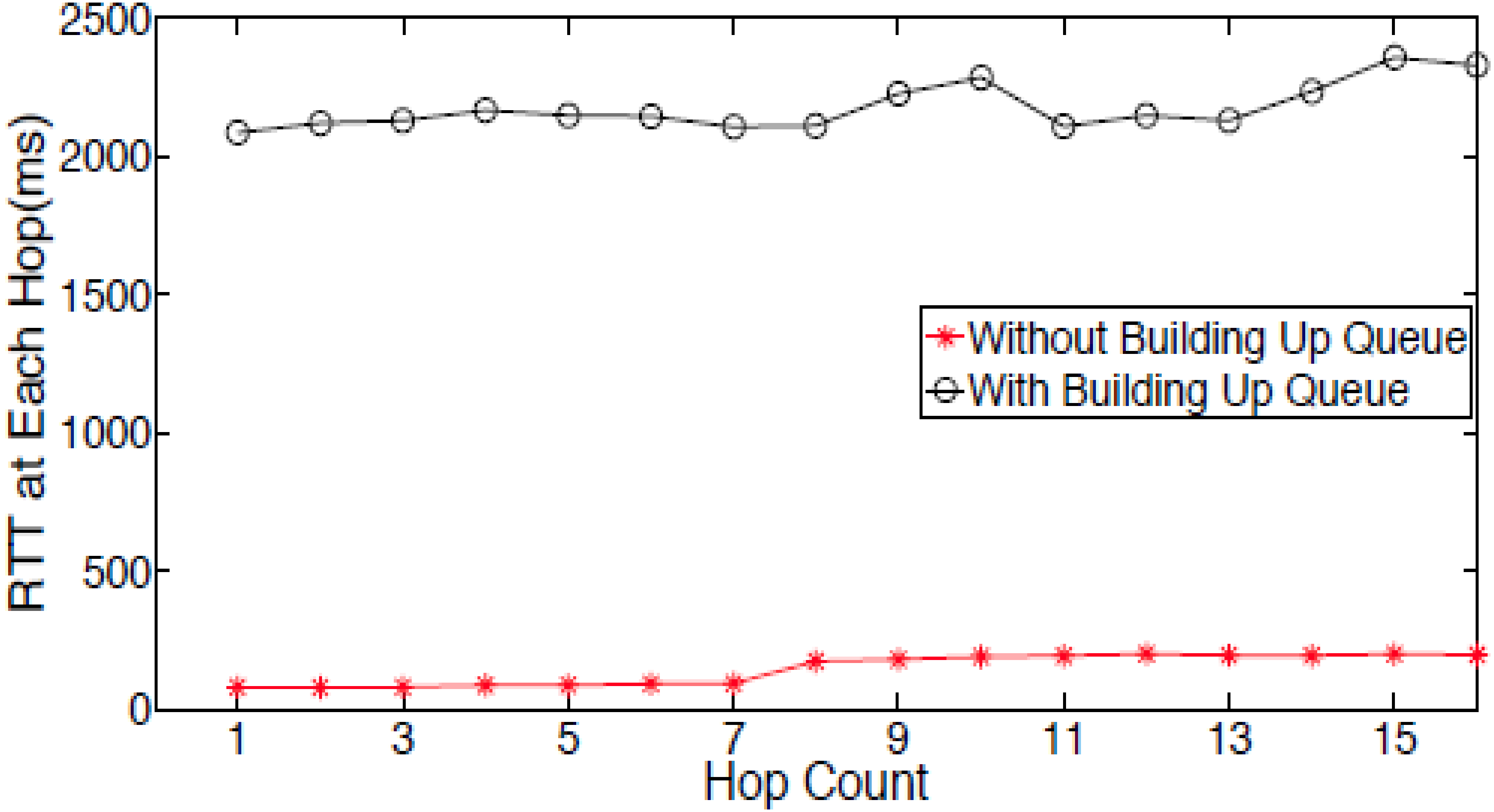
Estimating network buffer space



Estimating network buffer space

- Campus WiFi: baseline choice
- Despite long link distance and high bandwidth, WiFi experiment yields smaller results than cellular networks
- The cellular networks use buffer sizes beyond reasonable ranges; for example, Sprint supports over 1000 KB of in-flight packets, but its EVDO network does not support it [source?]
- How do we know this bufferbloat is occurring within the cellular network?

Queue build-up experiment



Queue build-up experiment

- Authors' observation: queuing delay begins at the very first IP hop which contains the cellular link
- What about other hops? Authors suggest packets are buffered on the way back as well due to the long queue already built-up

Simulating 3G network traffic

- Cellular network traffic:
  - Heavy traffic periods (e.g., video streaming or file transfer)
  - Inactive periods (e.g., not in use)
- In order to simulate the bursty nature of cellular network traffic, experiment employs an interrupted Poisson process with on-off periods

Arrival rate during *on* period

$$\lambda_{eff} = \frac{\beta \lambda_{on}}{\alpha + \beta}$$

Transition rates (between *on* and *off* periods)

Formula: expected delay

- Expectation of delay
  - Takeaway: when bottleneck processor is nearly fully utilized, as the buffer size K increases, the expected delay increases at a faster rate [how does one relate buffer size and delay time?]

$$E\left[D_{bottleneck}\right] = \frac{1}{\mu(1-\rho)} + \frac{(K+1)}{\lambda_{eff}}\left(1 - \frac{1}{1-\rho^{K+1}}\right)$$

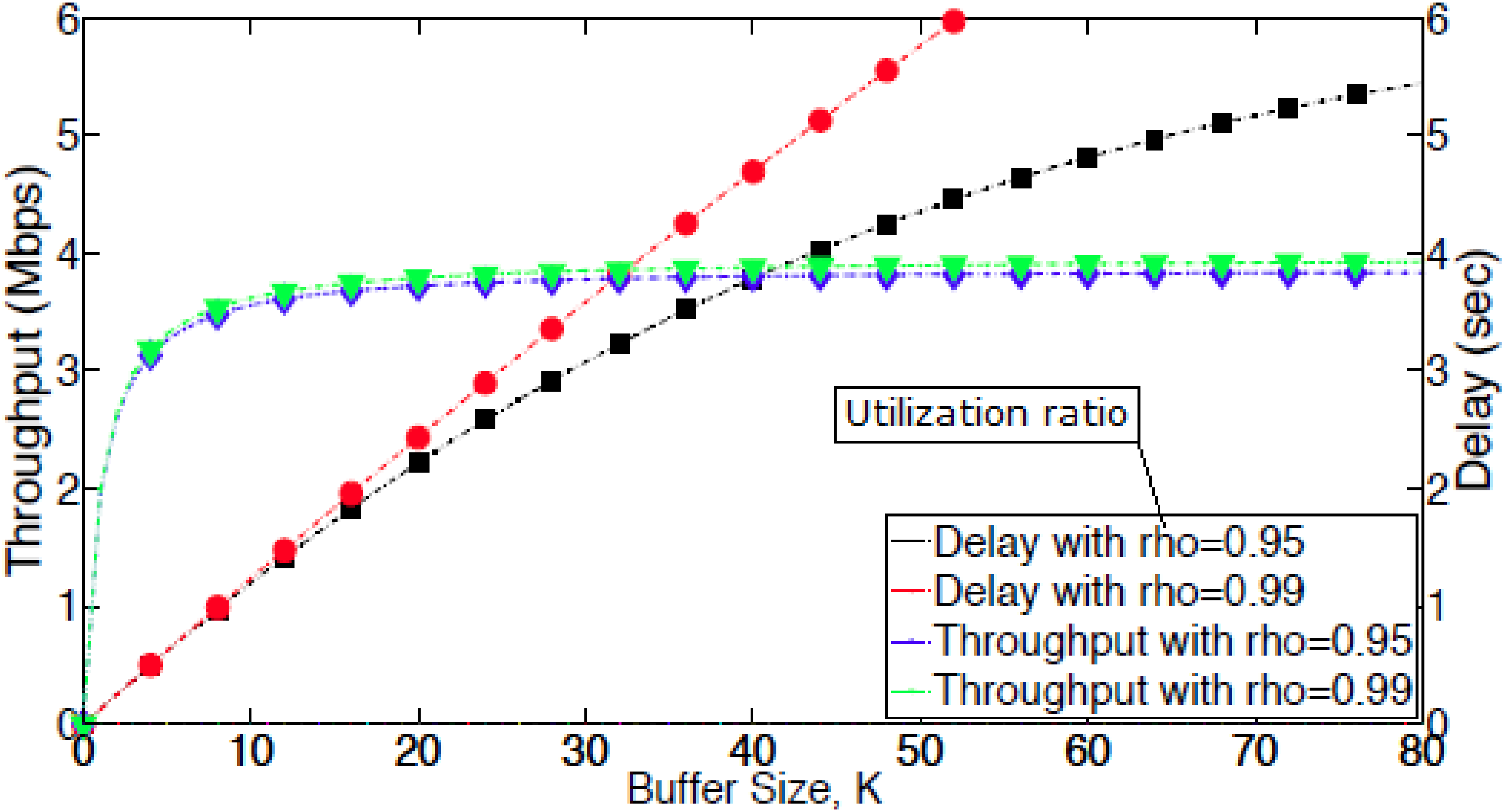
Formula: expected throughput

Expectation of throughput

- Takeaway: as the buffer size K increases, the expected throughput approaches a limit, so there are diminishing returns on performance

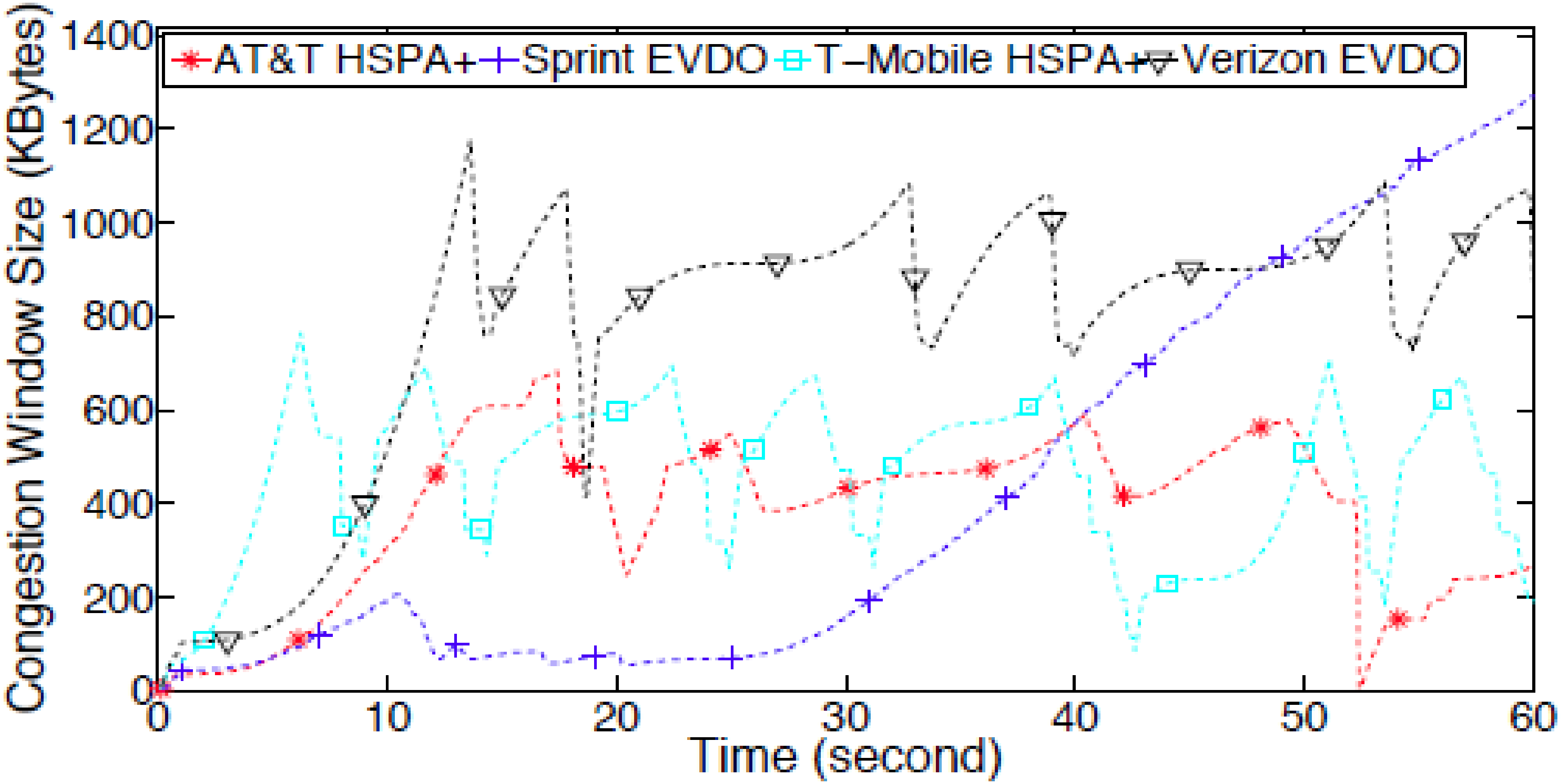
$$E[B] = \mu P_{working} = \mu + \frac{\lambda_{eff} - \mu}{1 - \rho^{K+1}}$$

Delay and throughput analysis



- Introduction to bufferbloat
- Authors' observations on bufferbloat in cellular networks
- Bufferbloat analysis
- **Analysis of the involvement of TCP**
- Existing and suggested solutions
- Some quick thoughts on this paper

TCP CUBIC behavior: *cwnd*

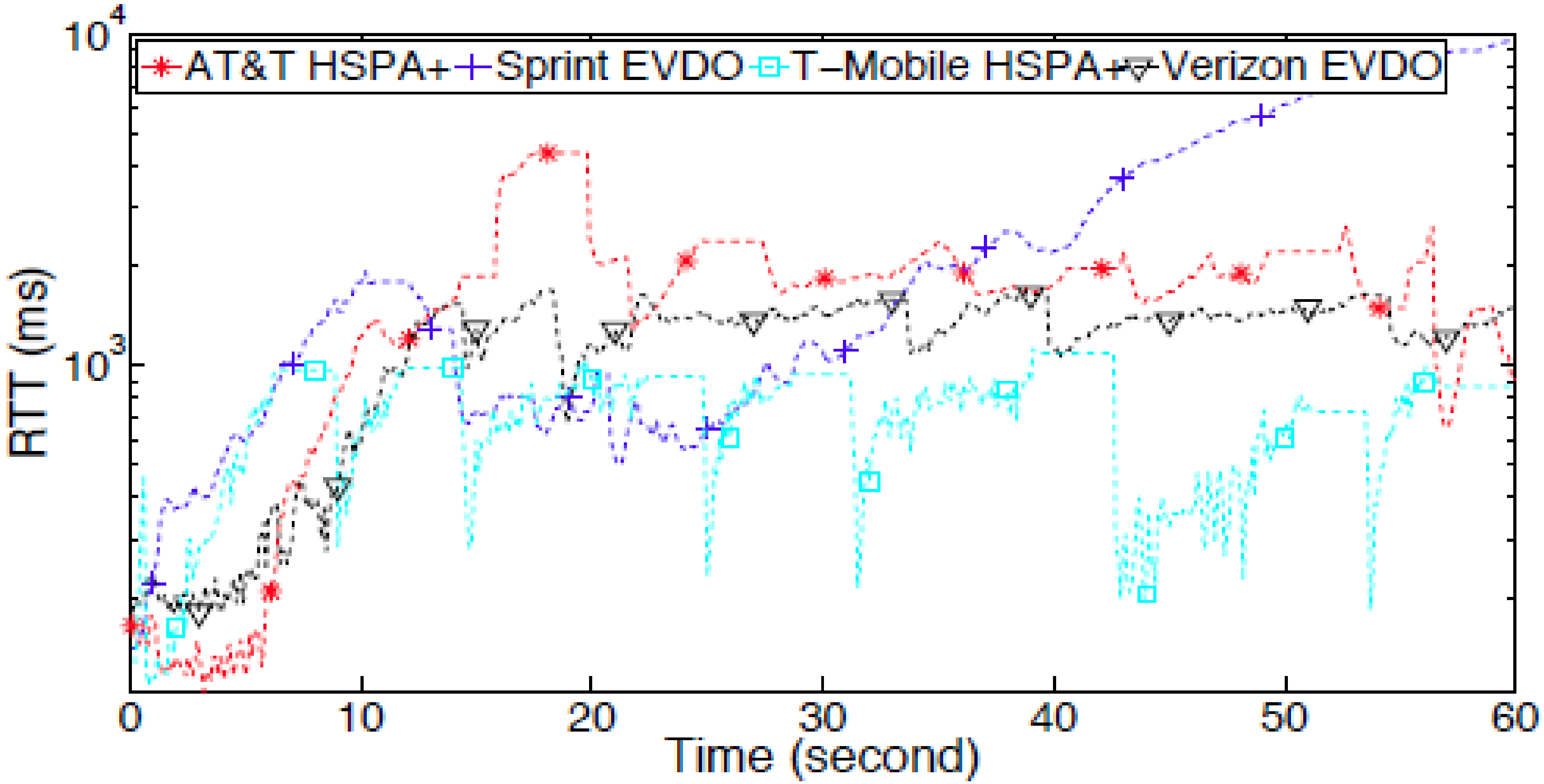


(a) Congestion Window Size

TCP CUBIC behavior: *cwnd*

- Why CUBIC? Paper source suggests the widespread use of “high-speed TCP variants such as BIC, CUBIC, and CTCP”
- Chart shows that the congestion window (*cwnd*) keeps increasing even if the size is beyond the bandwidth-delay product (BDP) of the underlying network
  - Example: EVDO BDP is approximately 58 KB, but *cwnd* increases far beyond that limit

TCP CUBIC behavior: delay

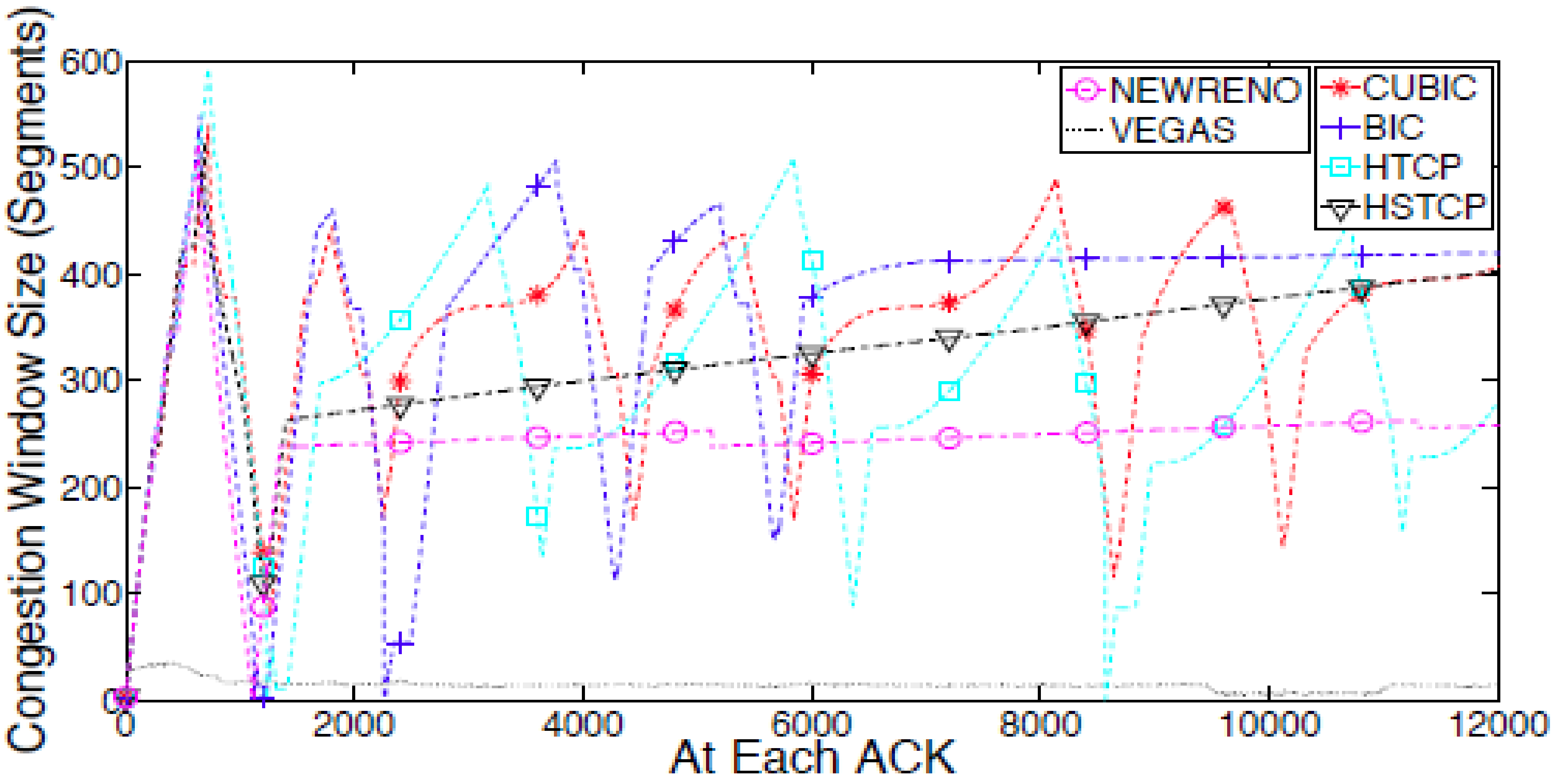


(b) Round Trip Time

TCP CUBIC behavior: delay

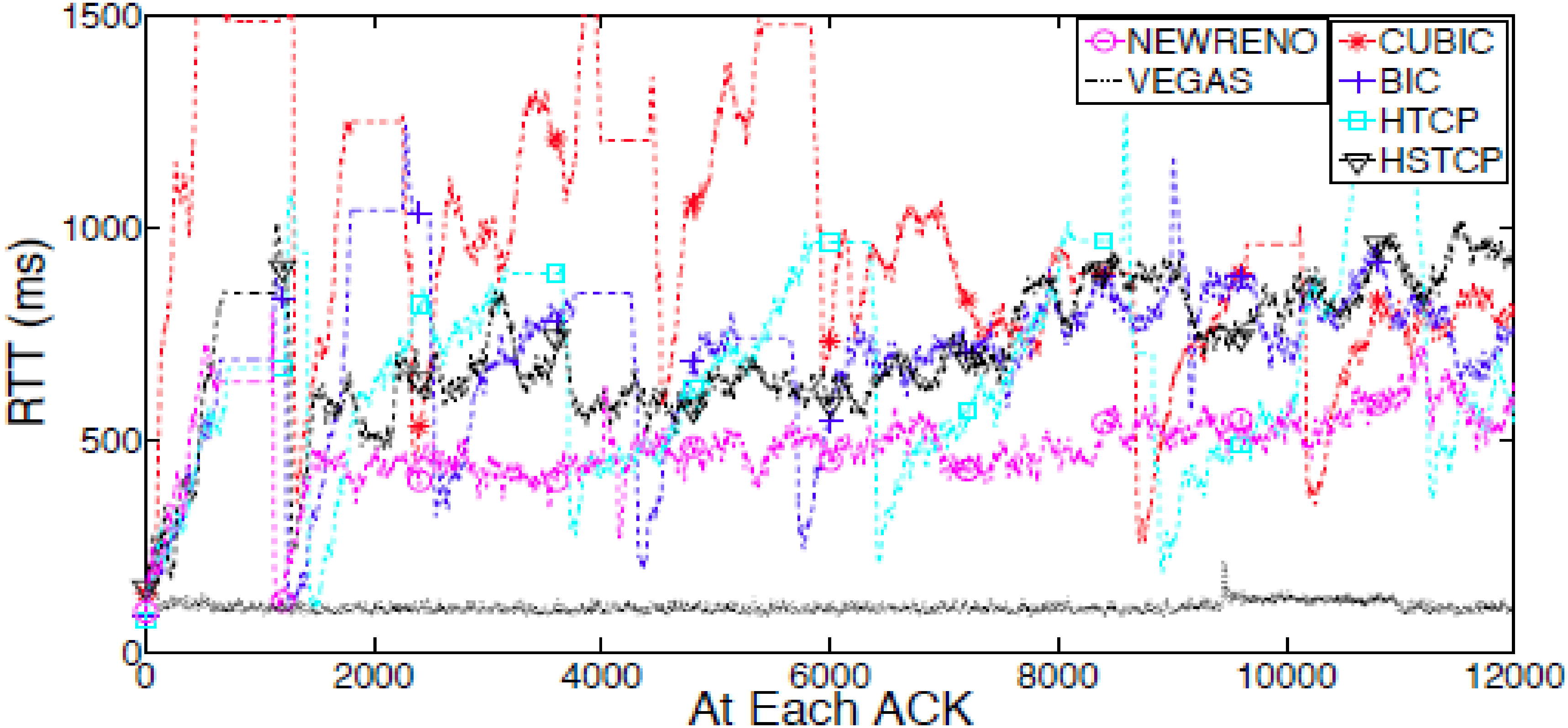
- The lengthy delays shown in the chart (up to 10 seconds) support the expected delay formula

The behavior of TCP variants



(a) Congestion Window Size

The behavior of TCP variants

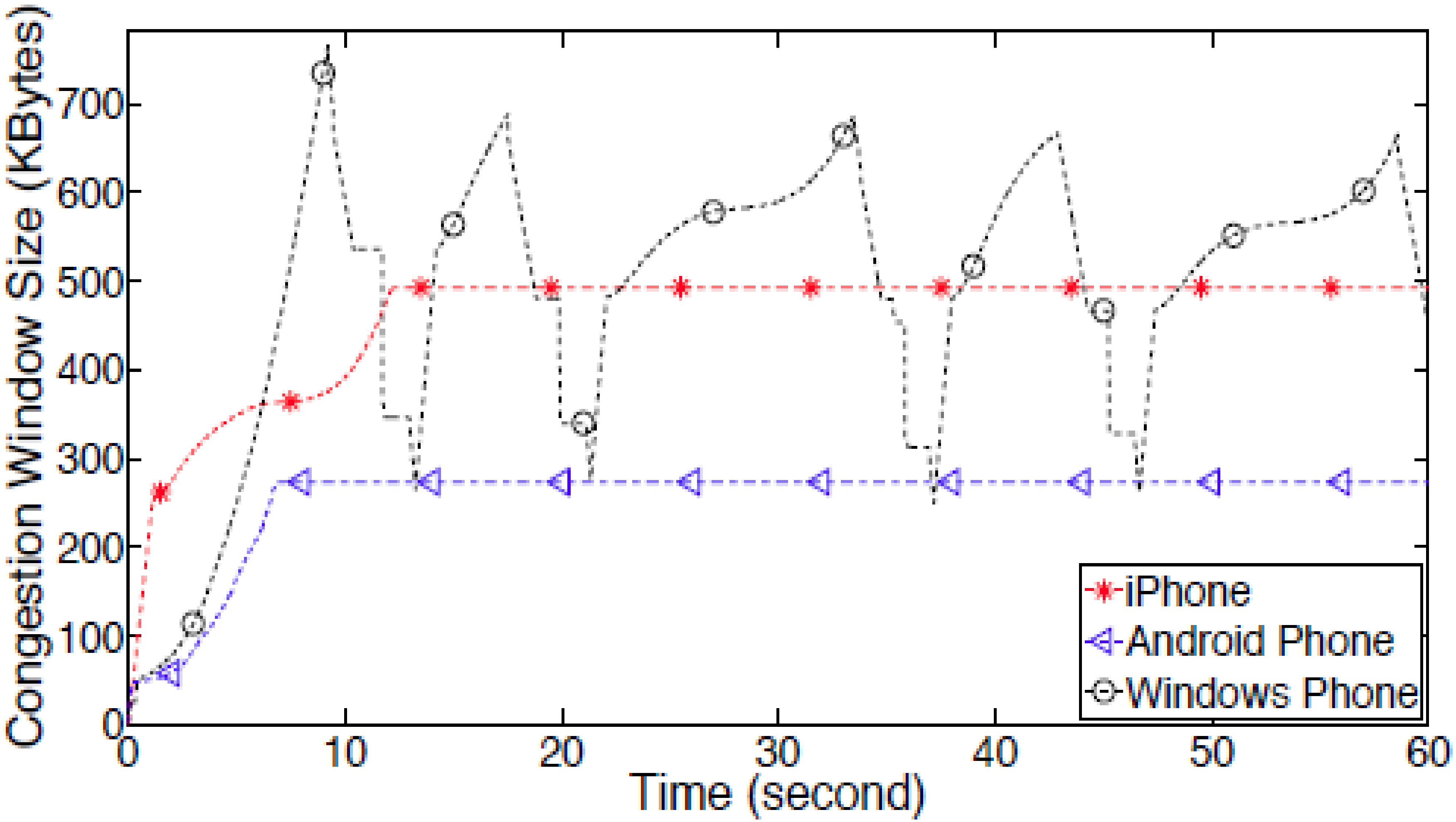


(b) Round Trip Time

The behavior of TCP variants

- The aggressive nature of high-speed TCP variants, combined with bufferbloat, results in “severe congestion window overshooting”
- TCP Vegas appears resistant to bufferbloat; this is because its congestion control algorithm is delay-based, not loss-based

The data behind the "untold story"



The data behind the “untold story”

- The Android and iPhone trials show a “flat TCP” pattern
  - *cwnd* hits a ceiling and remains flat until session ends
- The Windows Phone trials show a “fat TCP” pattern
  - This is characteristic of bufferbloat

- Introduction to bufferbloat
- Authors' observations on bufferbloat in cellular networks
- Bufferbloat analysis
- Analysis of the involvement of TCP
- **Existing and suggested solutions**
- Some quick thoughts on this paper

Existing solutions

- 1) The “untold story”
- 2) What the heck, let's just reduce buffer size
  - Aside from previously mentioned issues, reducing size would impact link layer retransmission and deep packet inspection
- 3) Incorporate Active Queue Management (AQM) schemes which involve randomly dropping or marking packets before the buffer fills (similar to RED) [this paper will never stop being referenced]
  - This carries the same challenges we've already seen (e.g., the complexity of parameter tuning or the purported limited performance gains in trying AQM)

Suggested solution

- Inspired by RED, modifying the TCP protocol itself has advantages:
  - More feasible than modifying routers
  - Easier and cheaper to deploy
  - More flexible; it may be server-based, client-based, or both
- Another factor to consider:
  - Delay-based TCP such as Vegas suffer from throughput degradation in cellular networks, replacing one demon with another

Suggested solution

- The authors suggest a TCP protocol that combines the favorable properties of both loss-based and delay-based congestion control while maintaining good performance across multiple network types (wired, WiFi, and cellular)
  - Dynamic Receive Window Adjustment (DRWA)
  - The solution is not presented in this paper; the authors forward the reader to another reference

- Introduction to bufferbloat
- Authors' observations on bufferbloat in cellular networks
- Bufferbloat analysis
- Analysis of the involvement of TCP
- Existing and suggested solutions
- **Some quick thoughts on this paper**

Review Notes

- Strengths
  - Interesting and prevalent topic
  - Establishes concern and highlights the issues behind bufferbloat
  - Provides good analysis of bufferbloat as it relates to major carriers
- Weaknesses
  - Riddled with grammar and spelling mistakes

WORCESTER POLYTECHNIC INSTITUTE

LEHR UND KUNST

1865