Dynamic Regulation of Mobile 3G/HSPA Uplink Buffer with Receiver-Side Flow Control

Yin Xu, Wai Kay Leong, Ben Leong  
National University of Singapore

Ali Razeen  
Duke University
Smartphones are everywhere

- US smartphone penetration exceeded 50% in Q2, 2012
- Mobile data traffic growing rapidly as well

Source: http://www.chetansharma.com/USmarketupdateQ22012.htm
Not just for surfing the web...

- Users uploading significant amounts of data in the form of photos and videos
  e.g. AT&T observed 40% more data uploaded than downloaded during a football match (7 Feb 2012)

- Uploads often conducted in the background
What are the users doing?

Hmm… I got the new HTC one X… lala

Hmm… I still have 10GB of data… lolo

Hmm… I got 25GB extra DropBox space!!

Hmm… Upload all my photos to DropBox!!!

Hmm… Show off my new phone!!!

Facebook… blabla….

ah!! I cannot refresh my Facebook wall!! Stupid phone!! Stupid network!!

I should complain!!

What happened?
The Problem

Background uploads can degrade downloads significantly!
Experiment Setup

- Android Phones
- 3 Local Telcos
  - 7.2 Mbps downlink
  - 2.0 Mbps uplink
Download Roundtrip Time

With Upload (s)

Without Upload (s)

ISP A
ISP B
ISP C
Download Throughput

With Upload (kbps)

Without Upload (kbps)

ISP A
ISP B
ISP C
Problem is exacerbated by low uplink throughput
Experiment Setup (Loopback)

Download 1MB

Continuous background upload
RTT dominated by uplink delay
NOT caused by ACK Compression

Data Pendulum Problem [Heusse et al. 2011]
- Sized properly, buffers take turns to fill up
- Sized improperly, low-speed link with large buffer becomes the sole bottleneck

Uplink is the bottleneck in a 3G/HSPA mobile network
Can we just size the uplink buffer correctly?
Understand the Problem

NOPE! CANNOT USE A FIXED SIZE BUFFER!
Previous Solutions

- Optimizing how the ACKs are sent
  [Balakrishnan et al. 1999/2002]

- Using different queues for data and ACK packets
  [Podlesny et al. 2012]

- TCP Vegas
  [Brakmo et al. 1995]

All Sender-Side Solutions
Why not Sender-Side Solutions?

- **Not General**
  - Only works for the devices already deployed with the solution
  - Devices may use network interfaces other than 3G/HSPA (e.g. Wi-Fi)

- “Implement complexity at the server, not the client”
  - It may take years to update client-side software [Adya et al. 2011]
Our Solution

Receiver-Side Flow Control (RSFC)
Our Approach

- Can implement at ISP network proxies
- Works transparently for any device using the 3G/HSPA mobile network
- Changes immediately deployable
Practical Deployment

Easily deployed at ISP proxy

A good place for RSFC
Receiver-Side Solutions

- Freeze-TCP [Goff et al. 2000]
- Reducing delay for interactive applications while maintaining throughput for bulk transfers [Spring et al. 2000]
- Improving fairness [Kalampoukas et al. 2002, Andrew et al. 2008]

Used for Other Purposes
Reduce # of packets in the uplink buffer by adjusting the TCP receiver window (rwnd)

What is the right rwnd?
Set to bandwidth-delay product (BDP)?

Not so simple…

- How do we estimate BDP?
- Network fluctuations
Approach: Negative Feedback

- Estimate time packet spends in buffer $t_{buff}$ using TCP Timestamp
- Set a threshold $T$
  - $t_{buff} > T$, clamp rwnd
  - $t_{buff} < T$, increase rwnd
Estimating $t_{\text{buff}}$

Timestamp:
$\text{TSval} = t_s$

$t_{\text{buff}}$

$\text{tu}$

$\text{time} = t_r$

$t_r - t_s = \text{RD}$
Relative Delay
Estimating $t_{buff}$

Timestamp: $TSval = t_s$

$t_u$

Sender

Packets in buffer

Receiver

$t_s$

Time

$t_r$

$t_r - t_s = RD_{min}$

Minimal Relative Delay

$t_{buff} = RD - RD_{min}$

No need to synchronize sender and receiver!
Estimating BDP

- Measure receive rate $\rho$ at receiver
- Minimal RTT ($RTT_{\text{min}}$)
- Ideal window is the bandwidth-delay product:
  $$rwnd = \rho \times RTT_{\text{min}}$$
Summary

- $t_{\text{buff}} > T$, $\text{rwnd} = \rho \times \text{RTT}_{\text{min}}$ (fast state)
- $t_{\text{buff}} < T$, $\text{rwnd}++$ (slow state)

- In our implementation
  - $T$ is set to $\text{RTT}_{\text{min}}$
Handling changes in the network

- Changes in bandwidth 😊
- Decrease in the delay 😊
- Increase in the delay
  - Slight increase: detect increased receive rate $\rho$
  - Large increase: monitor state

See details in paper!
Evaluation

- Reduces RTT
- Improves download throughput
- Reduces webpage loading time
- Fair and efficient
- Adapts to changes in network conditions
- Compatible with sender-side algorithms
Evaluation

- Reduces RTT
- Improves download throughput
- Reduces webpage loading time
- Fair and efficient
- Adapts to changes in network conditions
- Compatible with sender-side algorithms
Reduce RTT

Upload 1MB with Cube

Client -> Server

Server
Reduction in RTT
Reduction in RTT

![Graph showing the cumulative distribution function (CDF) of RTT with two lines: RSFC and Cubic. The x-axis represents RTT in seconds, ranging from 0 to 7, and the y-axis represents the CDF, ranging from 0 to 1.]
No Reduction in Throughput
Improve Download Throughput

Server

Client

Download 1MB (d1) with Cubic until download completes

Download 1MB (d0) without upload

Upload (u1)

Upload (u2) with RSFC until download completes
Better Downstream Performance

![Graph showing CDF of Downstream Throughput (kbps) with and without Cubic upload (d₁).]
Better Downstream Performance

CDF

With Cubic upload ($d_1$)
With RSFC upload ($d_2$)

Downstream Throughput (kbps)
Better Downstream Performance

CDF

With Cubic upload ($d_1$)
With RSFC upload ($d_2$)
Without upload ($d_0$)

Downstream Throughput (kbps)
Little Impact on Upstream

CDF

RSFC upload (u₂)
Cubic upload (u₁)

Upstream Throughput (kbps)
Improve Web Surfing

- Alexa top 100 sites

Web surfing without upload

Upload with RSFC until website is loaded
Webpages Load Faster

CDF

Time (minutes)

With Cubic upstream
Webpages Load Faster

CDF

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

0 2 4 6 8 10

Time (minutes)

With RSFC upstream
With Cubic upstream
Webpages Load Faster

CDF

Without upstream

With RSFC upstream

With Cubic upstream

Time (minutes)
Conclusion

- Saturated uplink can cause serious performance degradation
- Receiver-Side Flow Control
  - Reduces queuing delay significantly
  - Improves downstream performance
  - Reduces loading time of webpages
  - Compatible with existing TCP variants
  - Easily deployed at ISP proxies
THANK YOU
Fairness of Competing RSFC Uploads

- Run two RSFC uploads concurrently
- Calculate Jain fairness index:

\[
(R_1 + R_2)^2 / (2(R_1^2 + R_2^2))
\]
Fairness of Competing RSFC Uploads

CDF

RSFC Fairness

Fairness index

0.7 0.75 0.8 0.85 0.9 0.95 1

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Efficiency of Competing RSFC Uploads

- Run two RSFC uploads concurrently
- Compare the aggregate throughput to a single TCP:

\[
\frac{(R_1 + R_2)}{R_{tcp}}
\]
Efficiency of Competing RSFC Uploads

CDF

RSFC Efficiency

Efficiency
Adapting to Changing Network Conditions

- Compare with one RSFC version without:
  - Checking $\rho$
  - Monitor state
Adapting to Changing Network Conditions

Without the two methods

![Graph showing throughput and RTT over time with note: Delay increase cannot be detected. Inefficient!]
Adapting to Changing Network Conditions

With the two methods

![Graph showing throughput and RTT over time](image)

- Delay increase can be detected.
- Efficient!