Adaptive Flow Control for TCP on Mobile Phones

Shruti Sanadhya
Raghupathy Sivakumar

Georgia Institute of Technology

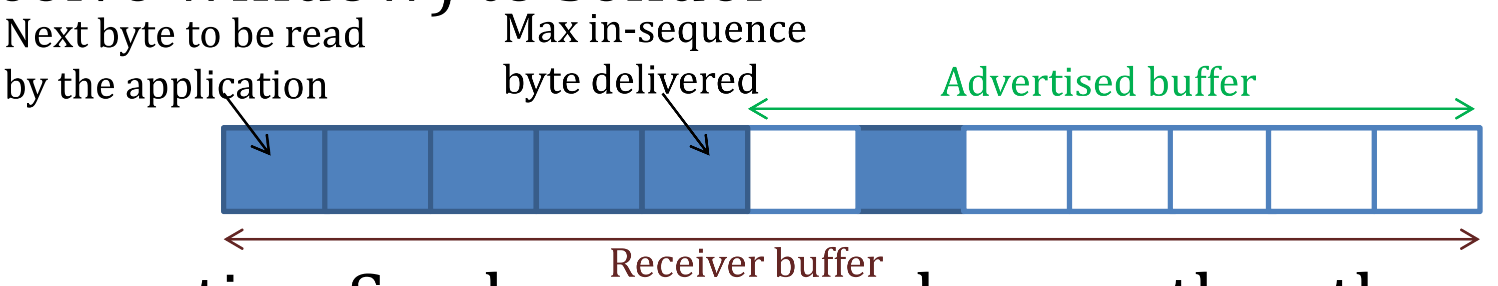
Introduction

- TCP employs flow control to prevent the sender from overwhelming the receiver
- Traditionally, flow control is not perceived to be a dominant function in transport layer operations
- Flow control assumes greater significance on resource constrained devices such as mobile phones

Focus of this work is to revisit the design of TCP flow control for mobile phones

TCP Flow Control

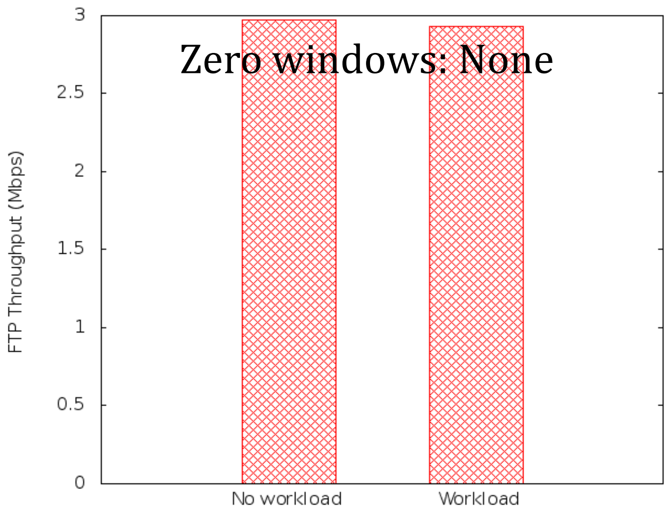
- Simple: Receiver advertises available buffer space (receive window) to sender



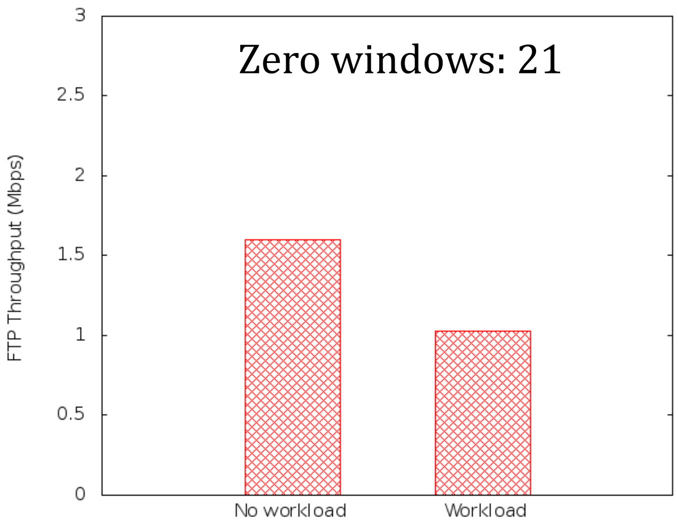
- Conservative: Sender never sends more than the advertised window
- *Zero windows* occur when receiver has no available buffer
- Sender cannot send data until it receives an *explicit open window* from the receiver

Flow Control on Mobile Phones

- FTP connection, on laptop and mobile phone, in the absence/presence of background workload



Dell Inspiron – Ubuntu 9.10 OS

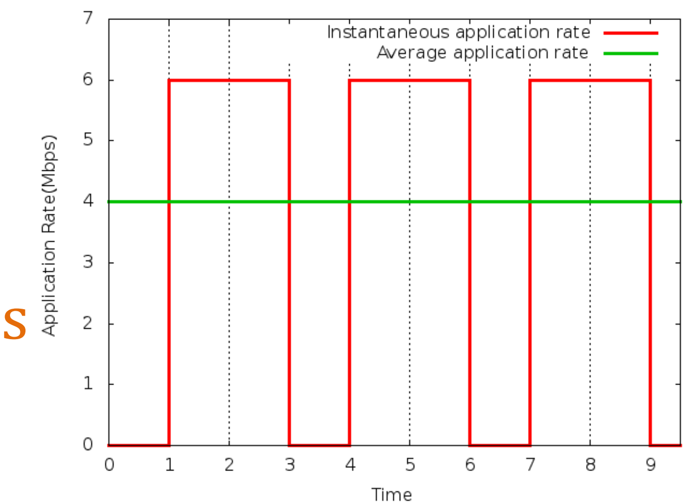


HTC G1 – Google Android OS

Impact of flow control is greater on resource constrained mobile phones

Flow Control on Mobile Phones

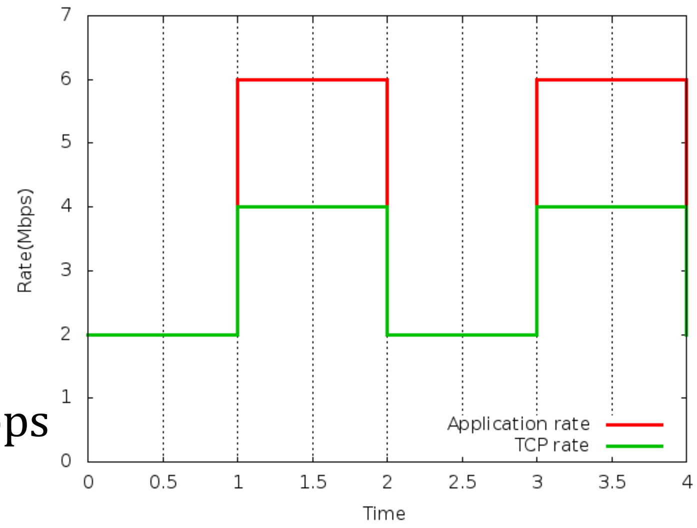
- NS2 simulation of TCP on mobile with fluctuating application rate
  - Sender-receiver on 15Mbps(NW) link
  - Application periodically reads at <0,6,6> Mbps
    Average AR(AAR): 4Mbps
  - Round trip time: 530ms
  - Expected throughput = 4Mbps
  - Observed throughput = 1.45Mbps (63% degradation!)



TCP flow control does not track the application read rate effectively

Limitation 1: Buffer Dependency

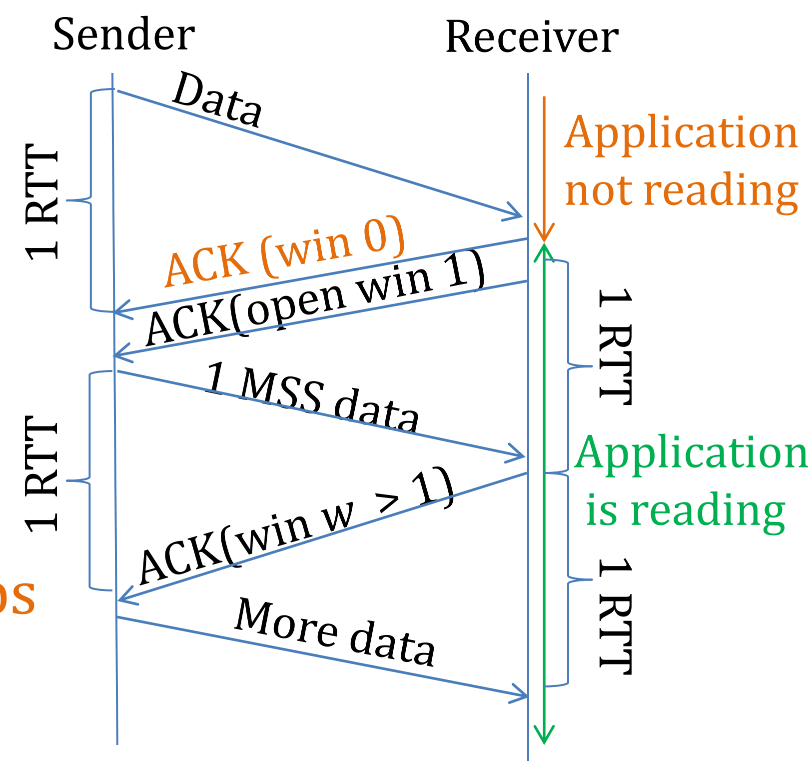
- Fluctuating application read rate:
  - NS2 simulation
    - RTT = 1s
    - Network rate(NW) = 4Mbps
    - Receive buffer = 512KB
    - Application profile = <2,6> Mbps
    - Avg. application rate (AAR) = 4 Mbps
  - Expected throughput: 4Mbps
  - Observed throughput: 3Mbps



Connection rate is capped at Buffer/RTT even when the application reads at a higher rate

Limitation 2: Zero Windows

- Zero window events
  - NS2 simulation
    - Network (NW)= 15Mbps
    - RTT = 530ms
    - Application: <0,6,6>Mbps
    - Receive buffer = 256KB
  - Expected throughput: 4Mbps
  - Observed throughput: 1.45Mbps
    - 328 zero windows
    - 656 idle RTTs of 1132 RTTs



At every zero window, sender waits for up to 2 RTTs before it can send any substantial amount of new data

Limitation 3: Buffer Auto Tuning

- Buffer auto tuning with fluctuating network rate
  - NS2 simulation
    - RTT = 530ms
    - Application profile = <0,6,6> Mbps (AAR = 4 Mbps)
    - Network profile = <2,4,4>Mbps, (Avg NW=3.3Mbps)
    - Receive buffer = min(*perceived* NW,AAR)*RTT (auto-tuning) = min(2, 4) Mbps*RTT = 128KB
  - Expected throughput: 3.3 Mbps
  - Observed throughput: 0.67 Mbps

Lower throughput rates observed when application read rate is low also hinder the growth of auto-tuning buffers

Theoretical Model of TCP

- TCP flow control is a closed loop system
- The equation for the advertised window is:
  - $W = \min(B_0, \int W' \, dt)$ .... (1)
- The target value of TCP is AR, thus
  - error = AR – TCP = W' .... (2)
- If network is not the bottleneck, classical flow control has:
  - TCP = α W, where $\alpha = 1/RTT$ .... (3)
- Using (1) in (3) and considering the time dependent part:
  - $TCP = \alpha \underbrace{\int W'}_{\text{error}} dt$ .... (4)

TCP is an integral controller

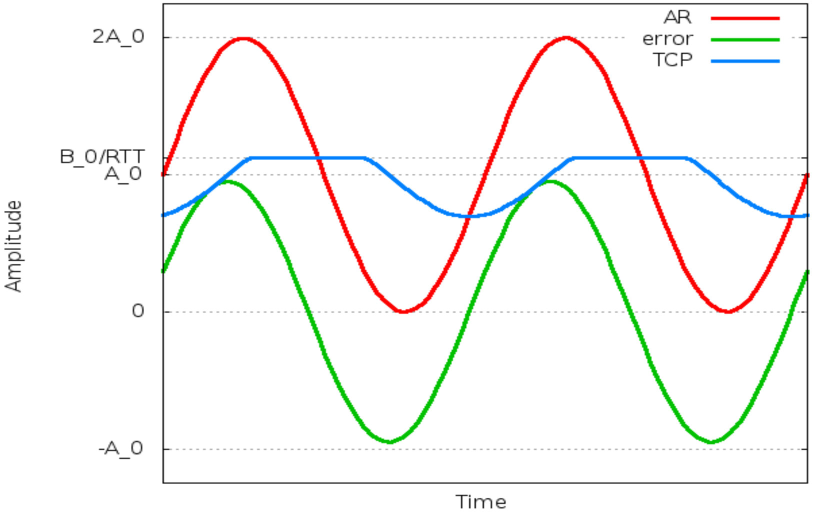
TCP : Connection rate
B_0 : Receive buffer size
B : Filled buffer
W : Advertised window (B_0 – B)
AR : Application rate
RTT : Round Trip Time

Theoretical Analysis of TCP Model

- Assuming $AR = A_0 (1 + \sin \omega t)$
  - error $= A_0 \sin\theta(\cos(\omega t - \theta))$, where $\theta = \tan^{-1}(\omega/\alpha)$ .... (5)

For fluctuating applications:

- *the error in TCP is non-decaying*
- *the amplitude of oscillation of error increases with the peak application read rate*



- Using (5) to derive TCP,
  - $TCP = \min(\alpha B_0, A_0[1 + \cos\theta \sin(\omega t - \theta)])$ .... (6)

Thus, with current flow control, TCP :

- *cannot converge to the application read rate*
- *is limited by B_0/RTT, if B_0 is not large enough*

Adaptive Flow Control

#1 Adapt to application rate:

- Add a corrective term AR in " TCP = α W "
  - TCP = α W + AR
  - error = $-\alpha B_0 e^{-\alpha t}$, which decays over time
  - TCP = $\alpha B_0 e^{-\alpha t}$ + AR , converges to AR over time

- Flow window = Advertised window + AR*RTT
  - Advertised Window: Classical flow control window
  - AR: Exponential weighted moving average of rate at which the buffer is drained
  - RTT: Round trip time

Adaptive Flow Control

#2 Handle buffer overflows

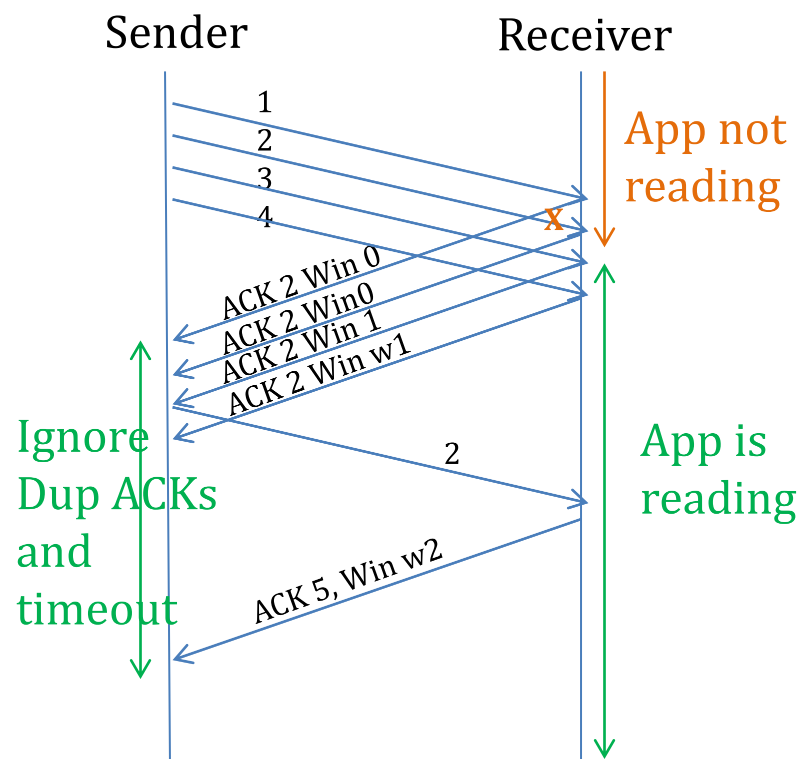
- Hide buffer losses from congestion control
- Ignore congestion indicators after zero window till fresh data is acknowledged

#3 Proactive feedback

- Receiver sends feedback to the sender whenever application rate changes drastically

#4 Burst control

- Delay packet transmission to avoid bursty traffic

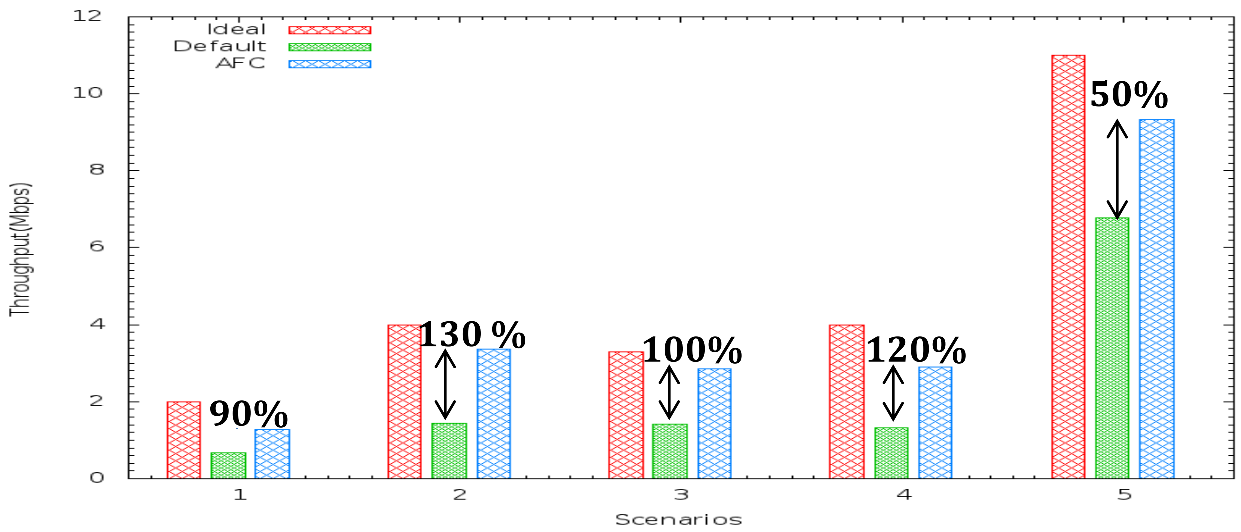


Performance Evaluation

- Evaluation methodology
  - NS2 with classic TCP flow control considered as default
  - AFC is Adaptive flow control implementation in NS2
  - SACK is enabled in all cases

Performance Evaluation (Contd.)

- Throughput analysis
  - Topology: Sender and receiver on a direct link
  - Link bandwidth (2-15Mbps) and delay (200-300ms)
  - Receive buffer = min(Avg AR, Avg NW)*RTT
  - *Application fluctuates in all scenarios*



#1: AAR < NW
#2: AAR > NW
#3: Network fluctuate, AAR < Avg NW
#4: Network fluctuate, AAR > Avg NW
#5: Network fluctuates, Max NW > 3* Min NW, AAR > Avg NW

Conclusion and Future Work

- TCP under-performs in flow control constrained connections, e.g. those on mobile phones
- Presented theoretical analysis of TCP flow control
- AFC shows considerable improvement in throughput across multiple scenarios
- Future work
  - Avoid unnecessary re-transmissions
  - Interplay of congestion control and flow control

Thank you!

Send questions and comments to
shruti.sanadhya@cc.gatech.edu