Real-time Interactivity Over 5G Networks

Jerry Chen, Daniel Qian, Jason Zhang

Background

New 5G mobile networks will provide more bandwidth to users, but more importantly, will also provide much lower latency (according to carriers)

- 5G: Below 10ms
- 4G LTE: 40-100ms

This is beneficial for applications in which users must interact with each other in real time (also autonomous vehicles)

We wanted to investigate how these properties can be used by developing our own real time protocol

Our Approach

1. Research and experiment with characteristics of cellular networks
2. Develop a real time, proactive, bandwidth estimation protocol that works over cellular links
3. Implement protocol in a proof of concept Android application that also demonstrates interactivity
4. Compare our protocol's performance on 5G networks and various 4G LTE carriers

Cellular vs Wired Links

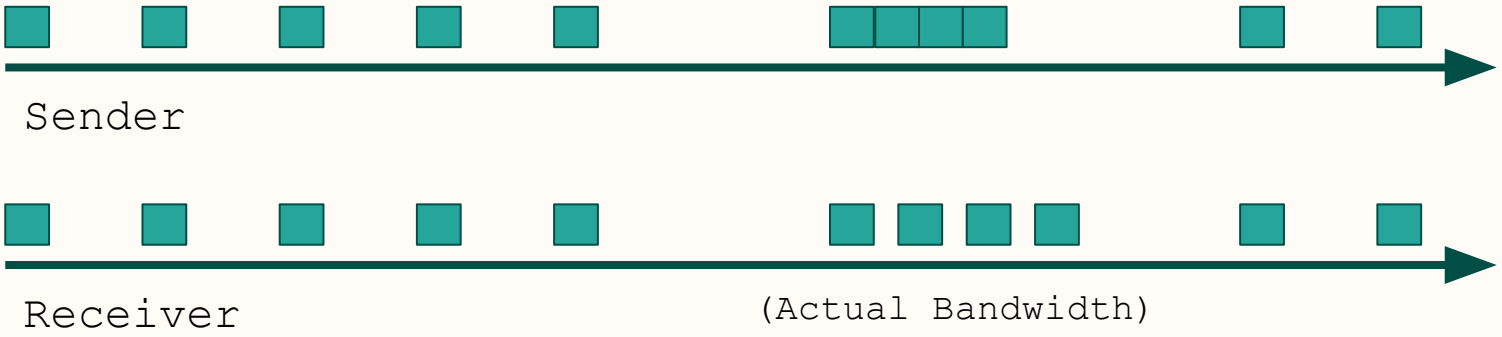
From research and initial experiments:

- Builds up large queues instead of dropping → latency spikes!
  - Necessitates proactive bandwidth estimation
  - We will demonstrate this later
- Packet bunching
  - Base towers can aggregate packets
  - Mobile phones use a separate baseband processor to handle network I/O
  - Not sure exactly how it works, but out of our control

Note our model assumes the cellular link is always the bottleneck

Protocol

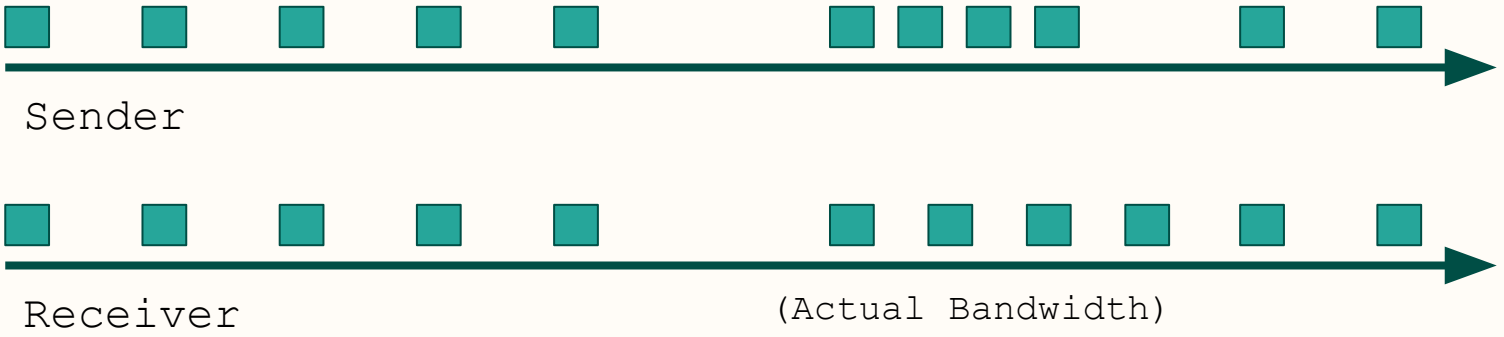
Our initial idea was to send a instantaneous burst of packets and measure interarrival times of the burst



However we ran into issues due to packet bunching

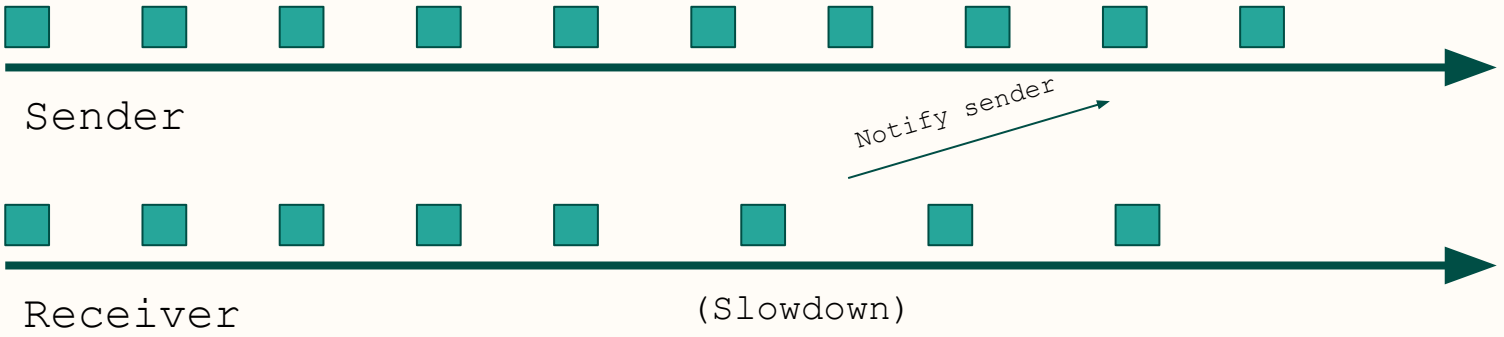
Protocol

Instead simply periodically send at higher (2x) rate. Can still detect underutilization of capacity.



Protocol

If at any point the bandwidth measured on the receiver side is less than sent, we send a control message and back off.



Protocol - Implementation Details

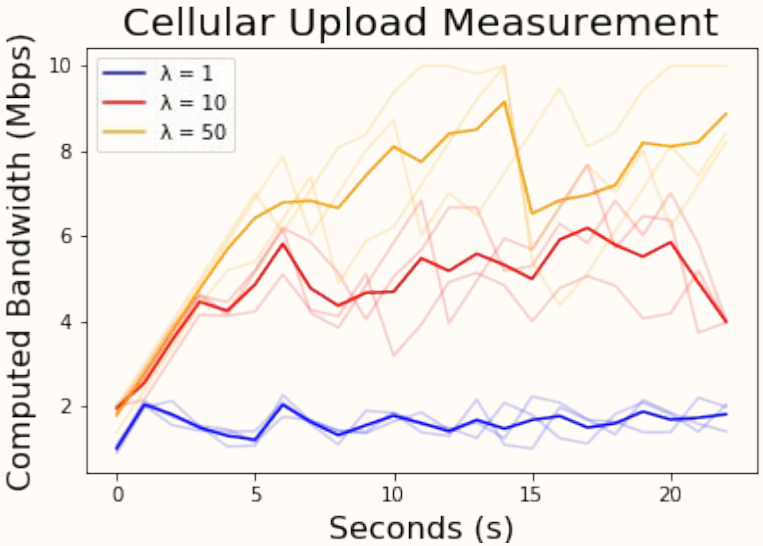
Sender

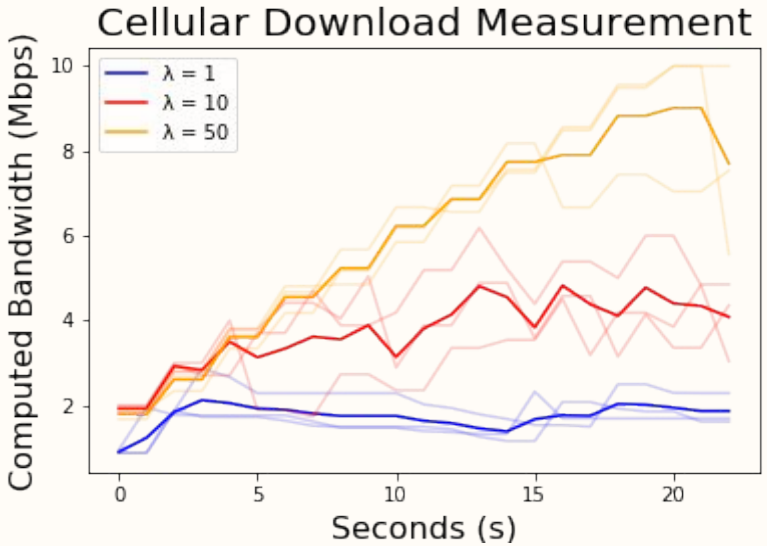
- Appends sending rate to header on every packet
- Frequency of bursts can be configured
- When a higher bandwidth is detected, increases are linear (+1 Mbps)

Receiver

- Measures received bandwidth using either running average or EWMA
- Grace period (λ) accounts for short period of lower bandwidth or delayed packets by waiting a certain number of packets before sending feedback
  - Higher $\lambda \rightarrow$ Less responsivity, but better utilization

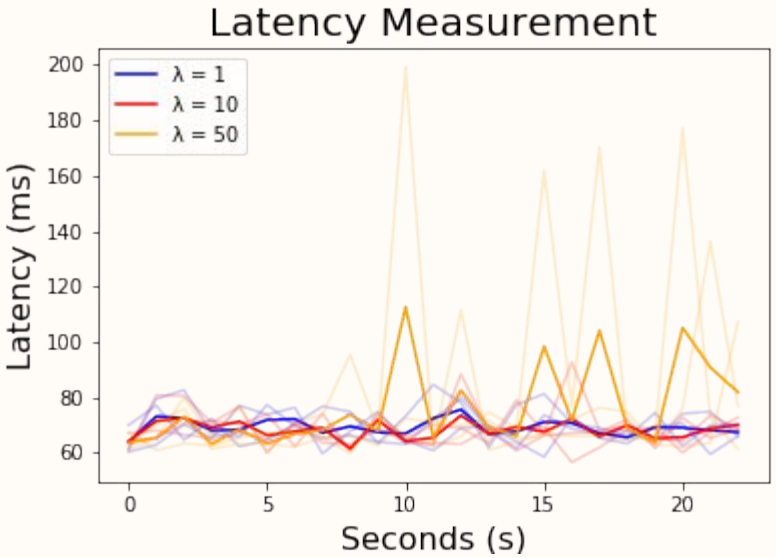
Test Results (changing λ)





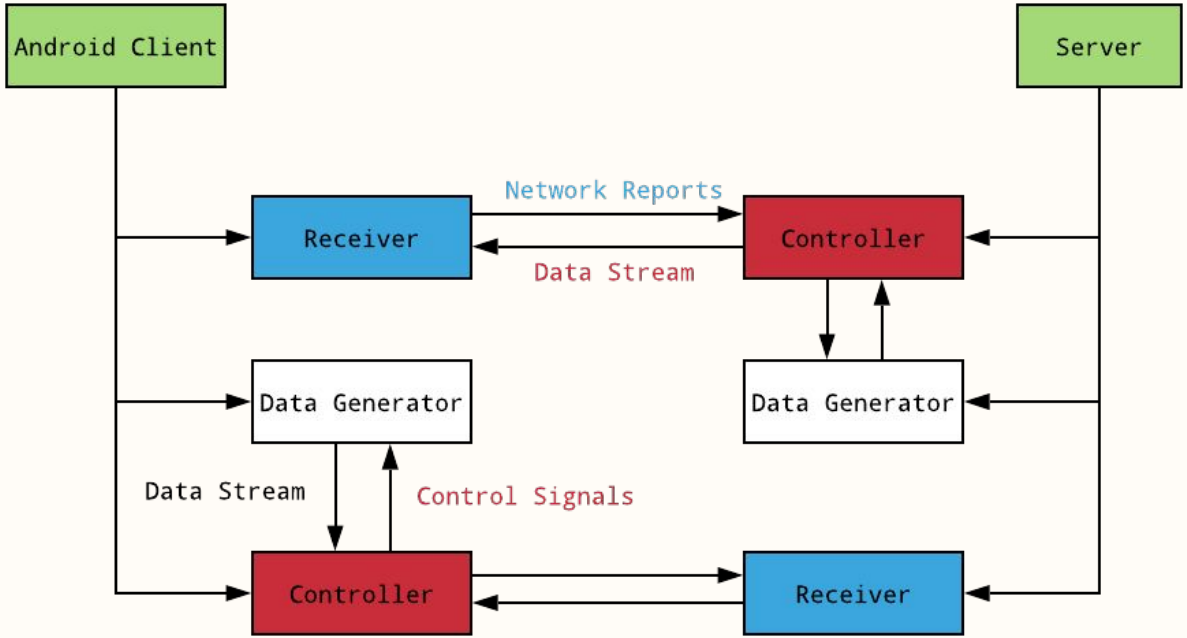
- Using T-Mobile 4G LTE, run for 20 seconds and record results from beginning
- Faint lines are each run, solid lines are average

Test Results (changing λ)



- Although we achieved better bandwidth with higher λ, we no longer have consistent latency!

Architecture

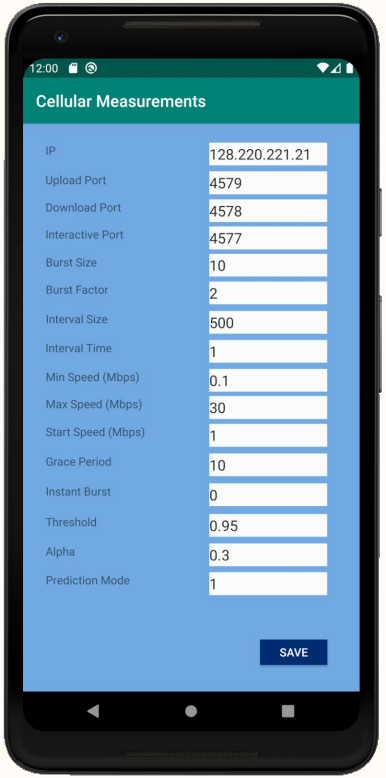


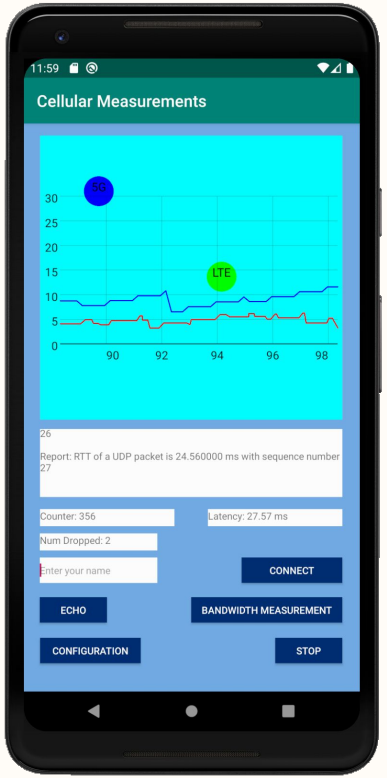
How We Developed the App

- Bandwidth Application
  - Bidirectional bandwidth measurement
- Interactive Application
  - Multi-user client-server program
  - Used to measure timeliness
- Native Android using JNI
  - Android activities calls C++ functions through JNI (Java Native Interface)
  - C++ functions starts sender, receiver, and interactive programs
  - Compiled on Android
  - Running on multiple Java threads

Bandwidth Application

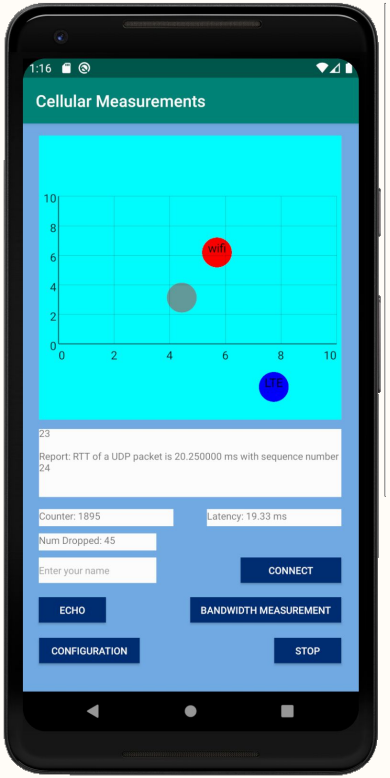
- Measures upload/download speed
- Supports both TCP and UDP
- Real-time Graph
  - Blue: download (Mbps)
  - Shows server sending rate
  - Red: upload (Mbps)
  - Shows client sending rate
- Configurable
  - Tune parameters





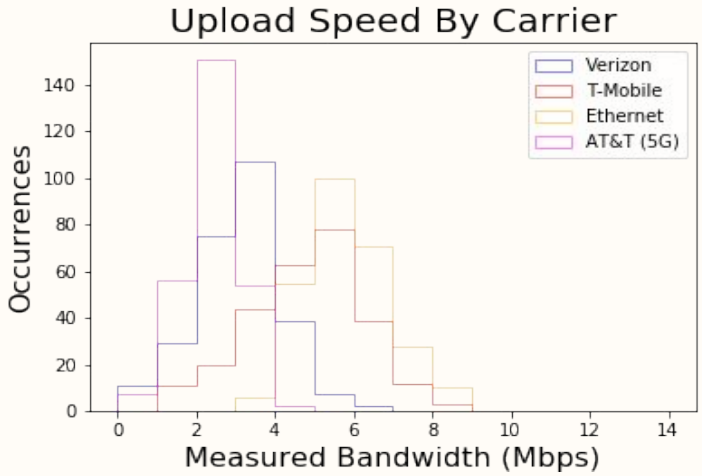
Interactive Android Application

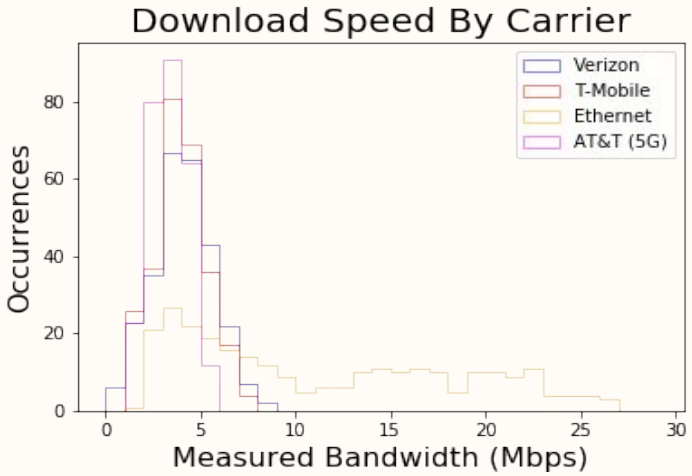
- Echo: measures RTT
- Multi-user interactions
  - Each user can connect with a name
  - Latency: RTT of the last sent packet
  - Count: number of sent packet
  - Num Dropped: number of lost / out-of-order packets
  - To better illustrate the latency, the circle with light color shows the last sent packet, and other circles represent locations of received packets



Demo

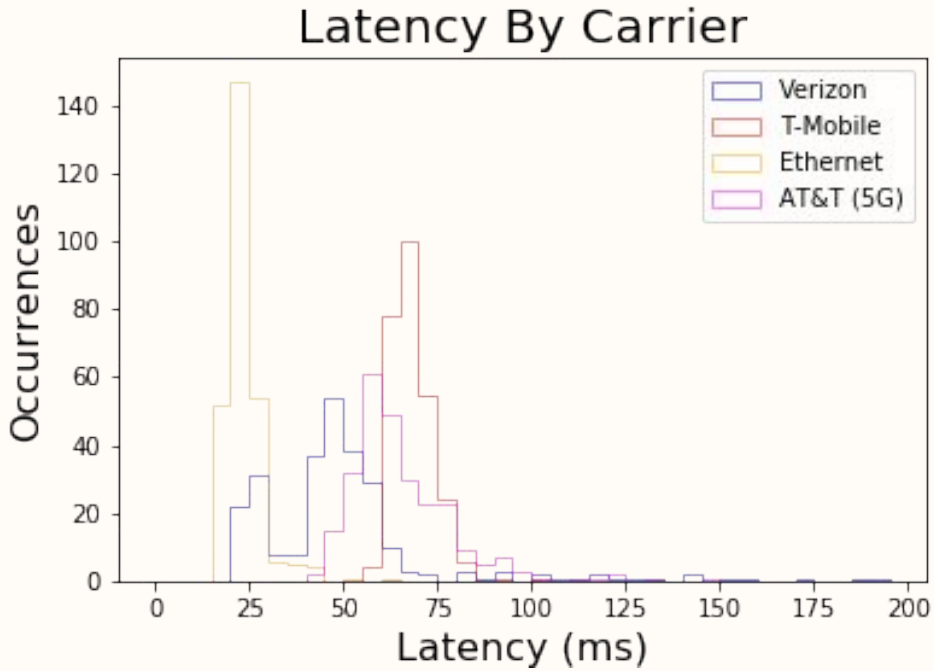
Test Results (Cellular Up/down)





- Ran protocol for 2 minutes in three separate trials, sampling speed and latency every second
- Discarded first 10 seconds (startup) and created histogram for each statistic

Test Results (Cellular Latency)



Potential Areas for Further Work

- Try to turn it into an API
- Experiment with more sophisticated (ie. statistical) techniques to estimate bandwidth on receiving side
- Support useful stream of data, ie. video
- Apply to real time applications, such as Augmented Reality

Questions?

References

Real-time Bandwidth Prediction and Rate Adaptation for Video Calls over Cellular Networks

Estimating Packet Arrival Times in Bursty Video Applications

Android NDK

Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks