Three Papers on Congestion Control

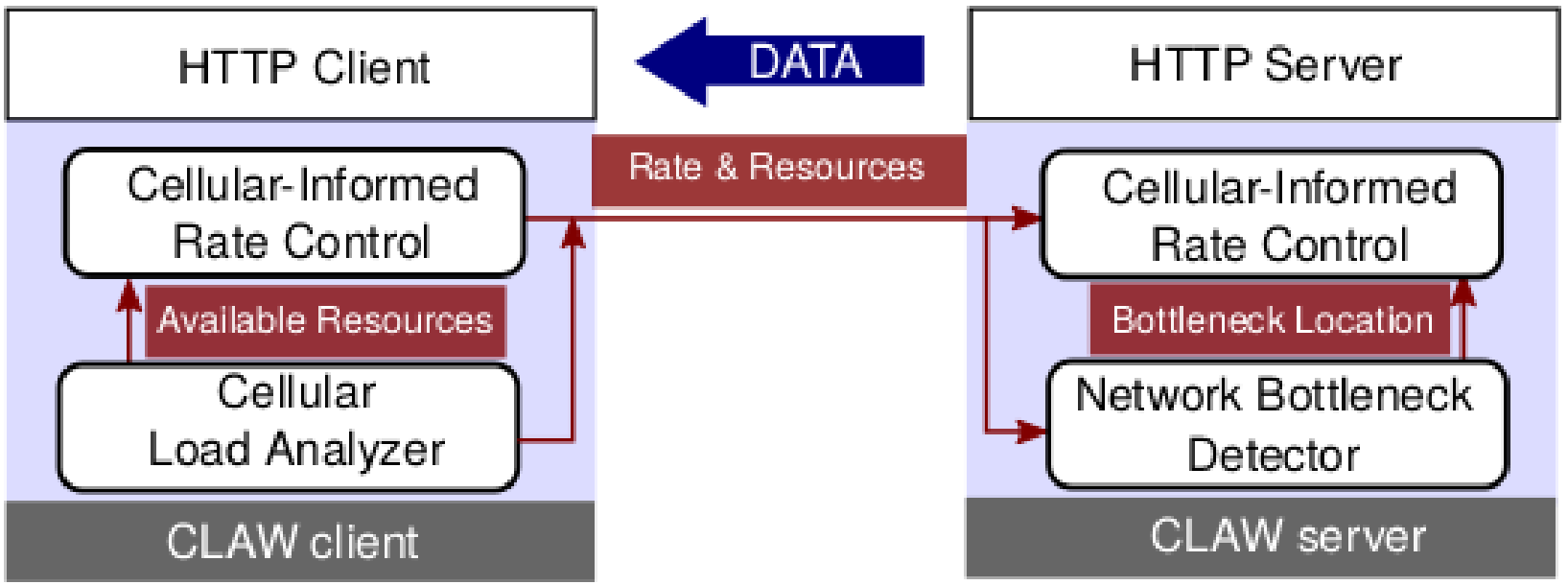
Kevin Xue

June 20, 2019

Overview

1. CLAW
2. Sprout
3. PropRate

Accelerating Mobile Web Loading Using Cellular Link Information (CLAW)



Challenge for Web over LTE

- RTTs in LTE fluctuate a lot. A temporary RTT surge may mislead TCP congestion control to regard it as a loss. We need least number of round trip probes to converge to the network bandwidth.
- TCP's multiplicative decrease overreacts to LTE link losses.
- Short mobile web loading flows cannot let TCP quickly converge to network bandwidth. Bandwidth estimation should not be related to source bit rate.
- Obtaining cellular load directly requires full knowledge of LTE BS's resource allocation, which is difficult in commodity mobile devices.

Idea

CLAW's design includes:

- cellular link analyzer that computes available resources allocated to a single client.
- cellular informed rate control to find the optimum bandwidth.
- network bottleneck detector to address the minor case that wireline is the bottleneck

Cellular Link Analyzer

Goal: calculate the cell load: the volume of allocated radio resources on the downlink channel.
Given information: RSSI, RSRP, RSRQ, MCS Fraction of allocated RBs in the $k-th$ subframe is:

$$r_k = A_k / N_{rb} = \frac{Q_k^{-1} - N_{ref}}{(N_{subc}^{rb} - N_{ref}) N_{ant}}$$

Then, within a sliding window T, we can compute the number of RBs allocated to all clients as well as the number of idle RBs.

$$N_I = N_t - N_a = 2TN_{rb}(1 - L)$$

S_k, The number of RBs allocated to the client itself, can be read from the phone's dignostic interface. Sum them up and we obtain

$$N_s = \sum_{k=1}^{T} S_k$$

And the number of available RBs in this window T is $N_f = N_I + N_s$

Cellular-Informed Rate Control

Use $F(\cdot)$ to map MCS and number of RBs per subframe (N_f / T_i) to transport block size (number of bytes that can be carried over a subframe). Further multiply it by T_i to obtain the potential capacity

$$\hat{B} = T_i F(N_f / T_i, \overline{MCS})$$

Ideally, $\hat{B}$ is the BDP. Client sends this to the server and server decides W_{claw}. This causes only one RTT.
New flows will use the maximum of cwnd calculated by cubic and claw to ensure fairness.
RLC-layer statistics are used to distinguish link loss from congestion.

Network Bottleneck Detector

CLAW compares the volumes of resources N_f available to a client and its consequently allocated amount of resources U after one RTT,

- If $U < \delta N_f$, detector declares a wireline bottleneck. Falls back to TCP-style control
- Bottleneck is asserted only after consistent observation over a sliding window of 3 RTTs.
- δ is set to 0.8 empirically.

Implementation

For client side,

- Android phones with Qualcomm chipsets are used.
- Read output logs by Qualcomm's QCAT.
- Modify `tcp_output.c` to set extra bit of confirming rate control algorithms

For server side,

- Test website is an Apache HTTP/2 server, which also acts as a CLAW sender.
- Upon receiving the estimated cwnd size by client, pass it to the rate control algorithm.
- Records packet losses, retransmissions, ACKs and behaviors of CLAW in real time.

Evaluation Setup

- Benchmark against CQIC and Verus.
- CQIC is upgraded from original HSPA+ to LTE and Verus is implemented in the Linux kernel.
- Websites are on the list of Alexa top sites. Elements may include dynamic JS ones.
- PLT is measured using Chrome Developer Tools.
- Phones run chrome browser with cache option disabled.
- Configure phones to LTE only.

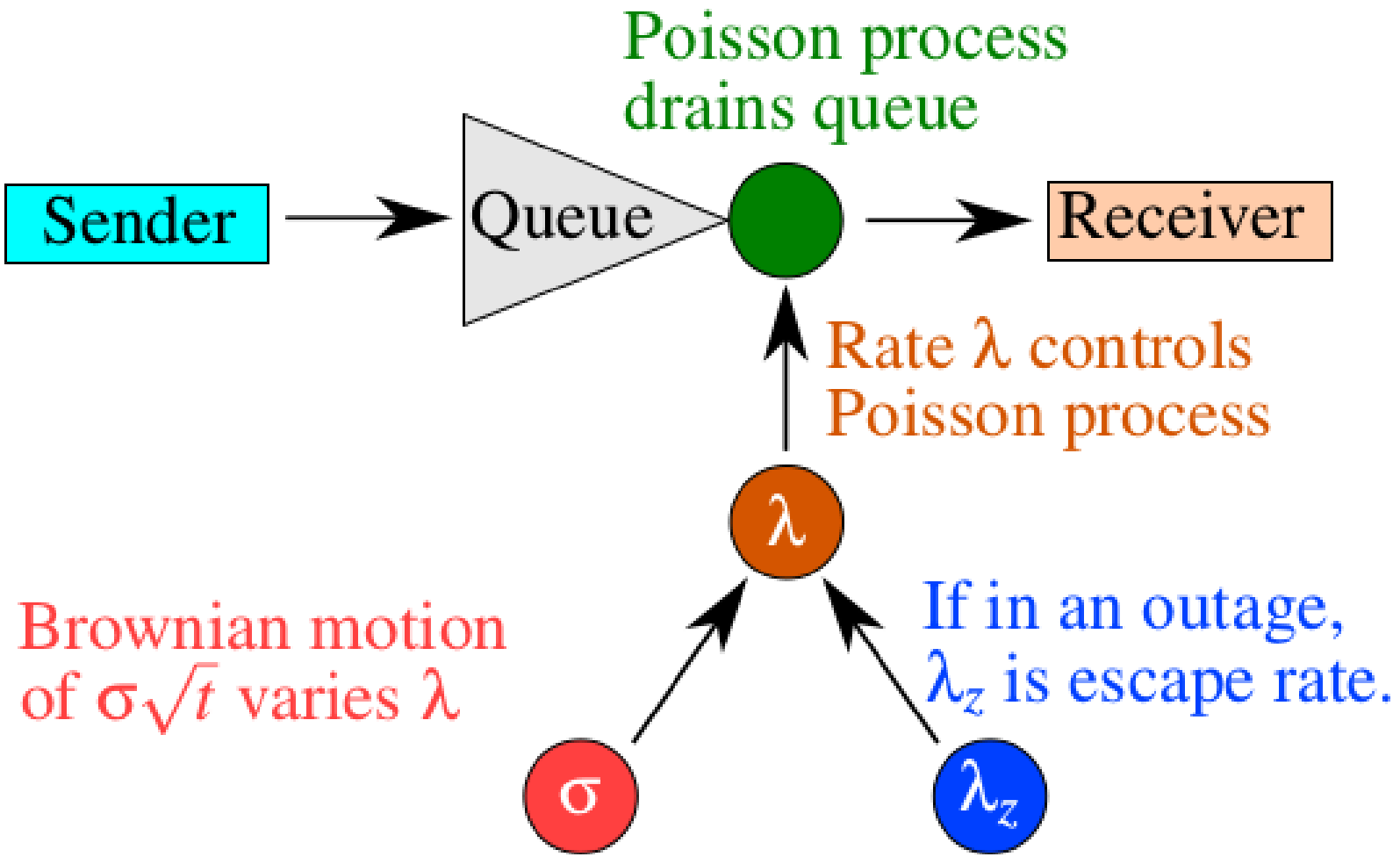
Micro-benchmarks

- Cellular load anaylzer: Generate `iperf` TCP traffic stream with linearly increasing bitrate. Smoothed cell load follows the traffic rate measured by `tcpdump`.
- Cellular-informed rate control: Load the same webpage using both CLAW and CUBIC and observe the cwnd size and LTE resource utilization.
- Bottleneck detector: Manually throttle the web server's network interface at 30kbps using `Dummynet` kernel module. Without the detector, CLAW's performance becomes comparable to CUBIC. With it, CLAW outperforms CUBIC by 20%.

System Level Test

- Load 20 selected pages from top 200 Alexa list and compare PLT with CUBIC and other transport protocols.
- Under different cell loads: run at different time of the day.
- Under different LTE channel qualities.
- Under different outdoor mobility scenerios.
- Over different cellular operators.
- Over different LTE hardware: Nexus 5 and Nexus 5X
- With HTTP/1.1 and HTTP/2
- TCP friendliness: Evaluate PLT on different combinations of two protocols.

Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks (Sprout)



Motivation, Context and Challenges

- Cellular wireless network is not stable. Bandwidth varies rapidly and outages occurs occasionally.
- Base stations schedules isolates queues of different users.
- Traditional methods may either over-buffer the queue or under-utilize the link.

The challenges for a good transport protocol:

- Link rate variations.
- Over-buffering results in high delay while under-utilization leads to low quality.
- Outages should be dealt with.

Basic Idea of Sprout

The interarrival packets follows a Poisson process. If the rate λ can be estimated using a certain model, it may help when infering how many bytes the link will be willing to transmit from its queue in near future.
Idea:

- λ itself varies in a Brownian motion. This property can be used to get a more precise value of λ.
- We can conduct an inference update procedure every 20ms (a tick) to determine the most appropriate value of λ in 256 discrete values sampled uniformly from 0 to 1000 MTU-sized packets per second.

Evolution of the probability distribution on λ

At every tick, Sprout conducts the following steps:

- Evolves probability distribution based on Brownian motion to each $\lambda \neq 0$. For $\lambda = 0$, use outage escape rate.
- Observes the number of bytes that actually came in during the most recent tick. Update the estimates of probability:

$$F(x) \leftarrow P_{old}(\lambda = x)\frac{(xT)^k}{k!}e^{(-xT)}$$

- Normalize the probability by the estimation:

$$P_{new}(\lambda = x) \leftarrow \frac{F(x)}{\sum_{i=1} F(i)}$$

Forecasting

- Sprout calculates a packet delivery forecast with a cautious estimate with error less than 5%. It sums over each λ to find the probability distribution of the cumulative number of packets that will have been drained by that point of time.
- Receiver will deliver this estimation to the sender. To help with the estimation, sender will provide required information (See sec3.4 in paper).
- Sender uses the most recent forecast to calculate a window size. It looks ahead five ticks and use the expected number of drained packets subtract the current queue occupancy estimate to obtain the "safe-to-send" bytes estimation.

Experimental Testbed

Goal: use trace-driven emulation to capture the variability of cellular network.

- Saturator: Use a laptop connected two phones, with one phone's link saturated and another phone returning feedbacks. Different types of android phones are used.
- Cellsim: Input saturated traces to an emulator. This link has a one-way delay for about 20ms (40ms RTT).
- SproutTunnel: Established over a UDP channel to carry arbitrary traffic across a cellular link. Seperates each flow into its own queue.

Evaluation

Compared against TCP Vegas, LEDBAT, and Cubic. Evaluates Sprout and Sprout-EWMA. Metrices:

- Throughput: total number of bits received by an application divided by duration of the experiment.
- self-inflicted delay: lower bound of end-to-end delay between sender and receiver (since sending the most recent packet)

TCP Congestion Control Beyond Bandwidth-Delay Product for Mobile Cellular Networks (PropRate)

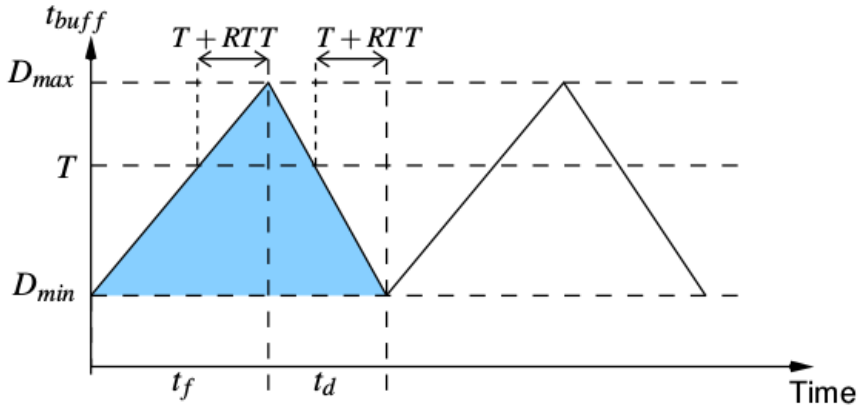


Figure 1: Buffer full (throughout optimal) case: buffer delay oscillates between D_{max} and D_{min}.

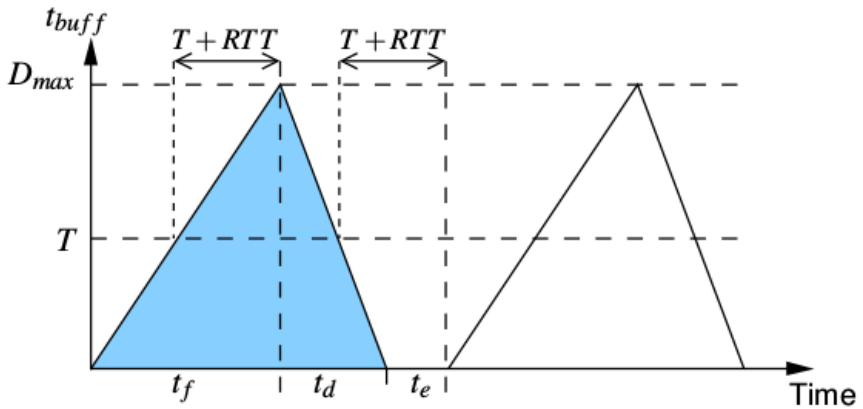


Figure 2: Buffer emptied (delay sensitive) case: buffer delay oscillates between D_{max} and 0.

Motivation

- Current cwnd-based congestion control mechanism intimately couples congestion control and packet dispatch, which provides TCP with only indirect control of effective data rate.
- Saturation of cwnd-based mechanism works well only for small buffers.
- Automatic retransmission mechanism and fair scheduling at LTE BSes are two important cellular network features.
- Idea: Directly regulate packets in the bottleneck buffer.

Features of PropRate

- Uses the buffer delay as the congestion signal, and performs rate control by having the sending rate oscillate around the estimated receive rate.
- Allows an application to set desired maximum latency.
- Uses one-way delay to do the estimation.
- Divided into two phases, buffer fill and buffer drain.

Variable Name	Meaning	Dependent on
$\bar{t}_{buff}$	User specified buffer delay	None
t_{buff}	Instantaneous buffer delay	Measurement
T	Threshold of switching states	$\bar{t}_{buff}$
k_f, k_d	Proportion to set sending rate	$\bar{t}_{buff}$
$\mathbb{U}$	Bottleneck-buffer utilization	T, k_f, k_d
D_{max}, D_{min}	Max/Min instant buffer delay	T, k_f, k_d

Big Picture

- User specifies the buffer delay $\bar{t_{buff}}$ and the algorithm tries to keep the periodic delay in $\bar{t_{buff}}$.
- Assume the estimated bottleneck bandwidth is ρ.
- When the instantaneous buffer delay t_{buff} is lower then threshold T, enter buffer fill state; otherwise enter buffer drain state.
- Buffer fill state sends a bit faster than ρ, and buffer drain state sends slower.
- Sending rate is adjusted to $\sigma_f = k_f \rho, \sigma_d = k_d \rho$.
- Utilization of the bandwidth is the time proportion when buffer is not empty.

$$U = \frac{t_f + t_d}{t_f + t_d + t_e}$$

Big Picture (Contd)

- It can be further calculated that

$$\bar{t_{buff}} = \begin{cases} \frac{D_{max}+D_{min}}{2} & \text{, buffer full case} \\ \frac{D_{max}}{2} U & \text{, buffer empty case} \end{cases}$$

- D_{max} and D_{min} can be determined by the buffer delay threshold T.
- k_f and k_d are slope of the $t_{buff} - time$ diagram.
- State change is delayed by one RTT. Thus the overhead is $T + RTT$

Optimization

Optimization for throughput (setting T and D_{min}):

- To minimize state change time (preventing the diagram from being skewed), set $T = \bar{t_{buff}}$.
- Set $D_{min} = \bar{t_{buff}}/2$ to fully utilize the link.

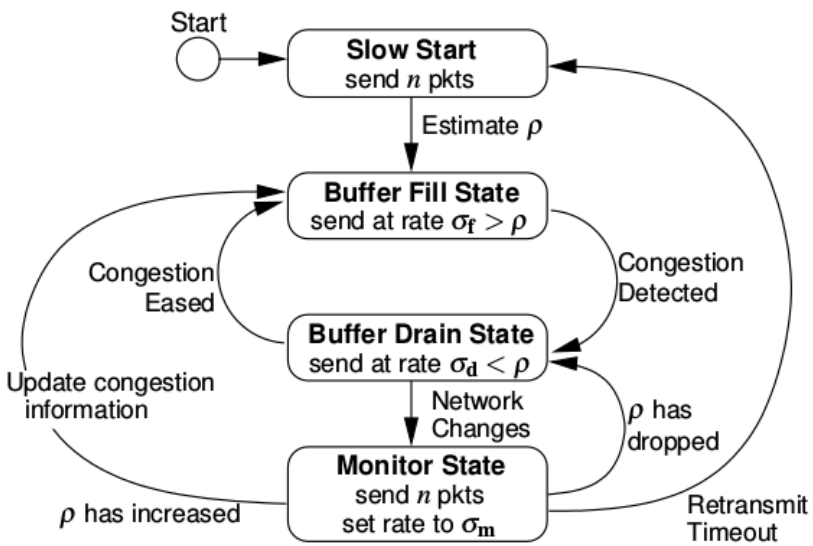
Optimization for latency (setting D_{max}):

- Transmission delay and queuing delay are considered.
- If an application specified the maximum allowed delay L_{max}, we should have $RTT + D_{max} \leq L_{max}$
- By experiment, let $D_{max} = U^3(L_{max} - RTT)$, where RTT is round trip time excluding buffer delay.

Then, $t_{buff} \leq L_{max} - RTT$, we can set all mentioned parameters accordingly. The threshold T can be adjusted with an exponentially weighted moving average empirically.

State Transition

- Slow start: send 10 packets to obtain ρ.
- Buffer fill and buffer drain state after slow start. Retransmission timeout returns to slow start state.
- If staying in buffer drain state for too long, go to monitor state to obtain the current receiving rate and check background link condition.



(b) Rate-based mechanism.

Implementation

Buffer regulation:

- Limit the number of packet allowed to be transmitted to $RTT * \rho$ before entering monitoring mode.
- t_{buff} is estimated by the difference of current observed one-way and the minimum one-way delay observed in the recent past.

Bandwidth estimation and rate control:

- Bandwidth is estimated at sender when TCP timestamp option is on.
- Server sends `TSval` back to the sender. Sender can estimate the bandwidth according to ACK number and difference between two `TSval` values.

Evaluation Testbed

- Cellsim is used to generate network traces.
- Collect two sets of traces: stationary and mobile. 3 ISPs are tested for each set.
- `tcpdump` is used to capture packet traces.
- Use `iperf` to generate traffics.
- Target average buffer delay t_{buff} is set to 20ms, 40ms, and 80ms.

Metrics and Results

- Compared against many different algorithms (10 algorithms).
- The throughput and latency trade-off is good in all 6 traces. However, PropRate does not respond well to network outage.
- Effectiveness of negative feedback loop (updating T): test with and without NFL.
- Uses practical LTE network to validate the effectiveness of Cellsim's emulation result.
- Tests contention against itself and CUBIC to validate TCP-friendliness.
- Tests computational overhead by measuring CPU utilization ratio.