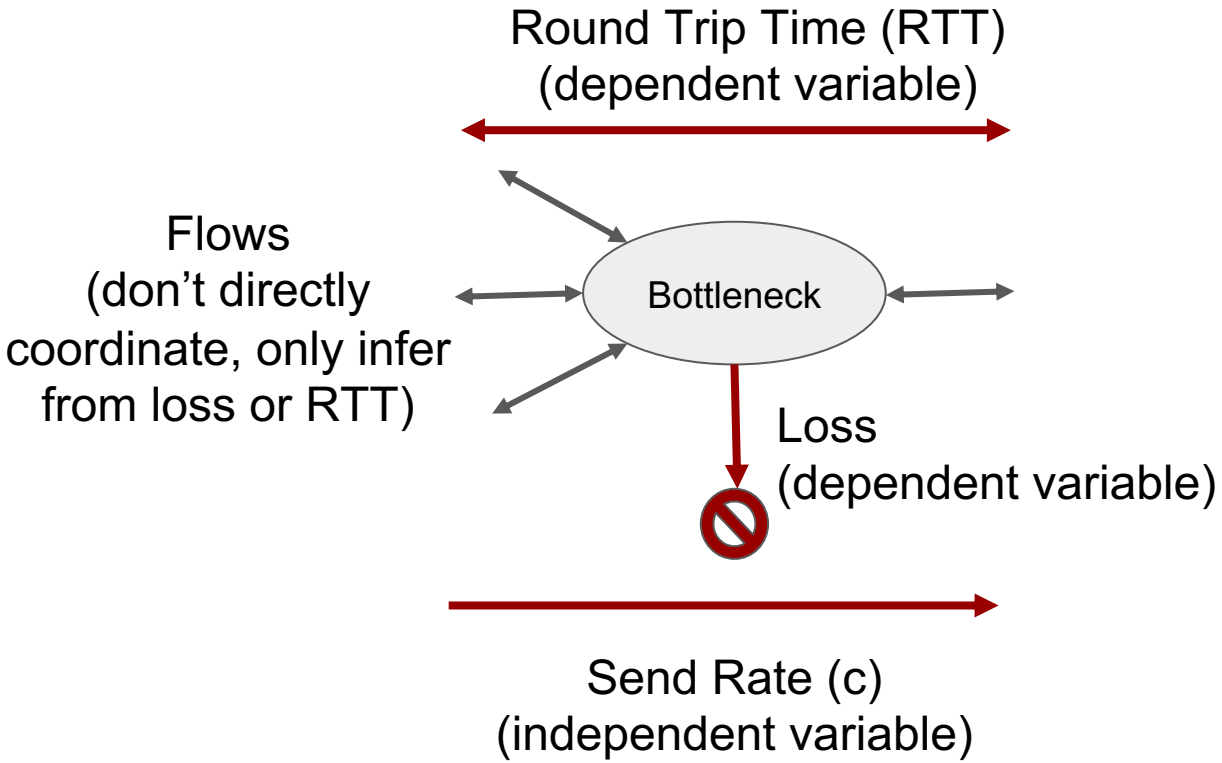
AI Applications in Network Congestion Control

Nathan Pemberton

Problem Statement

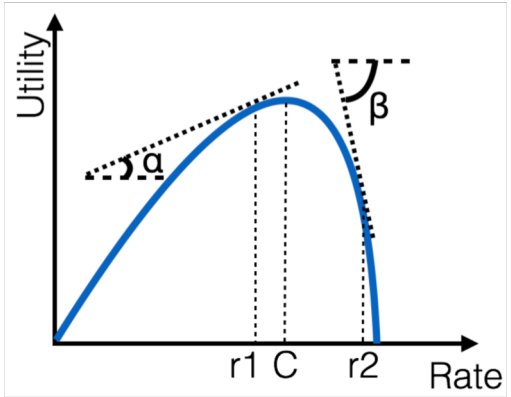


Main Objective:
Maximize send rate, minimize Loss and RTT

Congestion Control Other Desiderata

- **Fairness:** All flows should converge to the same properties
    - **Heterogeneity:** Should work even when other flows use different algorithm (especially TCP) or different goals (e.g. centrally controlled BW caps)
- **Regret-Free:** Adaptive schemes should be strictly better than a fixed policy
- **Optimality:** Each metric should be maximized at steady state (e.g. fair throughput should occur at min RTT)
- **Random Loss Tolerance:** Not all packet loss is due to send rate, algorithms must tolerate random packet loss as well
- **Dynamics:** Algorithms must react quickly to changes in NW, both short term (noise) and long term (new flows, mobile, etc.)

Background PCC Vivace



$$u\left(x_i, \frac{d(RTT_i)}{dT}, L_i\right) = x_i^t - bx_i\frac{d(RTT_i)}{dT} - cx_i \times L_i$$

Online Gradient Ascent
(adaptive learning rate)

Send Rate

Latency Gradient

Packet Loss

Maximize

Minimize

Vivace Strengths and Limitations

Strengths

- Provable Properties:
    - Formal bounds on parameters give proof of all desiderata (not all mutually achievable)
- Excellent Performance
    - Rapidly adapts to many situations
    - Best-in-class or better in most metrics, but much more stable and less fiddly (especially vs TCP)
    - Easily tuned (parameters mean something, and have associated proofs)

Limitations

- Slow adaptation in extreme environments (e.g. mobile)
- Random loss tolerance => higher congestion loss
    - Paper claims this is fundamental
- Conflicts with TCP: fundamental tradeoff between loss-based and latency-based algorithms

General E2E RL Approach

Actions: Change Sending Rate

State: Past N time windows (rate, latency, loss)

Rewards: $$R_{power} = \frac{throughput}{latency}$$

$$R_{V-loss} = throughput^{0.9} - 11.35 * throughput * loss$$

(Didn't use Allegro or Vivace utility for some reason?)

Model: Deep NN for policy

Training: Simple network simulator with single flow, vary only in latency and capacity

Results - RL Approach

“Our evaluation showed that Custard was fairly robust with respect to link capacity, latency and buffer size”

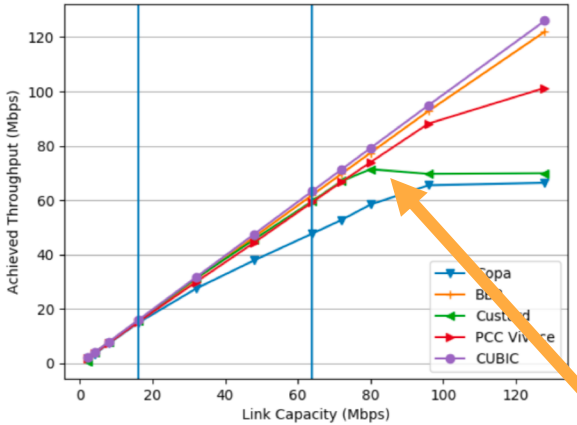
“Our model may be resilient to changes in link parameters, but it can suffer significantly from even minor changes in the environment”

“our model achieves near-capacity throughput with low self-inflicted latency”

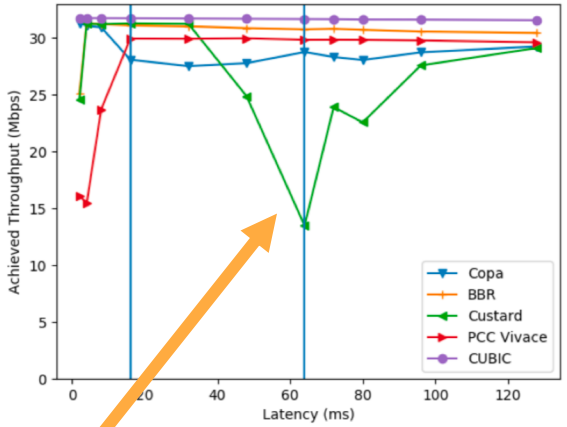
“Improving our agent’s robustness to multiflow competition is an interesting avenue for future work”

Results - RL Approach

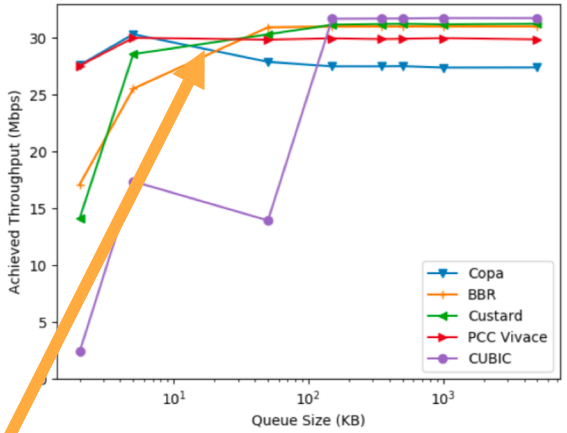
Capacity vs Throughput



Latency vs Throughput



Queue Size vs Throughput

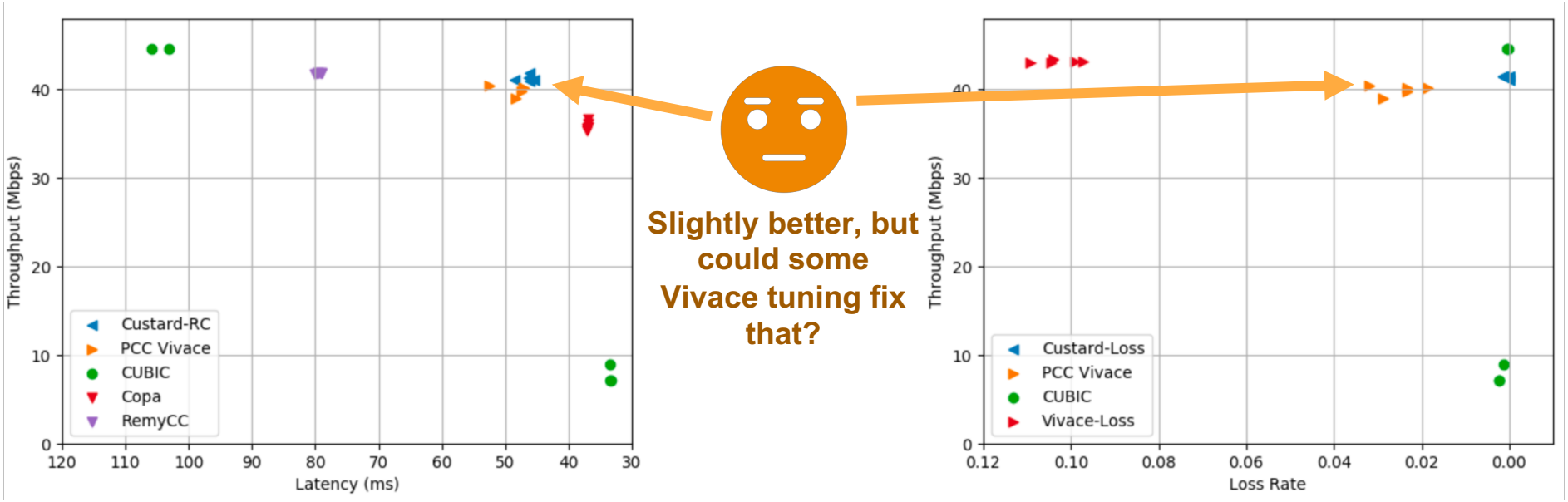


Bad training data

Regularization?

Somebody grab the champagne!

Results - RL Approach on dynamic links



Discussion

- Why were the RL results so mediocre?
- Better uses for ML techniques here?
    - Vivace is very good, has lots of parameters. Can we try to learn these parameters, maybe online?
- How significant are the different metrics (throughput, latency, loss)? What is the impact of these numbers?