PCC Vivace: Online-Learning Congestion Control

Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, P. Brighten Godfrey and Michael Schapira

Why replace TCP

- TCP performs poorly on lossy links, penalizes high-RTT flows, underutilizes high bandwidth-delay product (BDP) connections
- TCP's rate control architecture made hardwired mapping: certain predefined packet-level events are hardwired to certain predefined control responses.
- TCP assumes that the loss indicates congestion in the network
  - -> a packet loss halves the congestion window.

Performance-oriented Congestion Control (PCC)

- Goal: understanding what rate control actions improve performance based on live experimental evidence, avoiding TCP's assumptions about the network

- Sends at a rate r for a short period of time
- Observes the results
  - e.g. SACKs indicating delivery, loss, and latency of each packet
- Maximize the utility function
  - Achieving the objective like "high throughput and low loss rate"

Original PCC

- Utility function
  - $$u_i(x_i) = T_i \cdot Sigmoid_\alpha(L_i - 0.05) - x_i \cdot L_i$$
  - x_i: rate, T_i: throughput, L_i: Loss rate
- Starting: Doubles its rate at each monitor interval (MI)
  - Until utility decreases
- Decision Making State: Try $(1+\varepsilon)x$ and $(1-\varepsilon)x$ each for 2 MI
  - Choose a changing direction
- Rate Adjusting State:
  - Keep increase or decrease rate $x \leftarrow (1+\varepsilon)x$ or $x \leftarrow (1-\varepsilon)x$ until utility decreases
  - Then enter Decision Making State again

PCC Vivace

- Utility function considering change of RTT
  - $$u\left(x_i, \frac{d(RTT_i)}{dT}, L_i\right) = x_i^t - bx_i\frac{d(RTT_i)}{dT} - cx_i \times L_i,$$
  - x_i: rate, L_i: Loss rate
  - t, b, c: pre-defined parameters.
- A stable global rate configuration (Nash equilibrium) always exists for homogeneous agents
- “no-regret” guarantee: No worse than the best fixed rate policy
- Uses gradient-ascent algorithms for quick convergence

Fairness

- Regarded as a game, this problem is “socially-concave”

- Theoretically, consider a network model with *n* senders competing on a bottleneck link with a FIFO queue

Theorem 1. *When n Vivace-senders share a bottleneck link, and each Vivace-sender i's utility function is defined as in Eq. 1, the senders' sending rates converge to a fixed configuration* $(x_1^*, \ldots, x_n^*)$ *such that* $x_1^* = x_2^* = \ldots = x_n^*$.

Rate-control algorithm

- Start state:
  - Double sending rate every MI
  - Exit when utility drops
- Online Learning State:
  - Compute gradient
    - In the next two MIs, try $(1+\varepsilon)r$ and $(1-\varepsilon)r$, get two utility u_1, u_2. $\gamma = \frac{u_1 - u_2}{2\,\varepsilon\, r}$
  - Choose step size
    - Monotonically non-decreasing function $m(\tau)$ for integer τ. $m(0) = 1$
    - $\Delta r = m(\tau)\theta_0\gamma$, for τ consecutive increases or consecutive decreases
  - Clip the gradient, *dynamic change boundary*
    - $\Delta r \leftarrow \min\{\omega r, \Delta r\}$
    - $\omega = \omega_0 + k \cdot \delta$ for k consecutive changes exceeding ωr

Contending with Unreliable Statistics

- Estimate the RTT gradient via linear regression
  - Use the linear-regression-generated slope as the RTT gradient
- Low-pass filtering of RTT gradient
  - Ignore small RTT gradient to ignore Non-congestion- induced latency jitters
- Double checking abnormal measurements
  - Sending faster would not lead to lower loss
  - Thus re-calculate the gradient.
- MI timeout
  - Network condition changes may cause large number of lost packets.
  - Halves the rate when no packets can provide information in $T_{timeout}$

Implementation, Evaluation

Vivace Parameters

MI duration	1 RTT
sampling step ε	0.05
initial conversion factor θ_0	1
initial dynamic boundary ω_0	0.05
dynamic boundary increment δ	0.1
RTT gradient filter threshold $flt_{latency}$	0.01
MI timeout $T_{timeout}$	4 RTT
confidence amplifier $m(\tau)$	τ $(\tau \leq 3)$ $2\tau - 3$ $(\tau > 3)$

Table 1: Vivace's rate control default parameters

Consistent High Performance

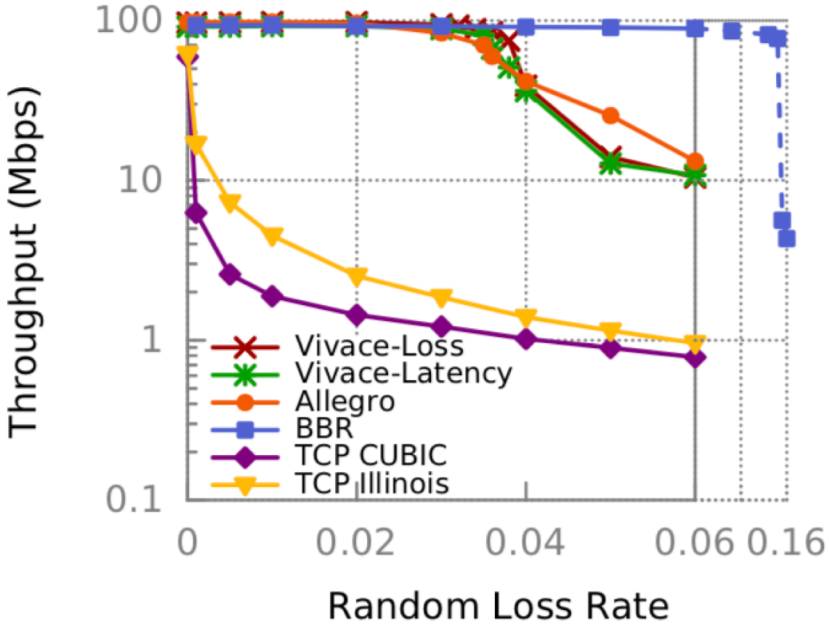


Figure 2: Random loss resilience

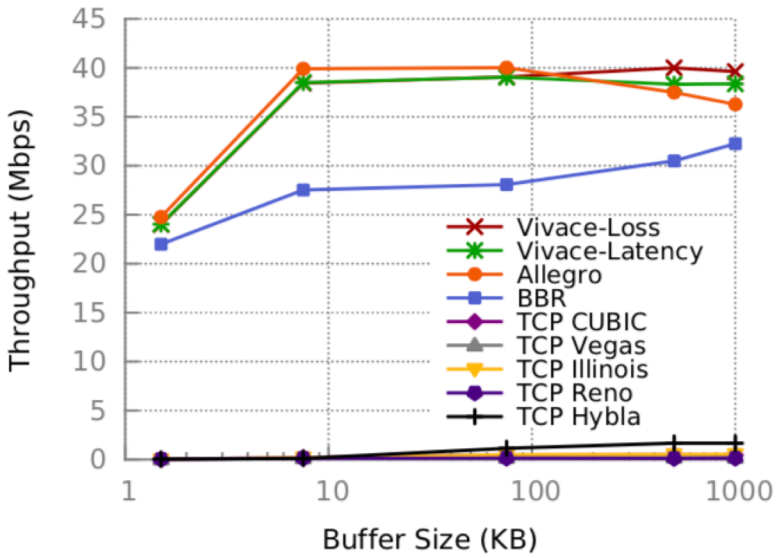
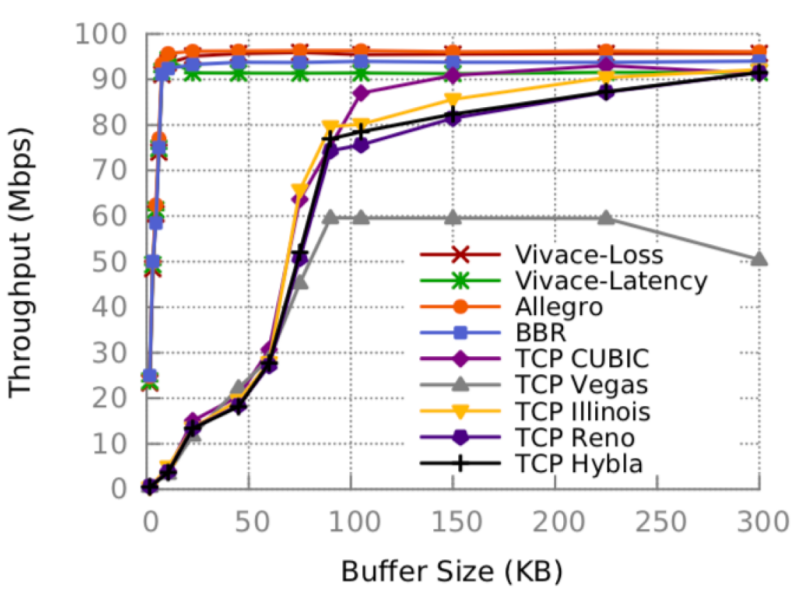
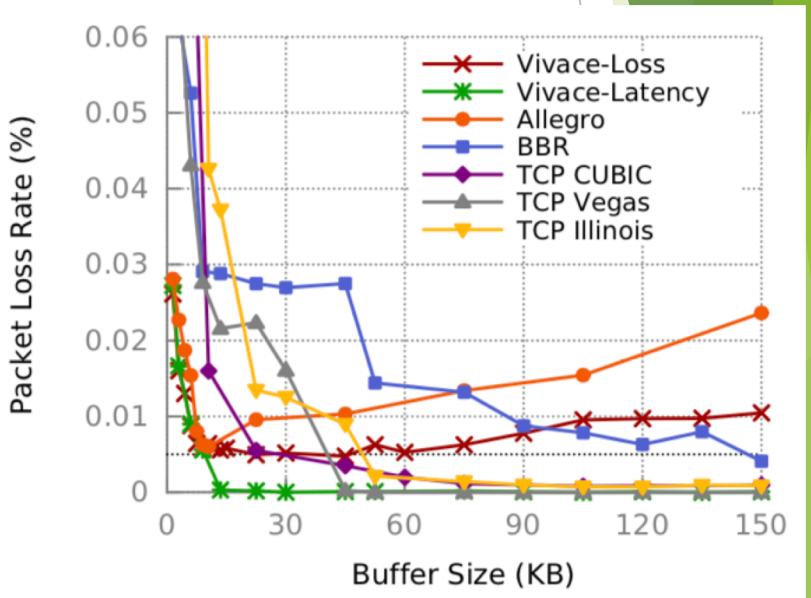


Figure 3: Long RTT tolerance

Influence of Buffer Size

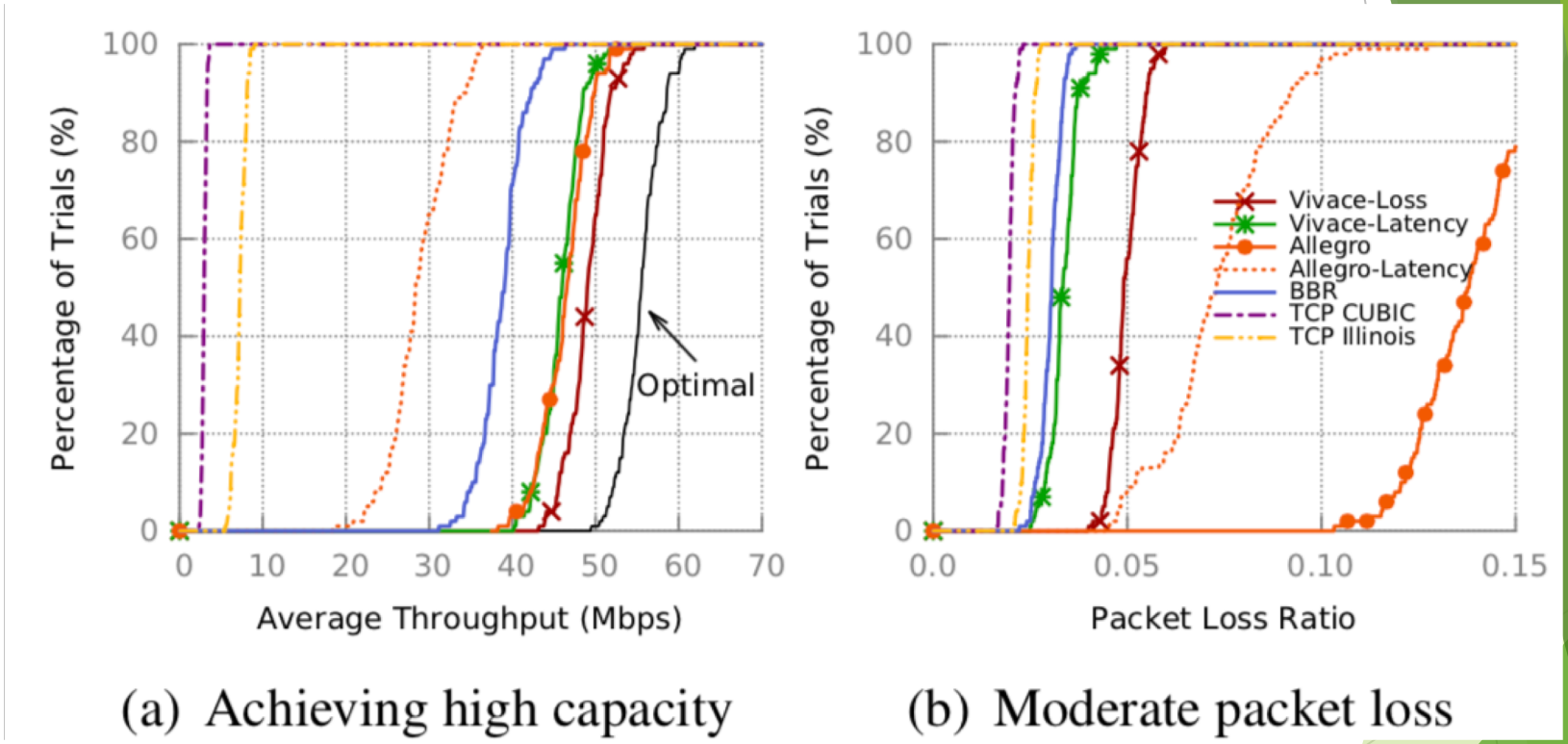


(a) High capacity



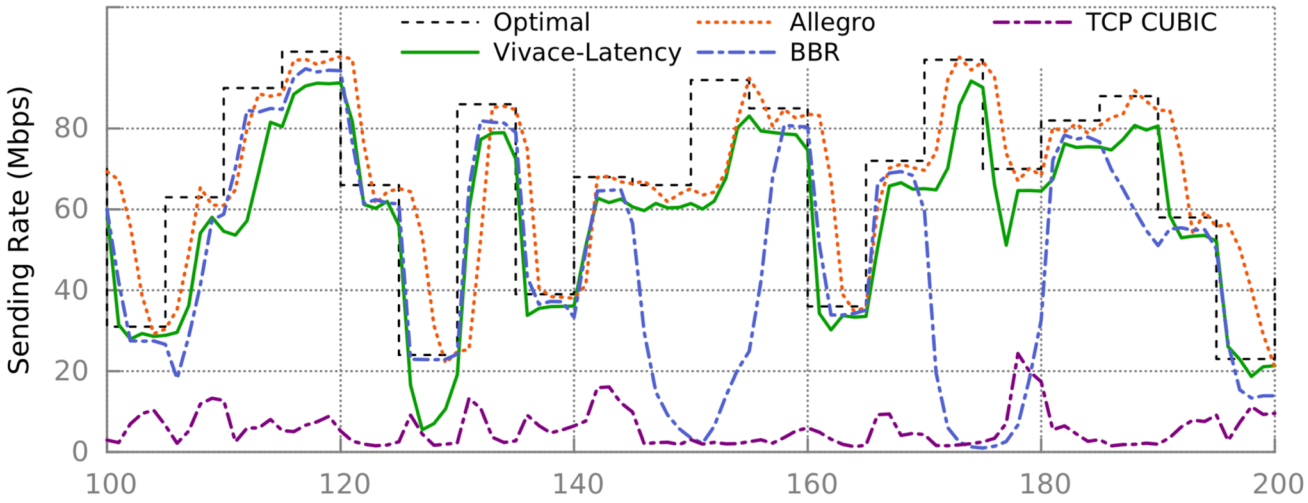
(b) Low packet losses

In rapid changing network condition

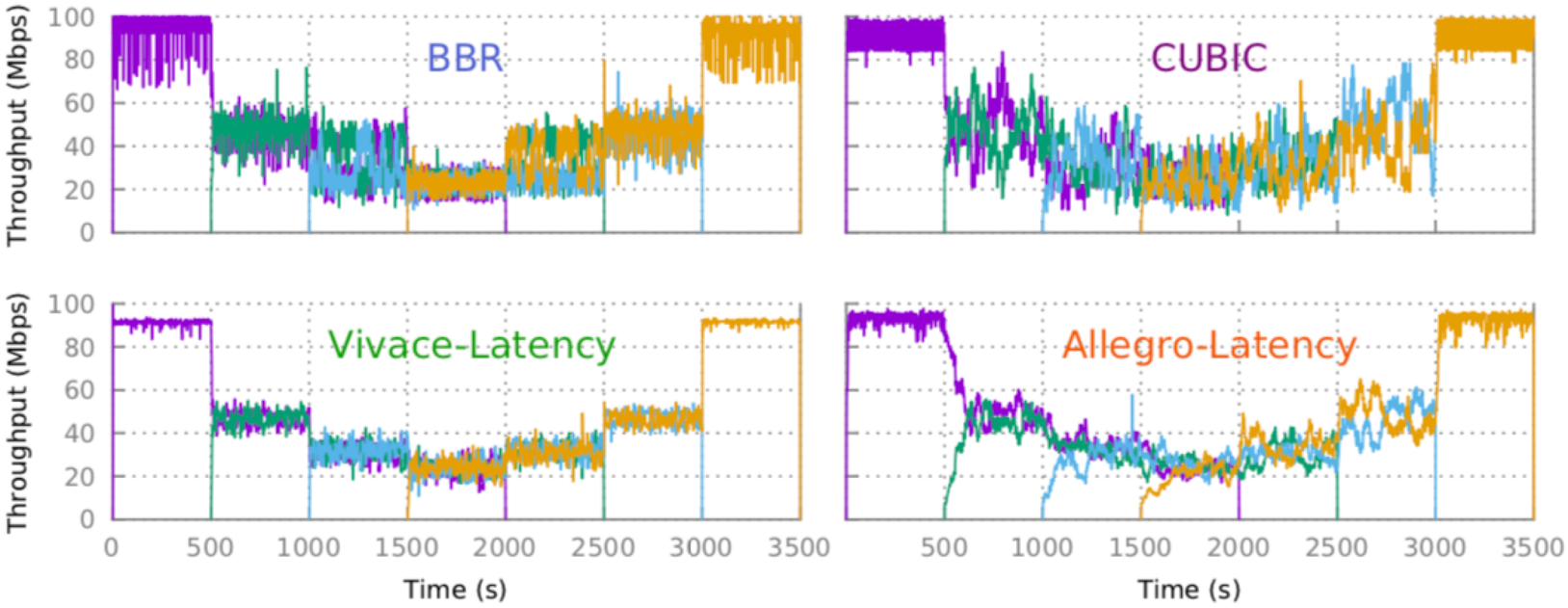


RTT, bottleneck bandwidth, and random loss rate all change every 5 seconds with uniform distribution ranging from 10-100 ms, 10-100 Mbps, and 0-1%

Cont’d

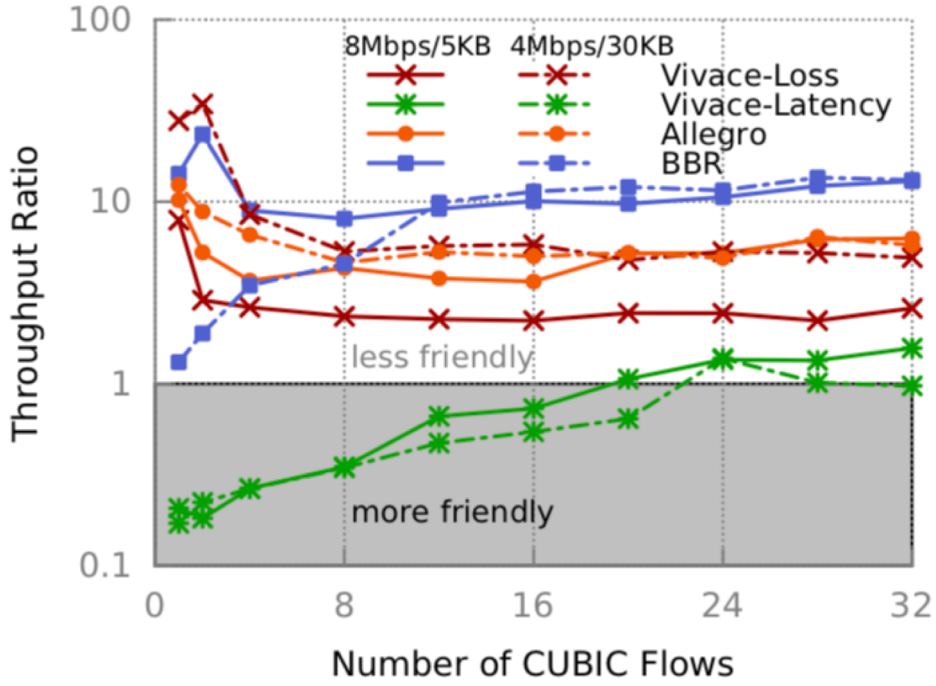

Optimal
Allegro
TCP CUBIC
Vivace-Latency
BBR
Sending Rate (Mbps)
0
20
40
60
80
100
120
140
160
180
200

Convergence Properties



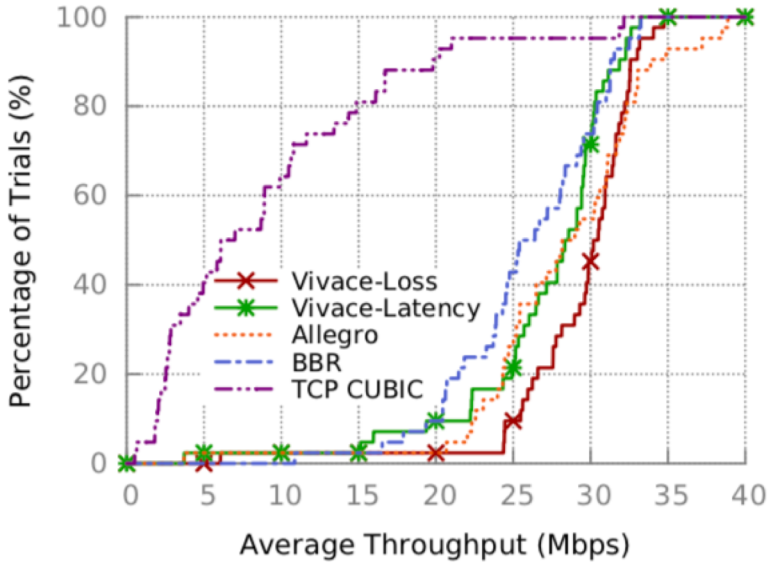
4 flows sharing a link with 100 Mbps bandwidth, 30 ms RTT, and 75 KB buffer

Improved Friendliness to TCP



Increase number of CUBIC flows, compare the ratio between the throughput of the new-protocol flow and average throughput per CUBIC flow

Performance in real world



(c) Throughput from home networks to AWS

3 senders in different Wi-Fi network, 14 receivers in AWS

Summary

- A PCC protocol considering latency
- Gradient-ascent-based online learning rate control to achieve provably fast convergence and fairness guarantees.
- Sufficient Experiments

- Online Learning can be powerful when handling production environment.
- There can be lots of implementation details that need consideration when implementing algorithms.