Improving TCP Congestion Control with Machine Intelligence

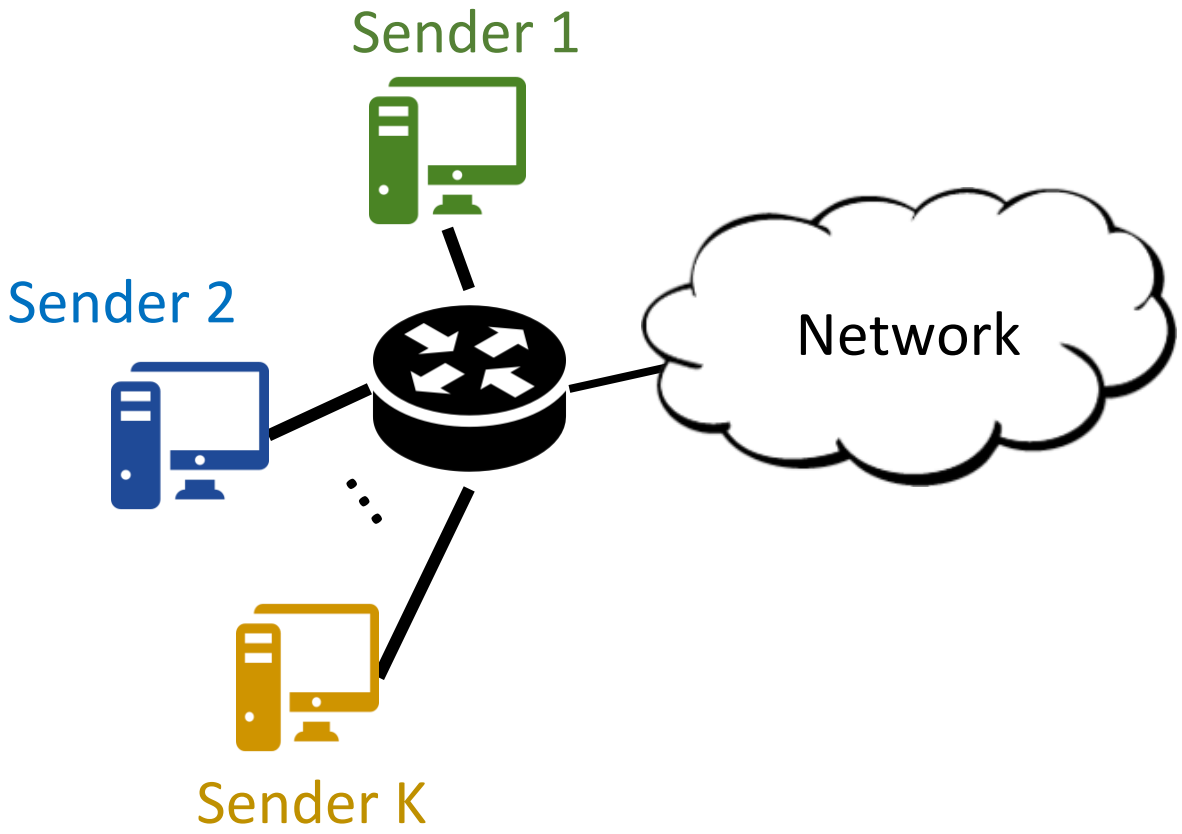
Yiming Kong*, Hui Zang†, and Xiaoli Ma*

*School of ECE, Georgia Tech, USA

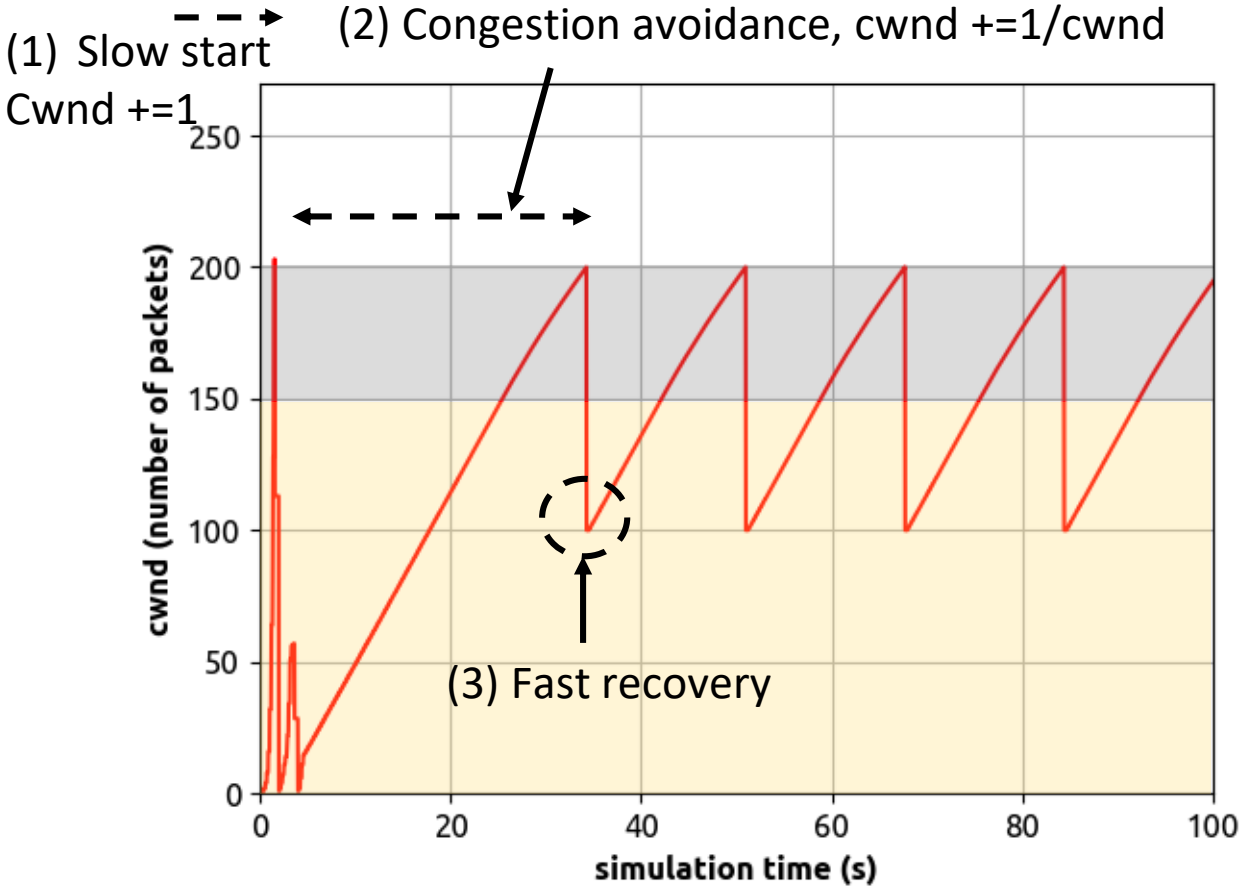
†Futurewei Technologies, USA

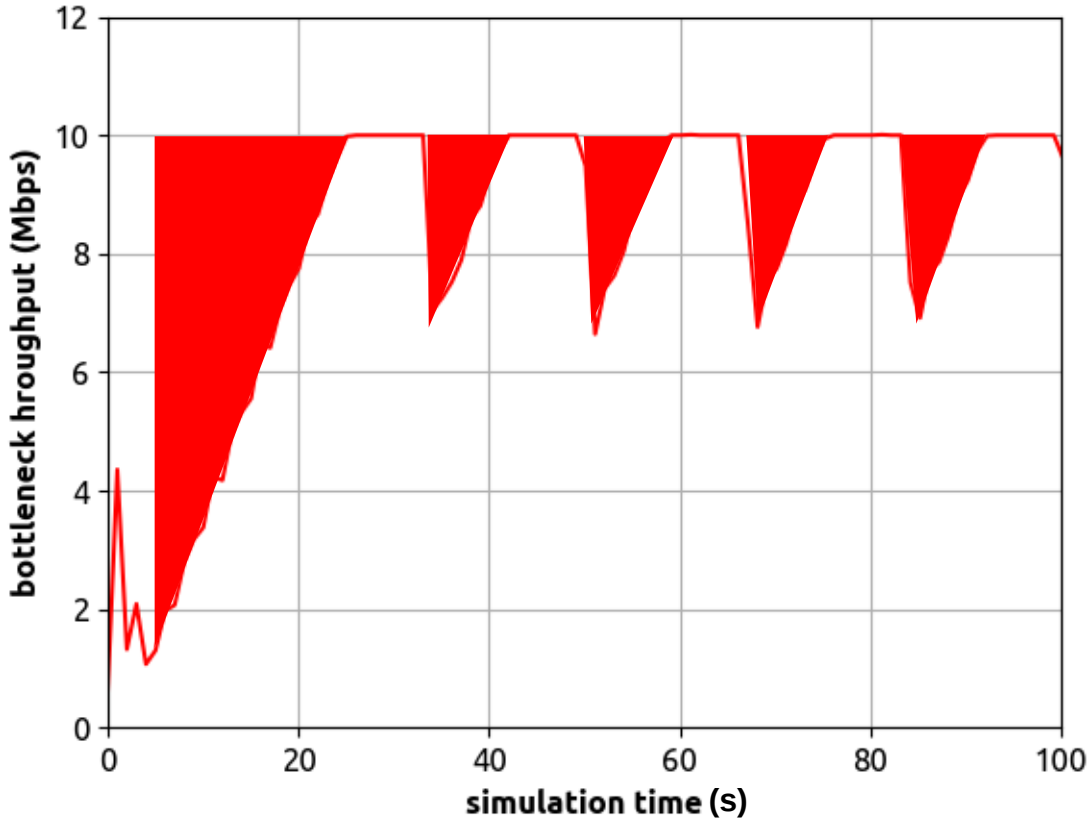
TCP congestion control

- A critical problem in TCP/IP networks
- End-to-end or with in-net support
- Adjust congestion window (cwnd)
  - Packet loss
  - Round trip time (RTT)
  - High throughput, low delay



TCP NewReno

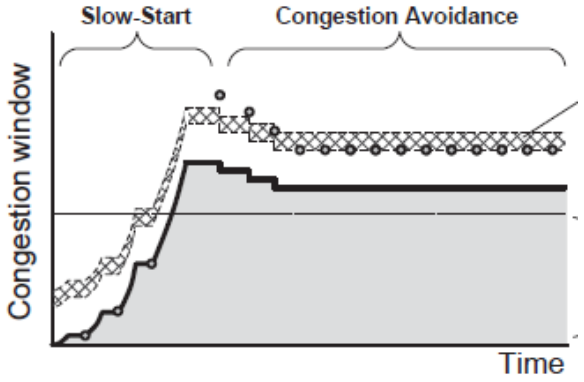




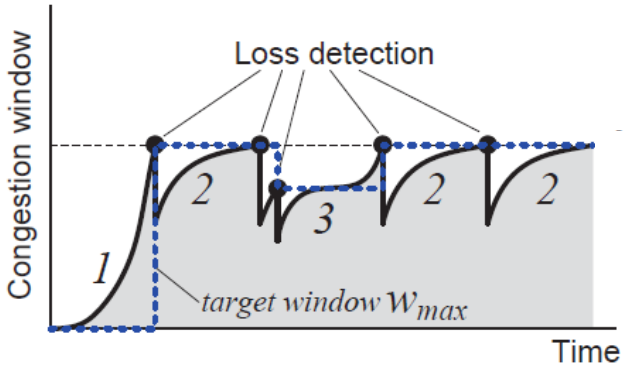
BW = 10Mbps, RTT_{min} = 150ms, Single NewReno flow

Other TCP congestion control schemes

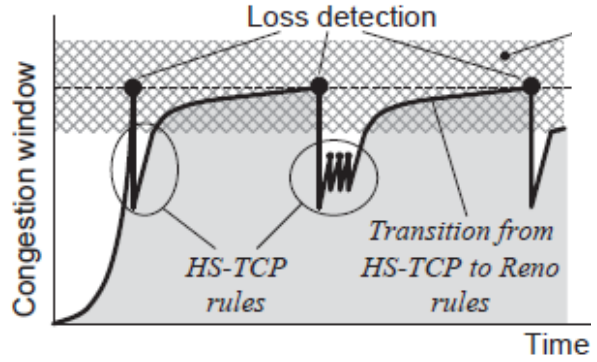
Vegas
[Brakmo et al. 1995]



Cubic
[Ha et al. 2008]



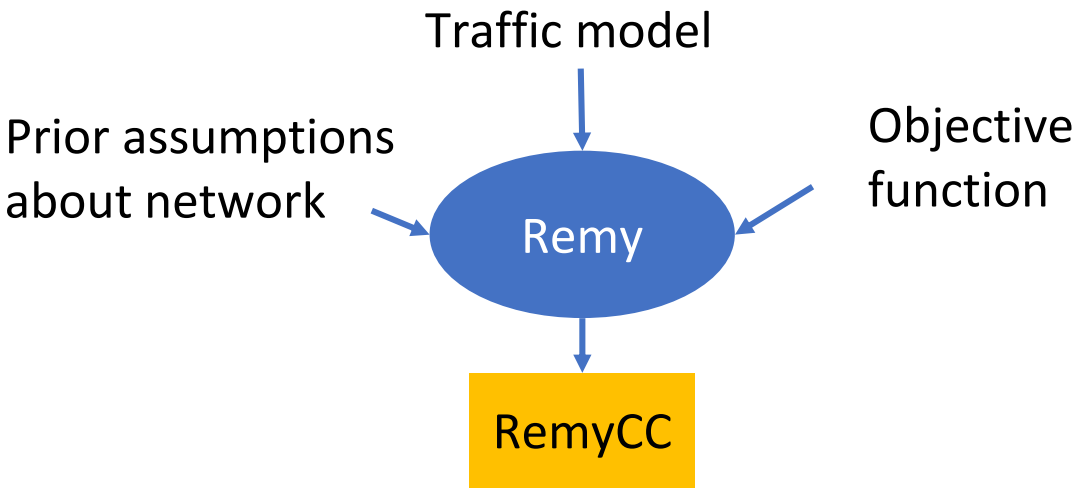
Compound
[Tan et al. 2006]



- Mechanism-driven instead of objective-driven
- Pre-defined operations in response to specific feedback signals
- Do not learn and adapt from experience

*Figures from: Afanasyev et al. 2010. Host-to-Host Congestion Control for TCP. IEEE Commun. Surveys Tuts., Vol. 12, No. 3, 304–342.

RemyCC [Winstein et al. 2013]



- Delay-throughput tradeoff as objective function
- Offline training to generate lookup tables
- Inflexible for the network & traffic model changes

Our contributions

- Teach TCP to optimize its cwnd to minimize packet loss events
  - LP-TCP
- Teach TCP to adaptively adjust cwnd according to an objective
  - RL-TCP

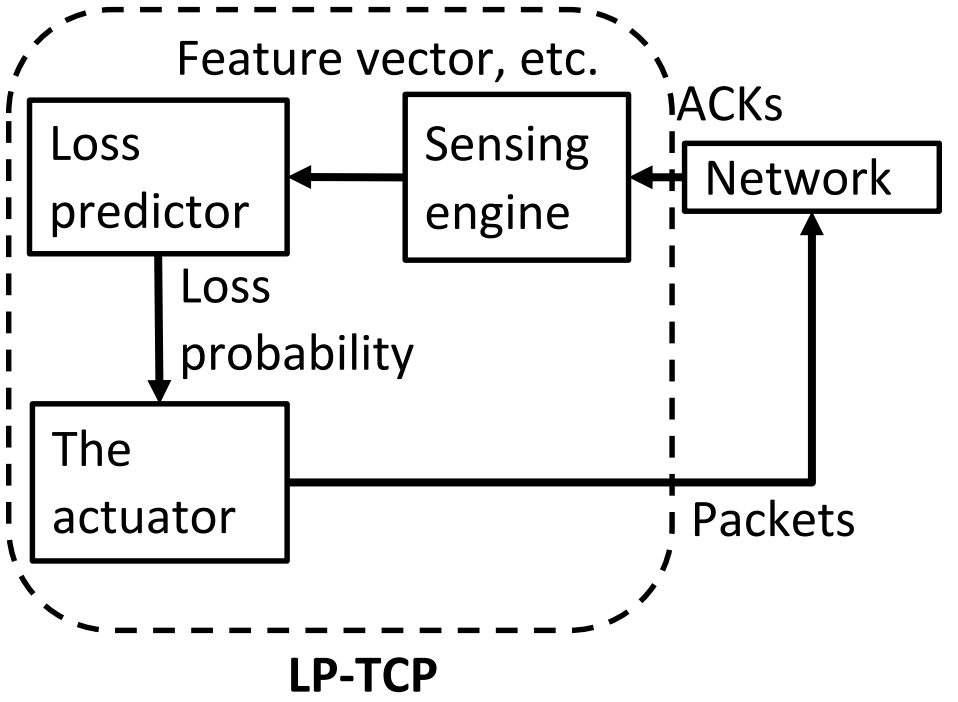
- Improved throughput -- up to 29% over NewReno for LP-TCP
- Reduced RTT -- up to 7% over NewReno for RL-TCP
- Maintaining fairness

Loss prediction based TCP (LP-TCP)

(during congestion avoidance)

- When a new ACK is received, cwnd += 1/cwnd
- Before sending a packet
  - Sensing engine updates the **feature vector**
  - Loss predictor outputs loss probability **p**
  - If p < threshold, the actuator **sends the packet**
  - Otherwise, the packet is not sent, and cwnd -= 1
- Set threshold to max

$$M_e = \log(throughput) - 0.1\log(delay),$$
$$where\ delay = RTT\ - RTTmin$$



Training the loss predictor

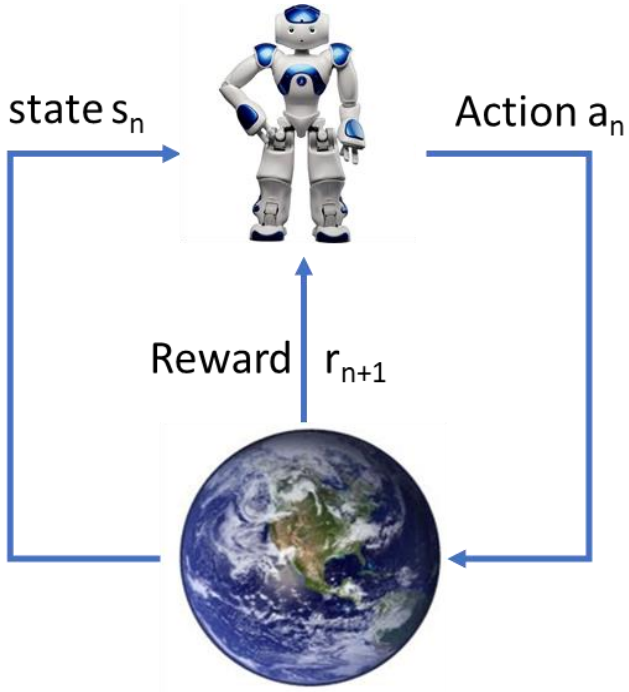
- **Collect training data** through NewReno simulations on NS2
  - Record the state right before the packet goes into transmission as a feature vector
  - If the packet is successfully delivered, this feature vector gets a label of 0
  - Otherwise, the label is 1 (for loss)
  - Stop the collection when we have enough losses in the data
- **Train a random forest classifier offline**

  Features
  cwnd, ewma of ACK intervals, ewma of sending intervals, minimum of sending intervals, minimum of ACK intervals, minimum of RTT, time series (TS) of ack intervals, TS of sending intervals, TS of RTT ratios, and etc.

- Re-train LP upon network changes

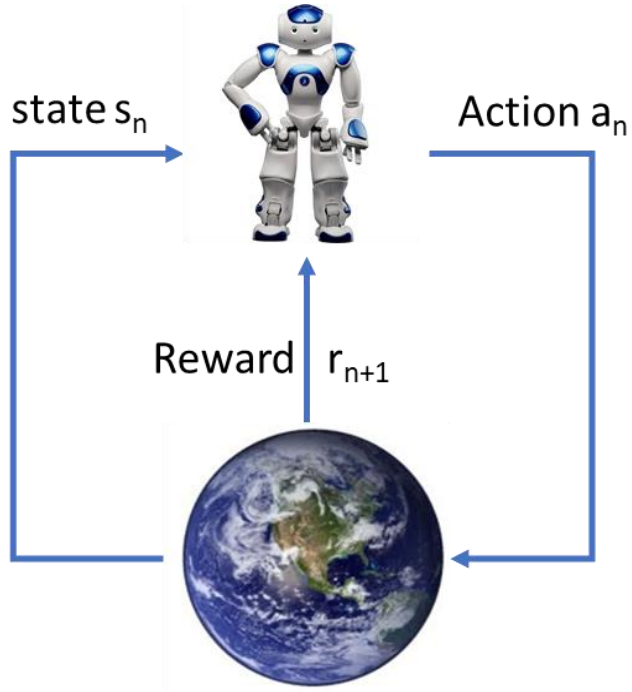
Reinforcement learning based TCP (RL-TCP)

- Q-TCP [Li et al. 2016]
  - Based on Q-learning
  - Designed with mostly a single flow in mind
  - Sufficient buffering available at the bottleneck



Reinforcement learning based TCP (RL-TCP)

Our RL-TCP
- Add variable to state
- Tailor action space to under-buffered bottleneck
- Propose a new temporal credit assignment of reward



- Objective of RL-TCP
  - Learn to adjust cwnd to increase an utility function

$$U = \log\left(\frac{tp}{B}\right) - \delta_1 \log(d) + \delta_2 \log(1 - p)$$

Bottleneck bandwidth — throughput — delay — Packet loss rate

Map to RL

- State s_n
  - ewma of the ACK inter-arrival time
  - ewma of packet inter-sending time
  - RTT ratio
  - slow start threshold
  - current cwnd size
- Action a_n
  - cwnd += a_n, where a_n = -1, 0, +1, +3

- Reward r_{n+1}

$$r_{n+1} = \begin{cases} 10, & \Delta_{n+1} \geq 1 \\ 2, & 0 \leq \Delta_{n+1} < 1 \\ -2, & -1 \leq \Delta_{n+1} < 0 \\ -10, & \Delta_{n+1} < -1. \end{cases}$$

where $\Delta_{n+1} = U_{n+1} - U_n$

Learning the Q-value

- Learning the Q-value function: Q(s, a)
  - Q(s,a): the value of being at a particular state *s* and performing action *a*
  - Updated every RTT, using SARSA

$r_{n+1} = f(U_{n+1} - U_n)$

$$Q(s_{n-1}, a_{n-1}) \xleftarrow{\alpha_t} r_n + \gamma Q(s_n, a_n),$$

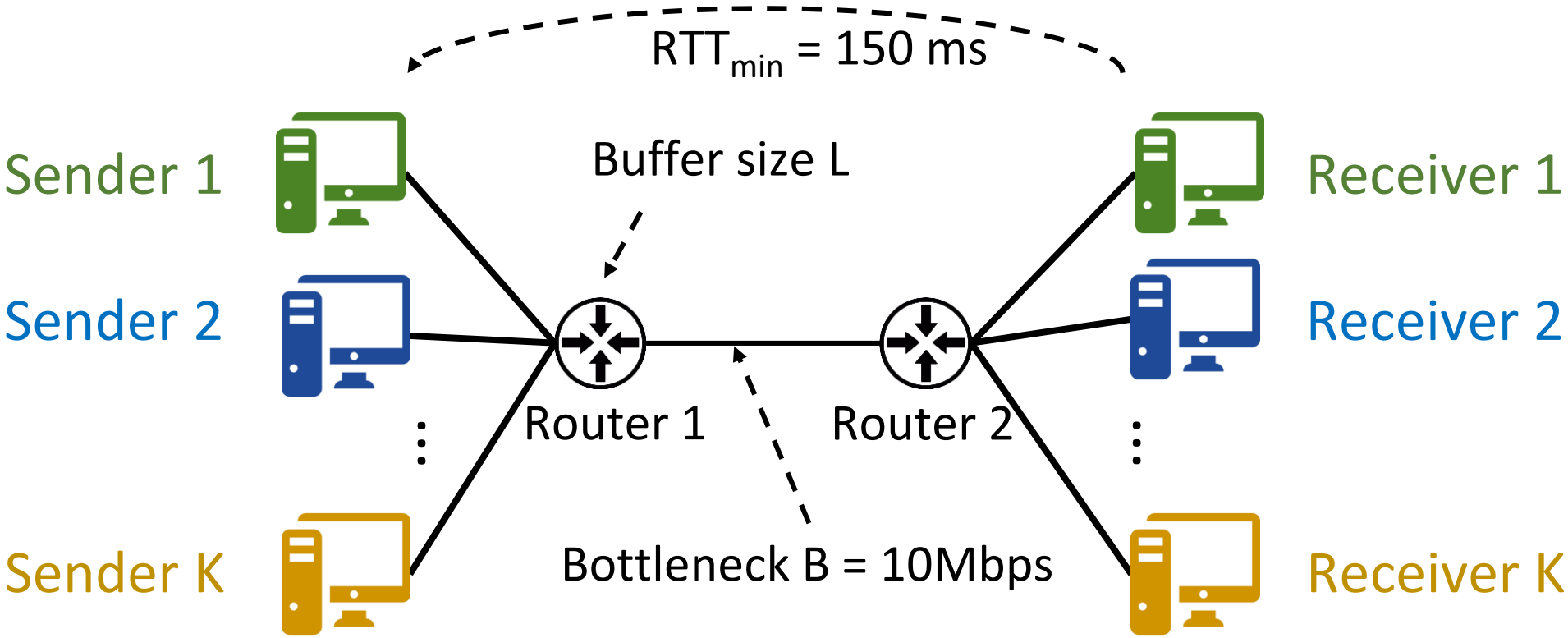
$$\text{where } y_1 \xleftarrow{\alpha} y_2 \text{ means } y_1 = (1 - \alpha)y_1 + \alpha y_2$$

  - This is the proposed temporal credit assignment of reward

- Action selection: ε-greedy exploration & exploitation

$$a_{n+1} = \begin{cases} \text{Randomly select an action from the action space, if rand() } < \varepsilon \\ \arg\max_{a \in \mathcal{A}} Q(s_{n+1}, a) \text{ , otherwise} \end{cases}$$

Experimental setup in NS2



- Bandwidth delay product = 150 packets
- Throughput (tp) = (total amount of bytes received)/(sender's active duration)
- Delay (d) = RTT - RTT_{min}

Single sender: performance of RL based TCP

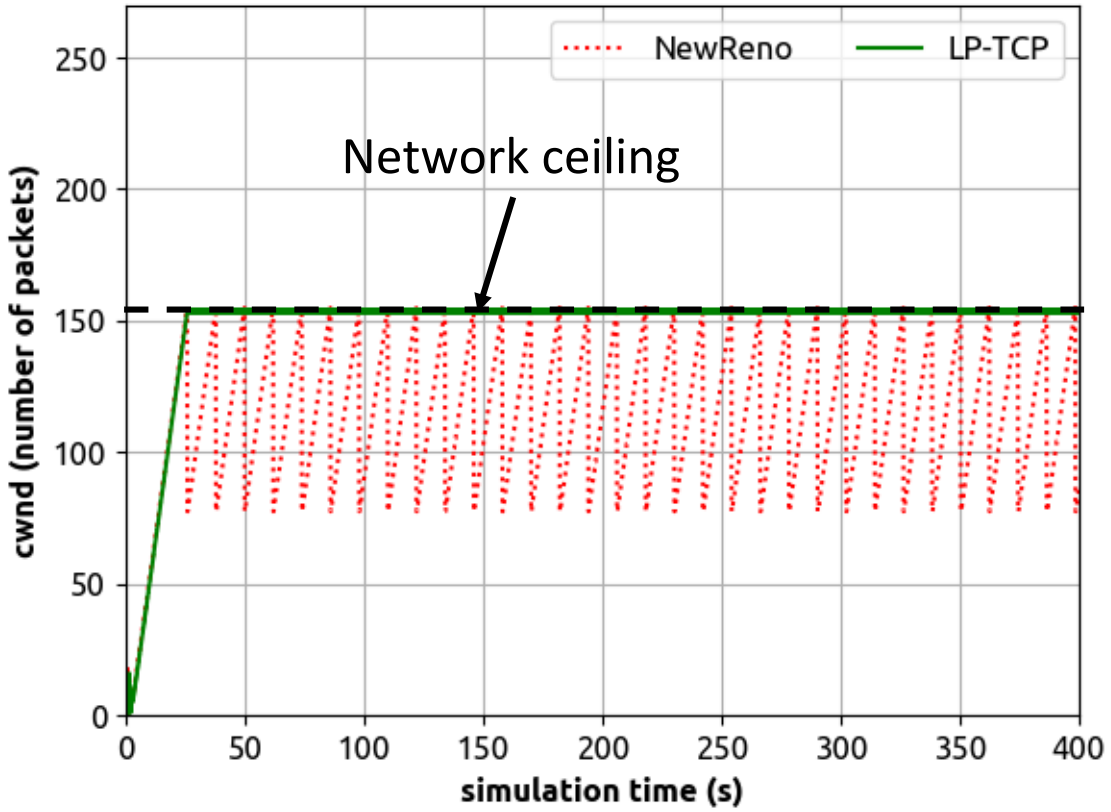
Table: Performance of RL based TCP. Buffer size L is 50 packets.

	E(tp)	V(tp)	E(d)	V(d)	M_e
Q-TCP	6.176	0.267	16.26	4.662	1.541
Q-TCP$_{ca}$	9.597	$8.72*10^{-3}$	20.31	3.690	1.960
Q_a-TCP	9.658	0.019	14.80	2.818	1.998
Q_a-TCP$_{ca}$	9.857	$8.10*10^{-5}$	3.74	$3.24*10^{-2}$	2.156
RL-TCP$_{no\text{-}ca}$	9.723	$9.30*10^{-3}$	13.87	3.152	2.011
RL-TCP	9.869	$7.49*10^{-4}$	3.86	$3.24*10^{-2}$	2.154

- Action space: Redesigning action space improves performance
- Credit assignment: The proposed credit assignment scheme improves performance

Single sender: performance of LP-TCP

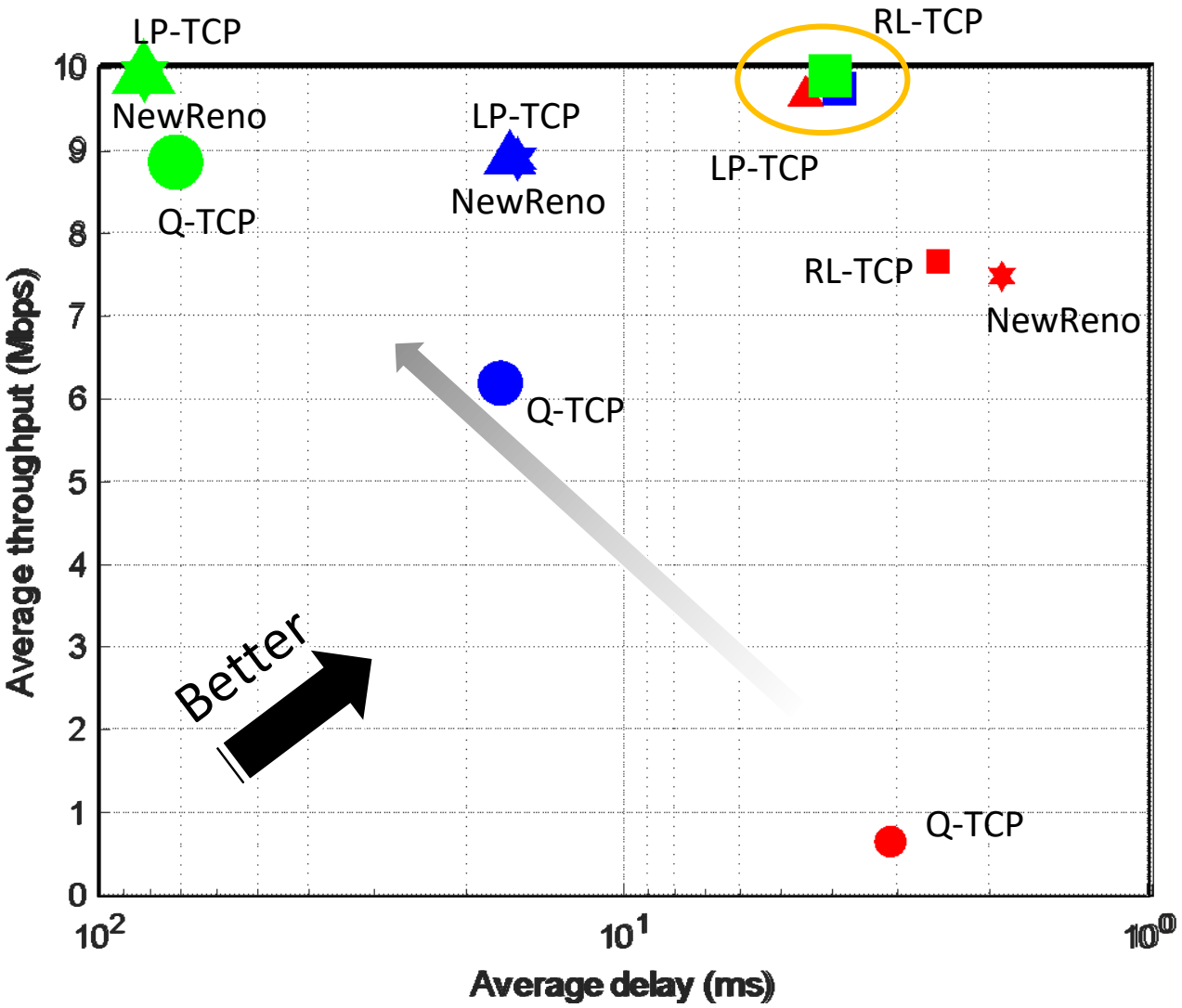
- Buffer size L = 5



- LP-TCP predicts all packet losses (during congestion avoidance) & keeps the cwnd at the network ceiling

Single sender: varying buffer size

L = 5
L = 50
L = 150

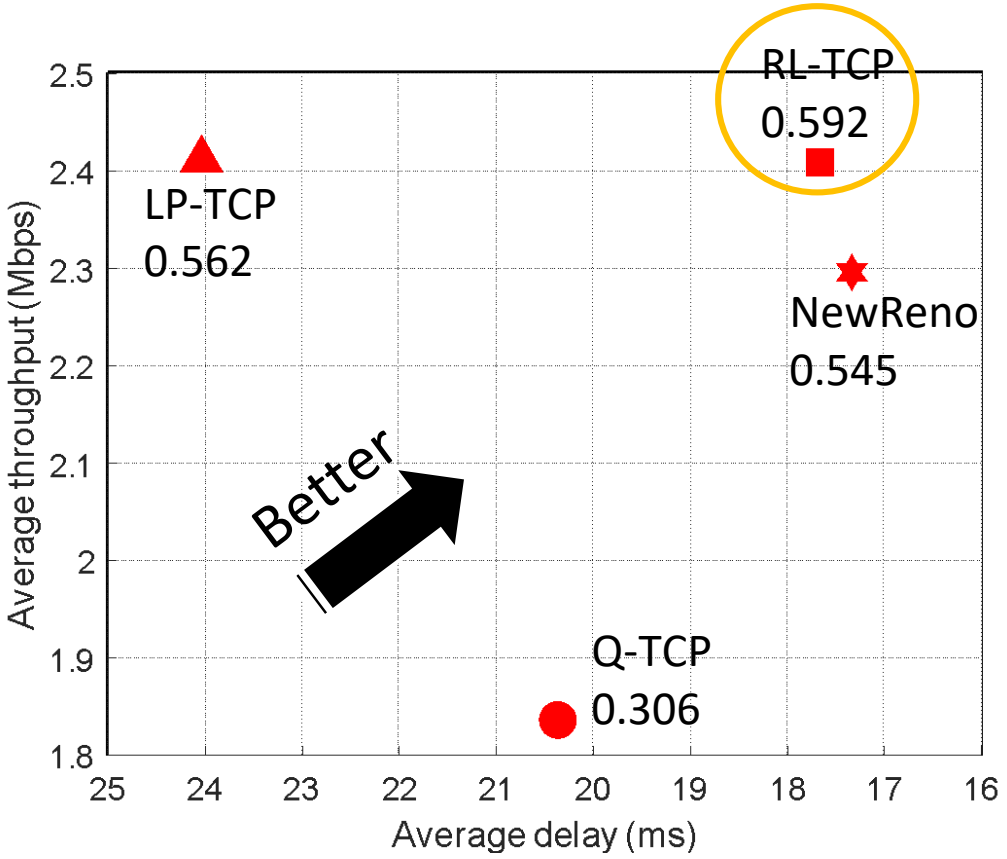


- LP-TCP has the best M_e when L = 5
- RL-TCP has the best M_e when L = 50, 150.

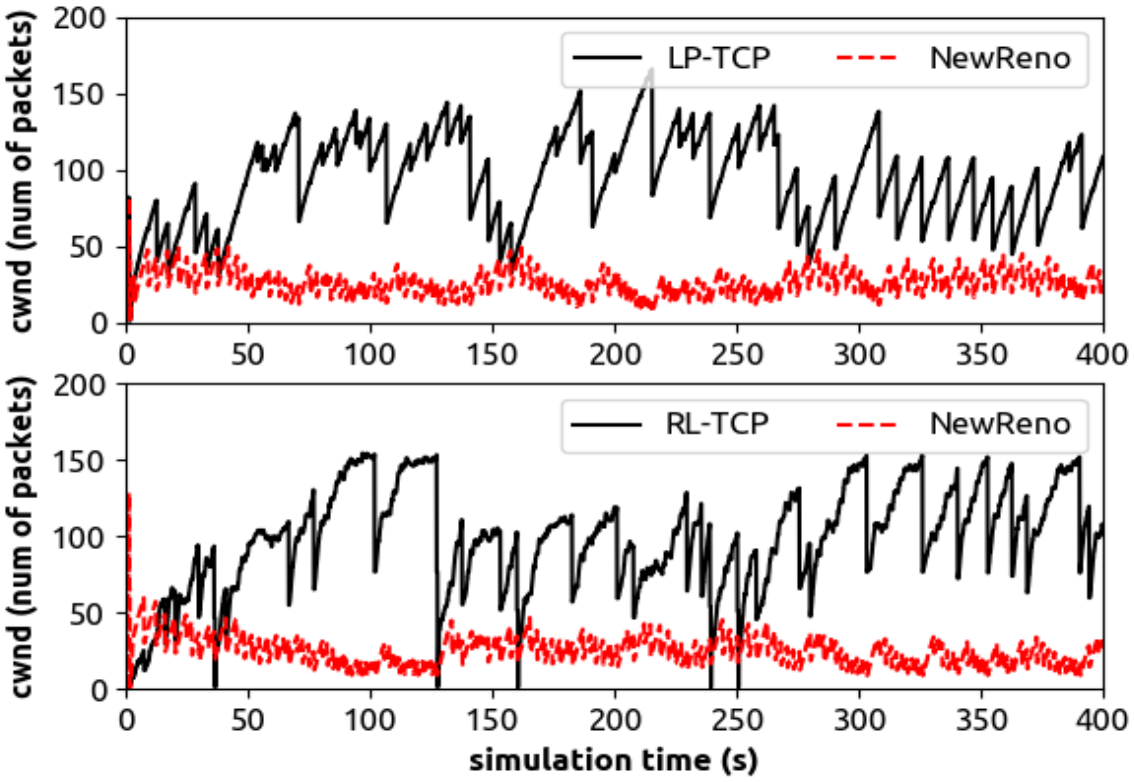
- Performance of RL-TCP is less sensitive to the varying buffer size

Multiple senders

- 4 senders, homogeneous, L = 50



- 3 NewReno, 1 LP-TCP or RL-TCP, L = 50



Conclusions

- Propose two learning-based TCP congestion control schemes for wired networks
- LP-TCP works the best with small buffers at the bottlenecks
- RL-TCP achieves the best throughput-delay-tradeoff under various network configurations

- Future work
  - Explore policy-based RL-TCP
  - Improve fairness for learning-based TCP congestion control schemes

Thank you!

Q & A