ML for Congestion Control

Anirudh Sabnis
What is Congestion Control?

- Use the shared network resources optimally
  - Maximize network utilization
  - Not cause congestion
- Fair Allocation of the network resources to all users
- Congestion Control Algorithms dictate rules which ensure the above goals are met
TCP NewReno

(1) Slow start
Cwnd += 1

(2) Congestion avoidance, cwnd += 1/cwnd

(3) Fast recovery

BW = 10Mbps, RTT$_{\text{min}}$ = 150ms, Single NewReno flow

Additive Increase Multiplicative Decrease
TCP Variants

Vegas
[Brakmo et al. 1995]

Cubic
[Ha et al. 2008]

Compound
[Tan et al. 2006]
Mathematical Model for TCP

- View TCP as a Decentralized Global Optimization Problem

- Global Objective:
  \[ \sum_i w_i \log(x_i) \]

- Subject to network constraints (ie. Link capacities)

- Proved that TCP ensures fair sharing
An Experimental Study of the Learnability of Congestion Control

Anirudh Sivaraman, Keith Winstein, Pratiksha Thaker, Hari Balakrishnan

MIT CSAIL
Learnability of Congestion Control

• How easy is it to learn a network protocol to achieve the desired goal, despite a mismatch in the set of assumptions?
Experiment Setup

Objective Function:
- log (tpt/delay)
- Avg. Flow Completion time

Training Networks \(\rightarrow\) Remy (SIGCOMM 13) \(\rightarrow\) RemyCC \(\rightarrow\) Test within ns-2

Testing Networks
Experiment Setup

- Simple Topology
- Simulations run on NS2
- Each experiment run 128 times
Prior Result (with perfect knowledge)

- 15 mbps dumbbell topology, 
n = 8 senders, 
flows of exponentially distributed byte length (mean 100 kilobytes), exponentially distributed off-time (mean 5 seconds)
Congestion Control with Imperfect knowledge

- How does REMY perform with imperfect knowledge about
  - link speed
  - degree of multiplexing
  - structure of the network
  - Incumbent protocols running simultaneously
Knowledge of Link Speed

<table>
<thead>
<tr>
<th>Tao</th>
<th>Link speeds</th>
<th>RTT</th>
<th>Number of senders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000x</td>
<td>1–1000 Mbps</td>
<td>150 ms</td>
<td>2</td>
</tr>
<tr>
<td>100x</td>
<td>3.2–320 Mbps</td>
<td>150 ms</td>
<td>2</td>
</tr>
<tr>
<td>10x</td>
<td>10–100 Mbps</td>
<td>150 ms</td>
<td>2</td>
</tr>
<tr>
<td>2x</td>
<td>22–44 Mbps</td>
<td>150 ms</td>
<td>2</td>
</tr>
</tbody>
</table>

(a) Tao protocols designed for breadth in link speed

<table>
<thead>
<tr>
<th>Link speeds</th>
<th>RTT</th>
<th>Number of senders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–1000 Mbps</td>
<td>150 ms</td>
<td>2</td>
</tr>
</tbody>
</table>

(b) Testing scenarios to explore breadth in link speed

Scenario for the “Knowledge of Link Speed” experiment
Knowledge of Link Speed

Being more specific about the Link Speed boosts the performance
Knowledge of degree of multiplexing

<table>
<thead>
<tr>
<th>Tao</th>
<th>Link speeds</th>
<th>On avg.</th>
<th>Off avg.</th>
<th>min-RTT</th>
<th># senders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tao-1–2</td>
<td>15 Mbps</td>
<td>1 sec</td>
<td>1 sec</td>
<td>150 ms</td>
<td>2</td>
</tr>
<tr>
<td>Tao-1–10</td>
<td>15 Mbps</td>
<td>1 sec</td>
<td>1 sec</td>
<td>150 ms</td>
<td>10</td>
</tr>
<tr>
<td>Tao-1–20</td>
<td>15 Mbps</td>
<td>1 sec</td>
<td>1 sec</td>
<td>150 ms</td>
<td>20</td>
</tr>
<tr>
<td>Tao-1–50</td>
<td>15 Mbps</td>
<td>1 sec</td>
<td>1 sec</td>
<td>150 ms</td>
<td>50</td>
</tr>
<tr>
<td>Tao-1–100</td>
<td>15 Mbps</td>
<td>1 sec</td>
<td>1 sec</td>
<td>150 ms</td>
<td>100</td>
</tr>
</tbody>
</table>

(a) Tao protocols designed for breadth in multiplexing

<table>
<thead>
<tr>
<th>Link speeds</th>
<th>On avg.</th>
<th>Off avg.</th>
<th>min-RTT</th>
<th># senders</th>
<th>Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 Mbps</td>
<td>1 sec</td>
<td>1 sec</td>
<td>150 ms</td>
<td>1–100</td>
<td>5 BDP, no drop</td>
</tr>
</tbody>
</table>

(b) Testing scenarios to explore breadth in multiplexing

Scenarios for the “knowledge of degree of multiplexing” experiment
Knowledge of degree of multiplexing

Tao protocols perform well across a wide range of multiplexing, but at the cost of diminished performance when there are very few senders. However, training to accommodate lower degrees of multiplexing degrades performance at higher degrees of multiplexing.
Conclusions

- Can tolerate mis-match link assumptions
- Need precision about number of senders
- TCP compatibility is a double-edged sword
- Can tolerate mismatch in number of bottlenecks
Improving TCP Congestion Control with Machine Intelligence

Yiming Kong     Hui Zang     Xiaoli Ma
Improving TCP Congestion Control with Machine Intelligence

- **Problem**: TCP lowers its cwnd to half on a packet loss thus underutilizing the network resources

- Teach TCP to optimize its cwnd to minimize packet loss events
  - LP-TCP

- Teach TCP to adaptively adjust cwnd according to an objective in a dynamic setting
  - RL-TCP
Loss Prediction based TCP (LP-TCP)

- When a new ack is received, cwnd += 1/cwnd
- Before sending a packet,
  - Sensing engine updates state as feature vector
  - Loss predictor outputs loss probability (p)
  - If p < threshold, actuator sends the packet
  - Otherwise cwnd -= 1
Training the Loss Predictor

• Collect training data using NewReno simulations using on ns2
  • Record the state right before the packet goes into transmission as a feature vector
  • If the packet is successfully delivered, the feature vector gets a label 1, else 0
  • Stop collection when we have enough losses in the data

• Train a Random Forest Classifier

• 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.
Reinforcement learning based TCP (RL TCP)

- Reinforcement Learning basics
- Q value for a state - Potential Value of being in a state and performing an action
  - Denoted as Q(s, a)
  - a is the action taken while in state s

\[ Q(s, a) \leftarrow r + \max_{a' \in A} Q(s', a') \]

- Goal: Learn the best action in each state
RL-TCP Objective

- Learn to adjust cwnd to increase the following 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

Utility function for RL TCP
RL TCP

- Map Congestion control as a *Reinforcement Learning Problem*

**State S**
- ewma of ack inter arrival time
- ewma of packet inter sending time
- RTT ratio
- Slow start threshold
- Current cwnd size

**Action a**
- \( \text{cwnd} \pm a \), where \( a = -1, 0, 1, 3 \)

**Reward r**

\[
\Delta_{n+1} = U_{n+1} - U_n
\]

\[
r_{n+1} = \begin{cases} 
10, & \Delta_{n+1} \geq 1 \\
2, & 0 \leq \Delta_{n+1} < 1 \\
-2, & -1 \leq \Delta_{n+1} < 0 \\
-10, & \Delta_{n+1} < -1.
\end{cases}
\]
Learning the Q-Value

- Learning the Q-Value: $Q(s,a)$
  - $Q(s,a)$ is the value of being in state $s$ and performing action $a$.
  - Updated at every iteration using SARSA

\[
Q(s_{n-1}, a_{n-1}) \leftarrow r_n + \gamma Q(s_n, a_n),
\]

- Action selection - Using exploration and exploitation.

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

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

- Buffer size $L = 5$

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

- 4 senders, homogeneous, $L = 50$

![Graph showing comparison of average throughput (Mbps) and average delay (ms) for different TCP variants: LP-TCP, NewReno, and Q-TCP. The graph indicates RL-TCP as the best choice with an average throughput of 0.592 Mbps and average delay of 0.545 ms.]
Conclusion

- Proposed two learning based TCP congestion control algorithms
- RL TCP works best with small buffers at bottlenecks
- RL TCP provides the best throughput delay tradeoff under various network conditions