I. Introduction

Introduction
The TCP congestion control algorithm has been contributing to the stability of the Internet. Given its crucial role, it is one of the popular research topics in networking, resulting in numerous TCP variants proposed over the years. Unfortunately, the current ns-3 standard release supports a very small subset of them. As part of our ongoing effort to extend ns-3 TCP functionalities, we implement an additional congestion control algorithm called TCP Illinois and validate the model by simulating the protocol under multiple network scenarios and comparing it with other TCP variants.

TCP Illinois Algorithm
TCP Illinois proposed to achieve high throughput in high-speed networks is a hybrid congestion control algorithm that uses both packet loss and delay information as congestion signals. The packet loss information is used to decide on whether to increase or decrease the congestion window, but the amount of change that is made to the congestion window depends on the delay measurements. Illinois makes the choice of the inflation factor, $\alpha$, and deflation factor, $\beta$, based on the network’s congestion status. When the network is far from congestion (i.e. when the average queuing delay is small), Illinois sets $\alpha$ large and $\beta$ small. As the network approaches congestion (i.e. when the average queuing delay is large), the value of $\alpha$ is decreased while the value of $\beta$ is increased. Thus, in Illinois, $\alpha$ and $\beta$ are functions of the queuing delay as calculated in Equation 1 and Equation 2, respectively. These values of $\alpha$ and $\beta$ are then used to update the congestion window during the congestion avoidance and fast recovery phases similar to the standard AIMD (additive increase multiplicative decrease) algorithm as shown in Equation 3 and Equation 4.

\[
a = f_1(d_a) = \begin{cases} 
\alpha_{\text{max}} & \text{if } d_a \leq d_1 \\
\frac{\alpha_{\text{max}}}{\beta_{\text{max}}} & \text{otherwise}
\end{cases}
\]

\[
\beta = f_2(d_a) = \begin{cases} 
\beta_{\text{max}} & \text{if } d_a \leq d_2 \\
\frac{\beta_{\text{max}}}{\beta_{\text{min}}} & \text{if } d_2 < d_a < d_3 \\
\beta_{\text{min}} & \text{otherwise}
\end{cases}
\]

\[\text{cwnd} = \text{cwnd} + \alpha \text{cwnd}
\]

\[\text{cwnd} = \text{cwnd} - \beta (\text{cwnd})
\]

II. IMPLEMENTATION

Class Interaction

III. IMPLEMENTATION

Main Functions in TcpIllinois Class

- CalculateAvgDelay() - Compute average queuing delay every RTT
- CalculateMaxDelay() - Compute maximum (average) queuing delay every RTT
- CalculateAlpha() - Calculate additive increase factor every RTT
- CalculateBeta() - Calculate multiplicative decrease factor every RTT
- PktsAcked() - Filter RTT measurements for basett and maxRtt
- IncreaseWindow() - Adjust cwnd following Illinois congestion avoidance algorithm
- GetSSThresh() - Compute ssthresh after congestion event

IV. SIMULATION STUDY

- We use dumbbell topology to evaluate the performance of the algorithm.
- The Table shows the shared parameters among all the simulation experiments.

<table>
<thead>
<tr>
<th>Simulation Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access link bandwidth</td>
<td>10 Mbps</td>
</tr>
<tr>
<td>Bottleneck link bandwidth</td>
<td>6-10 Mbps</td>
</tr>
<tr>
<td>Access link propagation delay</td>
<td>0.1 ms</td>
</tr>
<tr>
<td>B. link propagation delay</td>
<td>100 ms</td>
</tr>
<tr>
<td>Packet MTU size</td>
<td>1500 B</td>
</tr>
<tr>
<td>Error model</td>
<td>Uniform error model</td>
</tr>
<tr>
<td>Application type</td>
<td>Bulk send application</td>
</tr>
<tr>
<td>Simulation time</td>
<td>200 s-600 s</td>
</tr>
<tr>
<td>Queue size</td>
<td>BDP</td>
</tr>
</tbody>
</table>

V. SIMULATION RESULTS

- We measure the robustness of Illinois to random loss in comparison with the standard NewReno and HSTCP.
- Illinois performs better than both HSTCP and NewReno.
- We study the TCP-friendliness (left plot) and intra-fairness of Illinois (right plot).

VI. CONCLUSION

- In this poster, we present our implementation of TCP Illinois congestion control algorithm in ns-3 as part of our effort to extend ns-3’s functionalities. We then show some results obtained from our verification and validation of the model. Illinois, with its ability to dynamically adapt its additive increase and multiplicative decrease factors to the network condition, performs better than HSTCP and the standard NewReno in the presence of random loss. However, Illinois fails to achieve TCP-friendliness and intra-fairness in our simulation scenario. For future work, we plan to extend the study of Illinois in a higher BDP network environments.

VII. REFERENCES