Compound TCP (CTCP) Implementation in ns-3

I. INTRODUCTION and BACKGROUND

• Motivation

TCP is the dominant transport protocol to transfer data in the Internet. However, the performance of TCP changes in different environments like wireless networks or networks with a high bandwidth-delay product. The congestion control mechanisms of the original protocol react poorly in such environments; In order to solve this problem, various solutions have been proposed to increase the performance and fairness in such environments; however, comparing the performance of these solutions are not very easy; because an environment with similar condition is necessary. In order to contribute to an easier comparison and promote open source network simulators, we implement Compound TCP (CTCP) in ns-3 simulator and compare its performance with other available TCP variants in ns-3.

CTCP Algorithm

CTCP works based on both loss-based and delay-based mechanism.

In slow start phase, CTCP works the same as other standard TCP protocols. However, it changes the loss-based control law of the standard TCP by adding delay-based functionality to this component, which includes both the congestion avoidance and fast retransmit/recovery phases. CTCP defines another control variable, dwnd (delay window), to control the loss-based part of the algorithm. Therefore, the sending window is controlled by both dwnd and cwnd, which are responsible for the delay-based and lost-based part of the control law, respectively. Thus, the sending window is calculated as below:

\[ \text{dwnd} = \min(\text{cwnd} + \alpha \cdot \text{dwnd}, \text{dwnd}) \]

Where dwnd identifies the advertised window from the receiver.

\[ \text{cwnd} \]

is updated the same way as other standard TCP protocols in the congestion control phase. It is increased by 1MSS every RTT and divided by two when a packet loss is captured. However, dwnd is updated as below:

\[ \text{dwnd}(t + 1) = \left\{ \begin{array}{ll} \text{dwnd}(t) + \alpha \cdot \text{dwnd}(t) - \beta \cdot \text{dwnd}(t) - \eta \cdot \text{dwnd}(t) - \gamma \cdot \text{cwnd} - \text{dwnd}(t) \cdot (1 - \beta) & \text{if diff} \leq \gamma \\ \text{dwnd}(t) & \text{if diff} > \gamma \end{array} \right\} \]

\[ \text{dwnd} \]

shows the difference between the expected and actual throughput calculated as below:

\[ \text{diff} = (\text{expected} - \text{actual}) \cdot \beta \cdot \text{dwnd} \]

where baserrt keeps minimum RTT over the path.

II. CTCP ARCHITECTURE and EVALUATION

• CTCP Architecture

Figure 1 shows the class relationship in CTCP architecture implemented in ns-3.

• Simulation Results

We evaluate the CTCP protocol in different simulation conditions. The parameters are illustrated in table 1.

The topology used for the simulation is shown in Figure 2. Figure 3 shows congestion window and throughput for different protocols. Figure 4 illustrates fairness among different flows.

### Table 1: Simulation parameters

<table>
<thead>
<tr>
<th>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>2 Mbps</td>
</tr>
<tr>
<td>Access link propagation delay</td>
<td>10 ms</td>
</tr>
<tr>
<td>Bottleneck propagation delay</td>
<td>125 ms</td>
</tr>
<tr>
<td>Packet MTU size</td>
<td>1500 B</td>
</tr>
<tr>
<td>Delayed ACK count</td>
<td>2 segments</td>
</tr>
<tr>
<td>Delayed ACK timeout</td>
<td>200 ms</td>
</tr>
<tr>
<td>Error model</td>
<td>Uniform error model</td>
</tr>
<tr>
<td>Error rate</td>
<td>0.005</td>
</tr>
<tr>
<td>Application type</td>
<td>Bulk send application</td>
</tr>
<tr>
<td>Simulation time</td>
<td>600 s</td>
</tr>
</tbody>
</table>

III. CONCLUSION

• Summary and Conclusion

We show the design of the CTCP protocol and explain how it is implemented in ns-3. We also verify our implementation by comparing the behavior of CTCP in different conditions and with other protocols, namely Reno and Vegas. We choose these two protocols because they are loss-based and delay-based, respectively; and CTCP is the combination of both mechanisms. We verify different phases of the CTCP protocol with respect to other protocols. We observe that CTCP has higher throughput than the other two protocols. This is due to the faster recovery mechanism in the fast recovery/retransmit phases of CTCP.

Future Work

We plan to implement and compare HTCP algorithm with other transport algorithms available in ns-3 including Vegas, Reno, New Reno and Scalable. We plan to put more stress on CTCP to study its behavior in higher bandwidth-delay product.

IV. REFERENCES


