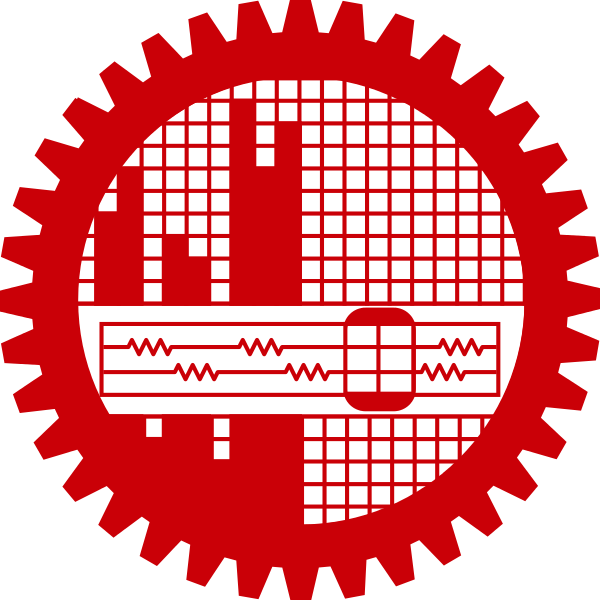
**NS2 Project Report**

**Md. Shahrukh Islam**

**Student ID: 1805098**



**Department of Computer Science and Engineering**

**Bangladesh University of Engineering and Technology**

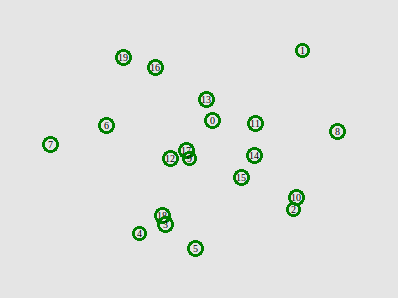
**Introduction**

NS2 is an open-source event-driven simulator designed specifically for research in communication networks. In this project, I have to vary different parameters such as number of nodes, number of flows, number of packets per second, speed of nodes and measure some metrics and plot them in some graphs. I also had to modify some mechanisms of Retransmission Timeout calculation and compare the modified version with the original implementation of NS2.The paper from which I have taken idea from is [The Peak-Hopper: A New End-to End Retransmission Timer for Reliable Unicast Transport](https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.2748&rep=rep1&type=pdf)

**Simulated Networks**

• Wireless 802.11 (dynamic) network using AODV routing in Random topology

• Wireless 802.15.4 (dynamic) network using DSDV routing in Random Topology



**Parameters and Metrics**

**Parameters**

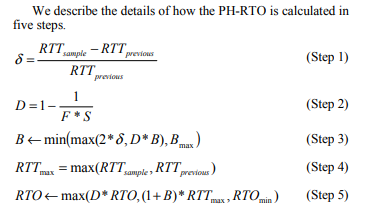
1. Number of nodes (20, 40, 60, 80, 100)
2. Number of flows (10, 20, 30, 40, 50)
3. Number of packets per second (100, 200, 300, 400, 500)
4. Speed of nodes (5, 10, 15, 20, 25)

**Metrics**

1. Network Throughput
2. End-to-End Delay
3. Packet Delivery Ratio
4. Packet Drop Ratio
5. Energy Consumption

**Overview of the proposed algorithm**

The PeakHopper algorithm essentially runs two RTO algorithms in parallel. One algorithm (Short-Term History RTO) monitors the present and short-term history in order to respond to RTT increases. The other algorithm (Long-Term History RTO) simply decays the current value of RTO, and can therefore be said to represent the long-term history.

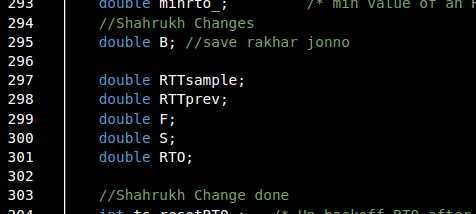
****

1. Having collected a new RTT sample, RTTsample, we compare this value to the previous RTT sample collected, RTTprevious, as shown in Step 1. We call the normalized change between these two samples delta. This is the measure of the short-term changes in RTT.
2. In Step 2, we further define a decay factor, D. D determines how rapidly the RTO is decayed. We also introduce a fader variable, F, which controls the speed of this decay (a high F gives a slow decay and vice versa).
3. In Step 3, we calculate a booster variable B. The booster variable determines how high the RTO should hop when a large RTT increase has been detected.
4. In Step 4, we set RTTmax to the maximum of the new RTT sample, RTTsample, and the previous RTT sample, RTTprevious. RTTmax is used to represent the short-term history of the RTT.
5. In step 5, We set RTO to the maximum of a long-term history (represented by the term D\*RTO) and the short-term history (represented by the term ((1+B)\*RTTmax).
6. In final step (Step 6), we ensure that the RTO does not fall below the minimum allowed RTO.

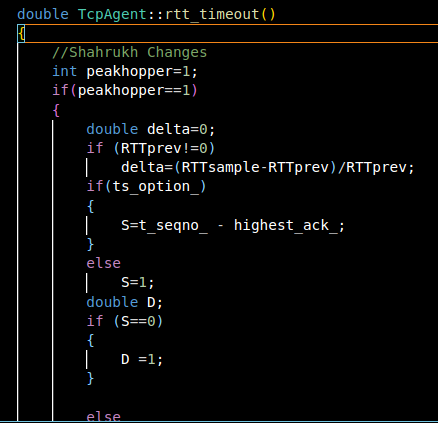
**Modifications:**

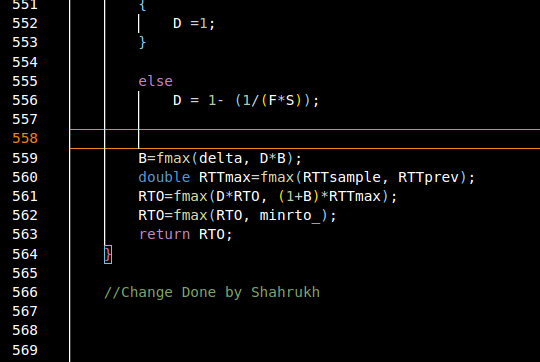
I have modified the RTO calculation algorithm in NS2. The relevant files were tcp.h and tcp.cc.

In tcp.h new variables are declared:



In rtt\_timeout() function new mechanism is implemented.





**Graphs:**

802.11 network:

**Varying no of nodes:**

**Varying no of flows:**

**Varying speed:**

**Varying packet per second:**

**802.15.4 network:**

**Varying no of node:**

**Varying no of flows:**

**Varying speed of nodes:**

**Varying packet rate:**

**Summary:**

**Explanation of Results found in 802.11 static network**

**Varying Number of Nodes**

* + Throughput: Decreases as increase of node congesting the network flow more
  + End-to-End Delay: Increases because of more congested network
  + Delivery Ratio: Decreases with increase of more nodes congested in same network
  + Drop Ratio: Starts to increase a little bit because of network congestion.
  + Energy Consumption: Increases because of more congested network.

**Varying Number of Flows**

* + Throughput: Increases due to more flow in the network
  + End-to-End Delay: Increases due to more flow in the network
  + Delivery Ratio: Shows a trend of decrease with increasing flows
  + Drop Ratio: Increases because of network congestion.
  + Energy Consumption: Increases because of more congested network.

**Varying Speed of Nodes**

* Throughput: Increases due to more movement of the nodes.
* End-to-End Delay: Decreases due to more movement of the nodes.
* Delivery Ratio: Shows a trend of decrease.
* Drop Ratio: Increases because of randomness in a large network.
* Energy Consumption: Remains almost same.

**Varying Number of Packets/Second**

* Throughput: Increases and then becomes flat.
* End-to-End Delay: Increases and then becomes flat.
* Delivery Ratio: Increases and then becomes flat.
* Drop Ratio: Decreases and then becomes flat.
* Energy Consumption: Increases and then becomes flat.

**Explanation of Results found in 802.15.4 static network**

**Varying Number of Nodes**

* + Throughput: Decreases as increase of node congesting the network flow more
  + End-to-End Delay: Decreases because of more nodes
  + Delivery Ratio: Decreases with increase of more nodes congested in same network
  + Drop Ratio: Starts to increase a little bit because of network congestion.
  + Energy Consumption: Increases because of more congested network.

**Varying Number of Flows**

* + Throughput: Increases due to more flow in the network
  + End-to-End Delay: Increases due to more flow in the network
  + Delivery Ratio: Shows a trend of decrease with increasing flows
  + Drop Ratio: Increases because of network congestion.
  + Energy Consumption: Increases because of more congested network.

**Varying Speed of Nodes**

* Throughput: Increases and then decreases due to speed increment.
* End-to-End Delay: Increases and then decreases due to more movement of the nodes.
* Delivery Ratio: Shows a trend of increase.
* Drop Ratio: Decreases because of randomness in a large network.
* Energy Consumption: Increases slightly.

**Varying Number of Packets/Second**

* Throughput: Increases and then becomes flat.
* End-to-End Delay: Increases and then becomes flat.
* Delivery Ratio: Increases and then becomes flat.
* Drop Ratio: Decreases and then becomes flat.
* Energy Consumption: Increases and then becomes flat.

**Explanation of Performance Metrics Result in RTT/RTO modification algorithm:**

From the mean retransmission graph in the paper, it’s seen that modified algorithm provides a lower mean retransmission than default Jacobson’s implementation. It was expected because a lower mean indicates that retransmission timer follows a narrow waiting time and a higher mean indicates to a longer waiting time.

Another metric, known as the mean of RTO\_N/RTT\_L where RTO\_N is the newly calculated RTO and RTT\_L is the last RTT sample prior to the RTO update. We then calculated P as the mean value of this fraction over an entire simulation run. This mean is found to be lower in modified algorithm than in default implementation which is expected too. A retransmission timer with lower mean of this ratios is expected to perform better.

In graphs we noticed that throughput has been increased after this modification that means per second number of received bits has been increased. That means a smaller RTO has occurred and packet drop is early detected. And for this reason throughput has been increased after modification.