EEE4121F Module B Project Design

Using buffer queuing metadata to aid in congestion control

Protocol architecture Design

- ***Service Requirements:***

  The protocol has been designed to aid in managing the network better. It will do so through the monitoring of not only the overall time taken to transmit a packet, but also the time spent in buffer queues, and how full the buffers are (packet assigned a unique ID pertaining to queue in buffer). This information, along with the already available link weights will be used to help reduce congestion and delays due to traffic in the network by adjusting the routing table where possible so that idle/less busy paths are used as an alternative to the currently busy.

- ***Protocol Specification:***

  Information to collect – source and destination addresses, current link weight (from prev hop), whether or not the packet had to be put in a buffer(1 or 0), t_d in buffer queue, buffer id (used to determine how many packets were already in the buffer). Protocol implements a function that updates the link weights to take into consideration the current congestion in the network indicated by the number of packets and the time spent in the buffers.

When the packet arrives and is put in the buffer, the buffer bit is set to 1, an ID is given to represent the position in the buffer queue on entry and a timer is started and only stopped when the packet leaves the buffer.

At each router, the respective fields are updated just before the packet is forwarded. If the packet was not in a buffer (bit = 0), then the links do not have to be changed, otherwise the t_d and buffer id fields are used to find a new link weight.

This protocol would work hand in hand with whichever layer 3 protocol that is being implemented and would be implemented first.

- ***Protocol Requirements*:**

  The protocol needs to segment the packet data to the network layer segment, monitor the traffic congestion using the state of each router buffer in the network [1]. The relevant fields need to be updated at each router

Implementation

With the use of p4 programming, an extra header field is added to the packets to be transmitted through the network. This header includes information pertaining to the buffer queues. The program makes use of the standard metadata that the v1 model tracks through queuing metadata. This includes the time a packet spends in the queue, the depth of the buffer queue as the packet arrived, and the depth of the queue as the packet leaves the queue [1]. This information is used to signal congestion; particularly, if the queues are getting longer. In doing so, the link state is set

to ‘fail’ mode and this temporarily disables that link. The next shortest path, if available is then recalculated using a simple shortest paths algorithm.

To verify the functionality, a script is run to send traffic into the network created. All the nodes are connected, and the paths taken are printed out. Over time, different hops are observed for the same end to end connections where a router in the middle has more traffic than an alternate longer path. This means that the time taken to transmit the packets should decrease.

The code used to implement the protocol can be found in the github repo below:

https://github.com/BerthaMolai/EEE4121F_ModuleB_Project.git

Conclusion

The results showed that although the time spent sending packets significantly decreases when the increase in traffic, the overall protocol does slow down the network. This may be due to the fact that the overhead of checking an extra header at each router overweighs the benefits of avoiding routing packets through busy paths when other paths are free. This implementation also did not take into full account the possibility of the queues growing in all possible paths. Thus, the network sees a peak in dropped packets.

This protocol could be quite useful if used with other congestion control mechanisms as an aid, and not the main congestion control protocol.

The experiment simply temporarily ‘disabled’ the link and made the following packets be rerouted if the queue depth was shown to be

increasing. In addition to using this condition, another condition: deq_qdepth && deq_timedelta > some threshold value could be used. Further, the queue depth and timedelta metadate could be used to recalculate link weights as initially intended thus triggering a need to recalculate the new shortest path. This would result in less packet loss.

Bibliography

[1] "Functions of a protocol," tutorialspoint, [Online]. Available: https://www.tutorialspoint.com/what-is-the-function-of-protocol .

[2] "p4lang github," [Online]. Available: https://github.com/p4lang/p4c/blob/main/p4include/v1model.p4#L59. [Accessed 12 May 2022].

[3] Networkx, "Networkx Documentation," Networkx, 2022. [Online]. Available: https://networkx.org/documentation/stable/reference/functions.html . [Accessed May 2022].

[4] "P4 learning github repo," [Online]. Available: https://github.com/nsg-ethz/p4-learning.git.