Evan Lancaster
CPE 400
December 12th, 2022

Dynamic Routing Mechanism Design with Focus on Throughput

Introduction

This project entails the simulation of a network that allows for the user to create routers which have their own queues, bandwidths, failures, etc. This simulation is intended to be as realistic as possible. The routers are intended to be in a decentralized network. The code provided sets up the example given by the project problem statement, with a total of 16 routers with varying bandwidth. The simulation also allows for nodes to periodically break, and routers will handle routing around the broken link. The simulation generally requires that routers ask each other for information, but bandwidth is given to the routers directly. For error handling, or routing setup/fixing, routers are required to broadcast packets with crafted information containing their new routing information. The simulation isn't perfect, but refrains from non-realistic assumptions regarding router communication as much as possible.

The routers are designed to maximize throughput. Dropped packets are a form of lost throughput, so they are also designed to react to dropped packets. The technical details of the handling of throughput and dropped packets are discussed in detail later in the report. The next section will introduce the simulation's router details.

Functionality

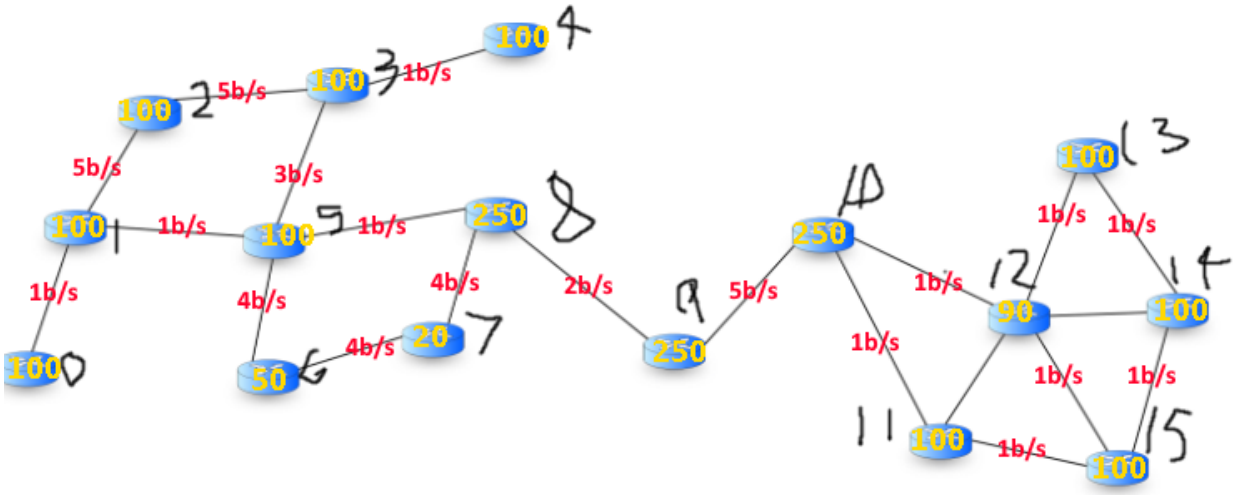

Fig 1. Setup for the simulation's routers' queue size (yellow) and bandwidth (red)

General Simulation Setup

Figure 1 shows some important router information. On each router, the yellow number represents the amount of data that can be stored in their queue. The red number represents a connection's bandwidth between two nodes, and it is the same in both directions. This bandwidth is allowed to change at runtime, and the routers will handle it. The black numbers represent the identification numbers for each router. The routers are created from information given by the two files "connection_info.ri" and "router_info.ri". The first file's entries represent a connection between two routers, and are entered as the following:

routerA|routerB|bandwidth

There should be commas between entries. The router_info.ri file represents the queue size for each router, and entries should simply be

queueSize

There should be commas between entries. Routers store their physical connections to other routers directly within their class, along with that connection's information. This is because the simulation uses a modified version of the Distance Vector Algorithm. With enough time, the system will always converge to the optimal path. The tradeoff for a small initialization time allows the routers to dynamically find new routes when

bandwidths change, or when routers in a path become inoperable, without the need for a centralized system.

Router Class Information

Routers store information about their ID, connections, routing table, queued packets, queue size, the number of routers in the AS, and information for the packet they are sending. The routing table entries indices correspond to their IDs, and are represented as the following

[destinationID, costToDestination, nextRouterObject]

The destination ID is any given router in the network that we eventually want to get to. The costToDestination is a number which represents the total cost to get to the router. The nextRouterObject is the connection we should take if we want to get to this router. The cost to destination is the most complicated attribute of the routing table, and it is part of the novelty of the routing algorithm. In simple terms, it is sometimes modified at runtime by routers, and it is some percentage of the amount of time it would take to send one unit of data through the connection added to the next router's cost to the destination (which is given by broadcasts). The following figure will describe some more setup for the routers.

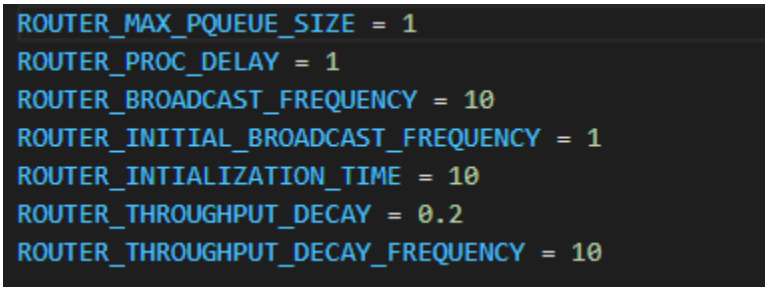

Fig 2. Constants for the routers in the simulation

Figure 2 lists some constants the simulation uses to decide when to do various things, how much to decay throughput reduction, etc. Below is a list of what each constant represents.

ROUTER_MAX_PQUEUE_SIZE: unused for the provided simulation; used to set the default queue size for all the routers when file input is not used

ROUTER_PROC_DELAY: the amount of time that it takes for the routers to process a packet

ROUTER_BROADCAST_FREQUENCY: the amount of time the router should wait in between sending simple packets to its connections containing changes to its routing table

ROUTER_INITIAL_BROADCAST_FREQUENCY: the amount of time the router should wait in between sending simple packets to its connections containing changes to its routing table during the ROUTER_INITIALIZATION_TIME
ROUTER_INITIALIZATION_TIME: the amount of time the routers are in "initialization mode", where their routing tables are completely unfilled and lots of information needs to be broadcasted often for network initialization
ROUTER_THROUGHPUT_DECAY: the multiplier which decides how much of a router's throughput is regained after each frequency tick if a connection has been throttled
ROUTER_THROUGHPUT_DECAY_FREQUENCY: the amount of time a router waits between allowing a connection to regain a portion of its throughput after packet loss

In order to maximize throughput, we need routers with as much queueing size as possible to minimize packet drops, and we need routers with as much bandwidth as possible to get the packets to where they need to be quickly. Additionally, we need the correct route.

Packet Data and Evaluation

An important functionality of the routing protocol is the 3 packet type definitions. Technically there is a fourth, but it is not in use. The packet types, their information, and how their information is used is listed below.

PACKET meta_type 1: Broadcasted Routing Packet
This packet is the bare bones of the routing functionality. On every tick of ROUTER_BROADCAST_FREQUENCY, the router will craft a packet with this meta_type. The packet will have a meta_table containing information about any changes made to its routing table. This packet is sent only to direct connections to routers, and no further.
On receiving this packet, the router will cross reference the information given with its own routing table. For each of the cross referenced routes that have changed and are used for the receiving router's routing table, the routing table is updated directly. This part of the system is almost exactly what you find in a decentralized routing protocol.

PACKET meta_type 2: Routing Trace Packet
This packet doesn't provide anything for the routing, but it allows us to track the exact path that a packet takes through the routers. Every time the packet is transmitted between two routers, the router adds its identification number to the top of the tracing stack as well as the cost of the path that it is going to take.

PACKET meta_type 3: *Bandwidth Throttle Packet*

This packet allows routers to throttle one way of the connection between them and another router. The packet provides a percentage value that they wish to set the bandwidth from another router to them. This packet type is always sent for packet loss. The default value for the percentage is 50%, which sets the bandwidth to half of the value it is currently at. An important part of this is that it only throttles one way; the sending router can still send at full bandwidth. The receiving routers remember the maximum bandwidth that they have been at, and eventually climb back up to it. Every ROUTER_THROUGHPUT_DECAY_FREQUENCY ticks, the router which received this packet will calculate the updated percentage based on the following formula in the checkConnectionThrottles function

$$if\ throttlePercentage\ >=\ 0.95,\ set\ throttle\ =\ 1$$
otherwise,
$$throttle\ +=\ (1\ -\ throttle)\ *\ ROUTER\ THROUGHPUT\ DECAY$$

This function will exponentially bring throttle back towards its maximum with respect to the ROUTER_THROUGHPUT_DECAY variable, since 0 > throttle >= 1. This throttling is meant to allow the network to continue to send as many packets as possible, but through different routes, so that one node doesn't become overloaded. In the worst case scenario, where there is no other route to take, throughput is still maximized, since we are limited to the bandwidth of the choke point. However, this throttling will help even in this worst case scenario, as we will see less dropped packets due to full queues.

Simulation Details

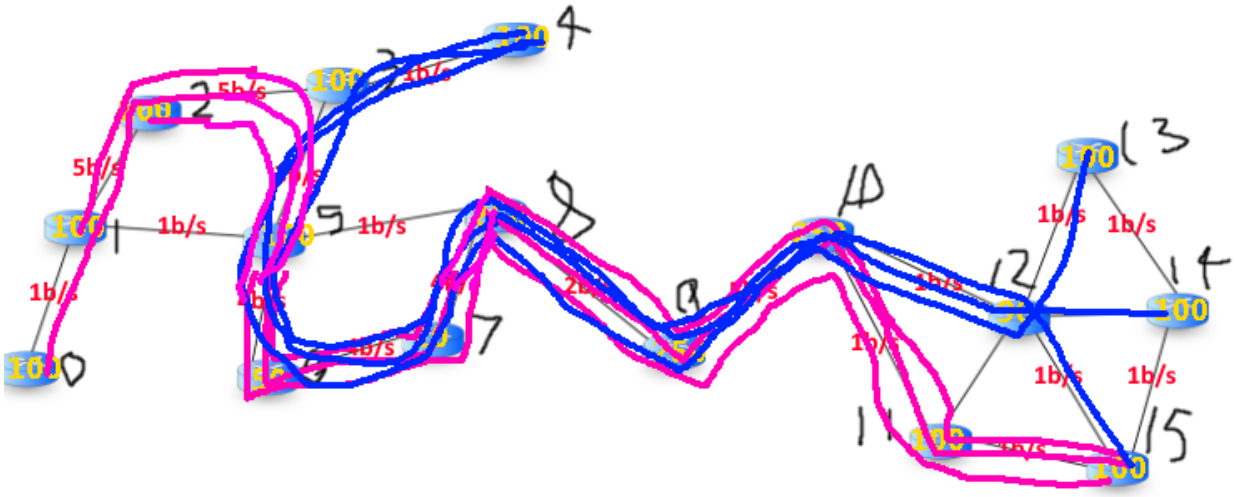

Fig 3. Example routing setup for congestion

Six routers are told to send 9 packets, each with a size of 10, to the other side of the network. The pink lines represent 27 packets from routers 0, 1, and 2 to router 15. The

blue lines represent packets sent from routers 15, 14, and 13 to router 4. The following is the output of the code setup and running of this simulation.

```
PS C:\Users\Evan\CPE400_Proj> & C:/Users/Evan/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Evan/CPE400_Proj/main.py
Routing Trace Packet from r0 to r15
Routing Trace Stack: r0 -> r1 -> r2 -> r3 -> r5 -> r6 -> r7 -> r8 -> r9 -> r10 -> r11 -> r15, cost: 518.3333333333334

BEGIN: NETWORK BANDWIDTH TEST; 54 LARGE PACKETS SENT FROM r0, r1, r2, r13, r14, r15

Begin packet traces during high network use
Packet drop: queue size exceeded at r7. Throttling connection to r6
Packet drop: queue size exceeded at r7. Throttling connection to r6
Packet drop: queue size exceeded at r12. Throttling connection to r14
Packet drop: queue size exceeded at r7. Throttling connection to r6
Routing Trace Packet from r2 to r15
Routing Trace Stack: r2 -> r3 -> r5 -> r8 -> r9 -> r10 -> r11 -> r15, cost: 423.33333333333337
Routing Trace Packet from r0 to r15
Routing Trace Stack: r0 -> r1 -> r2 -> r3 -> r5 -> r8 -> r9 -> r10 -> r11 -> r15, cost: 543.3333333333334
Routing Trace Packet from r13 to r4
Routing Trace Stack: r13 -> r12 -> r10 -> r9 -> r8 -> r7 -> r6 -> r5 -> r3 -> r4, cost: 478.3333333333333

Packet trace 50 ticks after network cleared
Routing Trace Packet from r0 to r15
Routing Trace Stack: r0 -> r1 -> r2 -> r3 -> r5 -> r6 -> r7 -> r8 -> r9 -> r10 -> r11 -> r15, cost: 518.3333333333334
END: NETWORK BANDWIDTH TEST
```

Fig 4. Output from the simulation for the congested packets

The simulation first gives us the uncongested trace packet from r0 to r15. Due to the setup of the simulation, the best path is taking what "looks" like the longest path from r0 to r15, but observing the bandwidth of the connections, we can see that it is maximized, rather than taking a greedy approach. During the queuing and sending of the 54 packets, the simulation outputs that we have packet drops at router 7 and router 12. The drops at router 7 happen due to the high amount of traffic from both sides of the network at the same time. To deal with this, the router throttles the connection of the source of the packet that got dropped. This information is propagated throughout the network. They happen at router 7 specifically because router 7 has a queueing size of only 20, which allows it to hold a total of 3 of these size 10 packets (one in sending buffer, two in queue). With 54 packets, this is not the best for router 7. By throttling the connection, the routers begin to instead route from router 5 to router 8. Since the connection is only throttled one way, the right side of the network is still able to use router 7 at full speed.

We see all of these properties demonstrated by the routing trace packets after the packet drops. The first route, from r2 to r15, goes from r5 to r8, as the connection of the generally desired path is throttled. The same thing happens from r0 to r15. However, from r13 to r4, the router still takes the route from r8 to r7 to r6 to r5, since it is only throttled from one side. We can also see that the cost from r0 to r15 has increased due to increased congestion to 543, compared to the original 518.

Finally, after the congestion is cleared, the trace packet from r0 to r15 at the bottom shows us that the network has reset to full bandwidth for its routers, sending through r6 and r7 as normal. Next, we will see the functionality of runtime failure for a router.

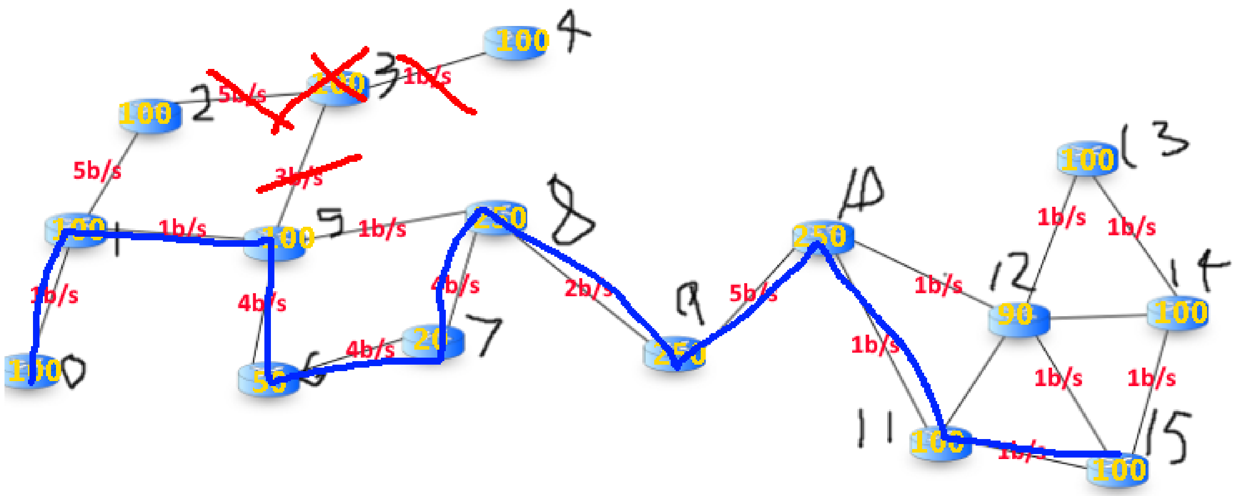

Fig 5. graphical representation of the network with a failed router 3

```
END: NETWORK BANDWIDTH TEST


Forcing Router 3 to fail. Sending trace packet after 5 network ticks.
Routing Trace Packet from r0 to r15
Routing Trace Stack: r0 -> r1 -> r5 -> r6 -> r7 -> r8 -> r9 -> r10 -> r11 -> r15, cost: 545.0
```

Fig 6. simulation output for router 3 failure

Fig 5. shows the path that the routers take when the third router fails. The routers are given five network ticks to propagate the change in routing. Subsequently, the path taken by router 0 is from r1 to r5, instead of r1 to r2 to r3 to r5. This propagation happens relatively fast for closeby routers. This code is run in the same simulation as the congestion, just afterwards, showing that this can happen and adapt at runtime. The next figure shows router 6 failing and the new path taken.

```
Forcing Router 6 to fail. Sending trace packet after 5 network ticks.
Routing Trace Packet from r0 to r15
Routing Trace Stack: r0 -> r1 -> r5 -> r8 -> r9 -> r10 -> r11 -> r15, cost: 570.0
```

Fig 7. simulation output for router 3 and router 6 failure

This portion has both router 3 and router 6 failing. This means that the routers dynamically route directly from r5 to r8 instead of around as in figure 5. The cost is slightly higher due to the lower bandwidth.

Novelty

The novelty of the routing algorithm is the propagation of information and the throttling of specific connections in order to maximize throughput of the system. It is very easily seen to be dynamic, as when routers fail or are added to the system, the routing tables

for the new or destroyed connection are quickly sent to the rest of the routers. The simulation is fairly realistic and requires routers to communicate only through packets. With the information they receive from packets, they update their tables, or their connections. The throttling of a single, one-way connection allows flexibility within the system to route packets in different ways so that one router is not overwhelmed, in order to maximize overall throughput rather than a single router's throughput. Further detail of these systems with visual simulations are in the Router Class Information and Packet Data and Evaluation sections above.

Results and Analysis

This routing algorithm is very good at picking up on new routes and getting all of the routers within the AS to update their routing tables accordingly. This is due to the decentralized distance vector based algorithm which procedurally updates as routers broadcast their changes. The throttling appears to be very useful in forcing the network to take more than one route to maximize throughput, as well as avoid packet loss. In the example given in the simulation, the change in route allows us to utilize the path with the maximum bandwidth for the right side, and a slightly lower bandwidth on the left side, which reduces strain on one particular spot of the network. By reducing packet loss in this way, we increase the throughput of the network.

A problem with this algorithm is that the way that I have set it up, it requires each router to have a routing table for the entire network. This is fine for an AS of only 16 routers, but if there were 1000, it may become difficult for routing tables to stay updated. Since every path change essentially requires a propagation of the change through every single usage of the path in the network, this may not be feasible for a large network, and would require a rework in the way packets are sent. However, for small networks this algorithm works exceptionally well.

Since each router takes their fair burden in calculating route costs, this algorithm is very easy to run. Nothing is particularly difficult to compute regardless, and the packets that are sent with routing information are relatively small, since we only need to know about things that have changed. When a large change is made, like a completely lost connection, these routing information packets may begin to clutter the network, but a scenario like this doesn't happen enough to cause an issue, since the routing information packets aren't large enough to cause significant routing delays.