A little update to the routing:

*Our MPR algorithm:*

Rather than have a set number of MPR nodes, we decided to do sort of an objective, dynamic MPR setup. What this means: none of the nodes considers itself strictly as an MPR or a normal node; rather, each node sort of acts as an MPR for its neighbors depending on whether it is the best node for sending packets to a given neighbor. If it determines that it is the best node, then it will forward packets. Given that our algorithm does not require packets to be specifically addressed to any other nodes, we just automatically forward packets accordingly, and each node decides if it cares about the packet (if it is a new packet, or if it a packet that the node has already seen).

All packets are used to update information about speed, velocity, and neighbors, the latter of which is new for this project. As such, packets now feature an area containing hybrid information: of neighbors of the source are stored here, and nodes that are likely to have been visited by the packet are also stored here. There are ten slots in this area of the packet, allowing hard-coded support for up to 10 nodes during simulation. Each possible node ID has its own slot (wherein the first slot is for ID 10100, and the last slot is for 10109) to allow for a direct search of whether a packet has visited a node.

To keep this information up-to-date, the source node enters its known neighbors in the packet as it adds its position and velocity information, and then passes the packet to at least one of its neighbors. When the packet reaches another node and is forwarded, that secondary node also enters its neighbors as a way of indicating exactly who has seen the packet, without clearing any of the previously stored neighbors. By the end of the chain, the packet should have just about every node being used in the simulation listed.

With this new information, nodes can now take an educated guess as to who the neighbors of other nodes are. This enables nodes to guess if they are the best chance to send the packet to a given node; if there is a stronger link assumed to be available, then the node will skip sending/forwarding the packet directly, instead relying on a known neighbor to pass the packet on.

In a roundabout way, this ensures that nodes alternate exactly who acts as an MPR. Take a given scenario, with a platoon of all ten trucks, already as in-range as possible. With this setup, we can assume that each truck can reach two trucks ahead and two trucks behind, at the very least. Assume that each truck already knows its neighbors (determined by a robust function present in project 1). Now:

* The frontmost truck wants to send a packet. It sends to all of its neighbors, so it updates its neighbors, the second and third truck.
* The second and third trucks successfully receive the packet. The second truck has 1, 3, and 4 as neighbors; truck 3 has 1, 2, 4, and 5. This means that truck two will try to send to 3, looks at the previously visited list, and determines 3 has probably seen it, but agrees to send because it is closer than node 1, so it is a more robust connection. However, it skips sending to 4, as it knows 3 is closer.
* This kind of pattern continues until the end, with some nodes only receiving the packet once, while others receive it multiple times.

With this setup, we make sure that packets are eventually sent. If a node receives a duplicate packet, it simply drops the packet and incurs only a tiny performance penalty; in that way, nodes has a decent amount of free time comparable to a more hard-coded MPR setup, without having to tell each node that it is indeed an MPR. Plus, during testing, we saw a range of throughput, from 700 packets/second to 1700 packets/second and greater (depending on the number of trucks), indicating that each node technically has the capability to handle such a dynamic setup without failing. A double benefit is that, should a node fail in the middle, the platoon algorithm (carried over from project 1) can continue on uninterrupted, without the need to reassign MPR nodes.

With only minor tweaks, we could choose to reduce packet redundancy by passing not only which nodes are likely to have been visited by a packet, but the chance that the packet has reached a node. This would allow the nodes to more intelligently perform their routing operations, but through fears that control packets would occasionally be lost (due to a non-zero drop rate), we prefer to take the safe route that is a bit less efficient in larger platoons but provides various improvements overall and retains acceptable levels of safety.

Results, improvements, and general trends:

1. *Scenario: Trucks start off in range, small number of trucks (~5 trucks):*

It takes a smaller number of trucks, potentially closer together, to difference of a dynamic MPR setup in terms of packets sent. As expected, with trucks 1 through 5 (where 1 is front and 5 is rear), the middle-most truck 3 had a noticeable increase in the number of packets it had to handle: with the MPR setup, it has to handle between 50 to 200 packets more (with between 1500 and 1600 packets total). Similarly, there is a trend that trucks 2 and 4, right outside the centermost truck, saw a decrease in packets with the MPR setup: those nodes saw 200 to 700 fewer packets in general, within a range of 1000 to 1700 packets total. Oddly enough, the performance of the nodes at the extreme ends – nodes 1 and 5 – remained fairly similar, with no remarkable difference. This is acceptable, as the overall number of packets handled must be the same, and the ends understandably had to deal with the lowest amount of routing in the first place.

1. *Scenario: Trucks start off in range, large number of trucks (~10 trucks):*

When we move to a full 10-truck platoon, results become more muddied. At high distances between trucks, the drop rate (percent of packets dropped due to excessive distance) decreased between 1% and 5%, which is an improvement (and reflects nodes being more proactive about transmitting data only if they are sure they are the best choice for connecting to neighbor nodes). At closer distances, the MPR-enabled platoon saw a decrease in the required throughput with a similarly low drop rate.

Trucks start off out of range:

Unfortunately, it was nearly impossible to get repeatable, predictable results when all trucks started out of range. It required more or less all but one truck to be in range, and then they have to join together fairly quickly. Other configurations were left untested for this round, due to success with the 5 truck and 10 truck tests above.

*Secondary thoughts:*

Control (broadcast) packets: With the tweaked algorithm, there exists a higher potential for broadcast packets to reach nodes farther down the chain of the platoon. In the old flooding algorithm, nodes would spread the control packets to their neighbors, and each neighbor would freely re-distribute the packets to anyone in range, and the packets would eventually make it to the end. Now, the initial broadcast still occurs, but each node will ideally only receive a duplicate copy of that broadcast once or twice, and as with all types of packets, the control packets are sent with greater confidence; nodes try to communicate only with close nodes if possible.

Note that during testing, with no forwarding of control packets across the platoon, a single platoon happily split into multiple separate platoons. This would occur across a range of distances, between 50 and 100 meters. Once re-enabled, trucks could be ~99 meters apart before they began to frequently lose synchronization; even then, thanks to the more directional path, the communication efficiently kept the trucks together.

One downside of sorts is that the minimum front-to-back packet transmission time increases in certain scenarios. Communication from truck to truck otherwise remains stable, and this results in a slightly decreased short-range communication time. Plus, the lack of dropped packets means that a packet has a higher chance of reaching the end of the platoon from the front of the platoon.

*What about platooning?*

Thankfully, normal platooning still works as expected, verified in the following graphs:

Old algorithm, 10 trucks,

~99 m starting distance between nodes,

Time versus position:

New algorithm, 10 trucks,

~99 m starting distance between nodes,

Time versus position: