Assignment 2

Experiments

2.1 Convergence study

Compare tick-to-convergence on 10-node clique, line, and random topologies using DV algorithm with the commands below.

Output

```
python3 simulator.py --check_convergence DV file_input --graph_file clique_n
10.graph

rt_algo: DV
input_type: file_input
check_convergence: True
inject_failure: False
graph_file: clique_n10.graph
num_nodes is  10
Graph created from file has 10 nodes, 45 edges
Converged at tick 1

SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

```
python3 simulator.py --check_convergence DV file_input --graph_file line_n10.
graph

rt_algo: DV
input_type: file_input
check_convergence: True
```

```
inject_failure: False
graph_file: line_n10.graph
num_nodes is  10
Graph created from file has 10 nodes, 9 edges
Converged at tick 9
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

```
python3 simulator.py --check_convergence DV rand_input --link_prob 0.4 \\ --
seed 2 --num_routers 10

rt_algo: DV
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 2
num_routers: 10
link_prob: 0.4
Random graph has 10 nodes, 16 edges
Converged at tick 4
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

Commentary

Comparing all the graphs, the graphs that were the fastest to converge were the ones that were very connected with their edges, and the graphs that were the slowest to converge were the ones that were slower to converge were the sparser graphs.

`clique_n10` is a completely connected graph

`line_n10` is a sparse graph in the sense that all nodes are connected in a straight line, at max only connected to two other nodes.

`line_n10` takes longer to converge because it takes more hops for all the nodes to be aware of the nodes they are not directly connected to. The nodes at very opposite ends of the graph won't know of each others' existence until their tables go through *all* the other nodes in the graph

The random is the middle ground since nodes not all nodes are connected to each other but each node still has multiple connections to other nodes, so slower than `clique_n10` but faster than `line_n10`

Output

```
python3 simulator.py --check_convergence DV rand_input --link_prob 0.5 --s
eed 1 --num_routers 10

rt_algo: DV
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 1
num_routers: 10
link_prob: 0.5
Random graph has 10 nodes, 16 edges
Converged at tick 4
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

```
python3 simulator.py --check_convergence DV rand_input --link_prob 0.4 --s
eed 1 --num_routers 100

rt_algo: DV
input_type: rand_input
check_convergence: True
inject_failure: False
```

```
seed: 1
num_routers: 100
link_prob: 0.4
Random graph has 100 nodes, 1974 edges
Converged at tick 7
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

```
python3 simulator.py --check_convergence LS rand_input --link_prob 0.5 --se
ed 1 --num_routers 10

rt_algo: LS
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 1
num_routers: 10
link_prob: 0.5
Random graph has 10 nodes, 16 edges
Converged at tick 3
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

```
python3 simulator.py --check_convergence LS rand_input --link_prob 0.4 --se
ed 1 --num_routers 100

rt_algo: LS
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 1
num_routers: 100
```

```
link_prob: 0.4
Random graph has 100 nodes, 1974 edges
Converged at tick 2
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

Commentary

DV was a lot slower to converge than LS. This is because while reliable flooding for LS requires that not much back and forth has to occur between routers, DV has a lot of back and fourth communication. Routers have to send a new broadcast each time they receive an update, and they might need to broadcast an update several times to the same router as soon as they get new information. Since LS focuses on reliable flooding first and computes distances once everyone has the same information, its a lot quicker and a lot less back and fourth has to happen between routers.

2.2 Inject Router Failure

Compare forwarding tables and routing information of DV and LS algorithms with the inject_failure mode
on the 3-node line topology with the commands below.

Output:

```
python3 simulator.py --inject_failure DV file_input --graph_file line_n3.graph

rt_algo: DV
input_type: file_input
check_convergence: False
inject_failure: True
graph_file: line_n3.graph
num_nodes is  3
Graph created from file has 3 nodes, 2 edges
```

```
Node 2 fails at tick 100
Router 0 forwarding table: {1: 1, 0: 0, 2: 1}
Router 0 distance vector: {1: 1.0, 0: 0, 2: 2.0}
Router 1 forwarding table: {0: 0, 1: 1, 2: 0}
Router 1 distance vector: {0: 1.0, 1: 0, 2: 3.0}
Router 2 forwarding table: {}
Router 2 distance vector: {}
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

```
python3 simulator.py --inject_failure LS file_input --graph_file line_n3.graph

rt_algo: LS
input_type: file_input
check_convergence: False
inject_failure: True
graph_file: line_n3.graph
num_nodes is  3
Graph created from file has 3 nodes, 2 edges
Node 2 fails at tick 100
Router 0 forwarding table: {0: None, 1: 1, 2: None}
Router 0 link state: {0: {1: 1.0}, 1: {0: 1.0}, 2: {1: 1.0}}
Router 1 forwarding table: {1: None, 0: 0, 2: None}
Router 1 link state: {1: {0: 1.0}, 0: {1: 1.0}, 2: {1: 1.0}}
Router 2 forwarding table: {}
Router 2 link state: {}
SUCCESS: Routing and offline algorithm agree on shortest paths between all n
ode pairs
```

Commentary:

The forwarding tables of DV and LS look quite different at the failure that is injected at tick 100. For example, the state of router 0s forwarding table does not have a means of reaching 1 or 2 for LS, but router 0 for DV stores the next hops for itself to 1, 0, and 2. This is similar to the forwarding table for router 2, where DV has paths to all the other nodes, but LS does not. Because we know that router 2 has failed, we know that there should not be a path to it. However, because distance vector has to communicate with its neighbors in order to convey these updates, it is likely that the table did not have a chance to update to reflect that there is no longer a path to 2- it needs time to advertise this change. LS, on the other hand, will be quicker to recognize this failure due the fact that nodes broadcast to all other nodes. It can then run Dijkstra to recompute shortest paths.