Assignment 2 Report

Rohit Chawla, Alan Tran

Convergence Study

Exp 1: python3 simulator.py --check_convergence DV file_input --graph_file clique_n10.graph

Result:

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 node pairs

Observation: This is a fully connected graph with all weights = 1, which is how the distance vector algorithm was able to converge in 1 tick. All the nodes are able to advertise their distance vectors to everyone, and nothing changes, since the shortest paths to each node are just their direct edges, so no more ticks occur.

Exp 2: python3 simulator.py --check_convergence DV file_input --graph_file line_n10.graph

Result:

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

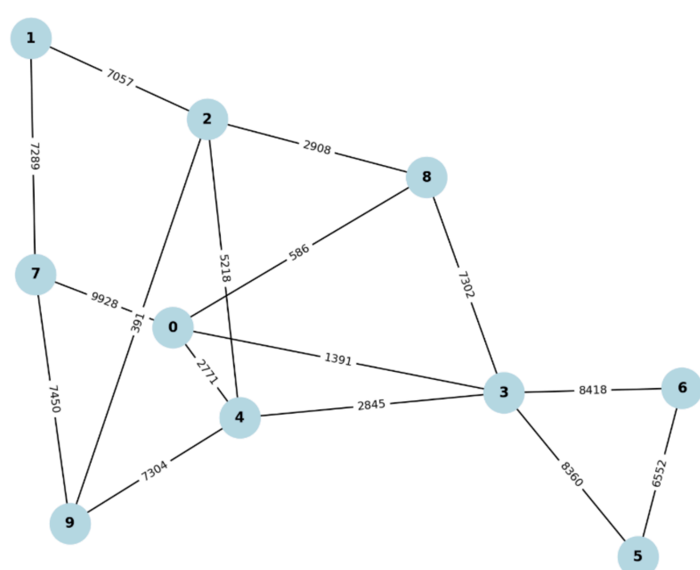
SUCCESS: Routing and offline algorithm agree on shortest paths between all node pairs

Observation: This is a line graph so the distance vector algorithm converges in E ticks, where E is the number of edges in the graph. It would take E broadcasts for the first node in the line to find out about the distance to the last node.

Exp 3: python3 simulator.py --check_convergence DV rand_input --link_prob 0.4 --seed 2 --num_routers 10

Random Graph:

10
0 3 1391
0 4 2771
0 7 9928
0 8 586
1 2 7057
1 7 7289
2 4 5218
2 8 2908
2 9 391
3 4 2845
3 5 8360
3 6 8418
3 8 7302
4 9 7304
5 6 6552
7 9 7450



Result:

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 node pairs

Observation: When running randomly generated graphs, the number of ticks and edges differ based on the seed number. It appears that the more edges there are (more dense the graph is), the faster the graph converges. The graph converges faster than experiment 2's graph but not as fast as experiment 1's graph and number of edges for each experiment follow the same pattern Exp 2 edges < Exp 3 edges < Exp 1. Furthermore, looking at the specific randomly generated graph diagram, the longest shortest path (by weight) has 4 hops, so this results in DV converging in 4 ticks.

Exp 4: python3 simulator.py --check_convergence DV rand_input --link_prob 0.5 --seed 350 --num_routers 10

Result:

rt_algo: DV
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 350
num_routers: 10
link_prob: 0.5
Random graph has 10 nodes, 26 edges
Converged at tick 3

SUCCESS: Routing and offline algorithm agree on shortest paths between all node pairs

python3 simulator.py --check_convergence LS rand_input --link_prob 0.5 --seed 350 --num_routers 10

Result:

rt_algo: LS
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 350
num_routers: 10
link_prob: 0.5
Random graph has 10 nodes, 26 edges
Converged at tick 2

SUCCESS: Routing and offline algorithm agree on shortest paths between all node pairs

Observation: We ran the experiment for 2 different seeds since we wanted to find a graph that produced a different number of ticks to converge between the 2 algorithms. Seed 3 had both algorithms converging with the same number of ticks. However, for seed 350, the LS algorithm converged in 2 ticks, whereas DV converged in 3 ticks. LS might've converged faster if the graph was more dense than sparse.

Exp 5: python3 simulator.py --check_convergence DV rand_input --link_prob 0.4 --seed 1 --num_routers 100

Result:

rt_algo: DV
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 2
num_routers: 100
link_prob: 0.4
Random graph has 100 nodes, 1978 edges
Converged at tick 8

SUCCESS: Routing and offline algorithm agree on shortest paths between all node pairs

python3 simulator.py --check_convergence LS rand_input --link_prob 0.4 --seed 2 --num_routers 100

Result:

rt_algo: LS
input_type: rand_input
check_convergence: True
inject_failure: False
seed: 2
num_routers: 100
link_prob: 0.4
Random graph has 100 nodes, 1978 edges
Converged at tick 3

SUCCESS: Routing and offline algorithm agree on shortest paths between all node pairs

Observation: Since there are more edges due to more nodes present, the ratio of nodes to edges is about 1:20, which means the graph is more dense and connected. LS benefits from more connected graphs, hence why the algorithm converged faster than the DV. DV is slower to converge in dense graphs because it takes more hops for the shortest path info to propagate.

Inject Router Failure

Exp 6: python3 simulator.py --inject_failure DV file_input --graph_file line_n3.graph

Results:

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: 19800.0}
Router 1 forwarding table: {0: 0, 1: 1, 2: 0}
Router 1 distance vector: {0: 1.0, 1: 0, 2: 19799.0}
Router 2 forwarding table: {}
Router 2 distance vector: {}

SUCCESS: Routing and offline algorithm agree on shortest paths between all node pairs

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

Results:

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 node pairs

Observations: DV keeps ghost routes with inflated costs because of the count-to-infinity problem. You can see how router 0 and router 1 ping-pong between each other, thinking that they need to get to router 2 through each other. You can also see how their distance values are 1 off because they keep incrementing. LS immediately removes failed nodes from the forwarding table (ex: node 2); however, it still retains the past LSAs in the link state adjacency matrix.