RongJun Cao

Jesse Goodspeed

Brian Hernandez

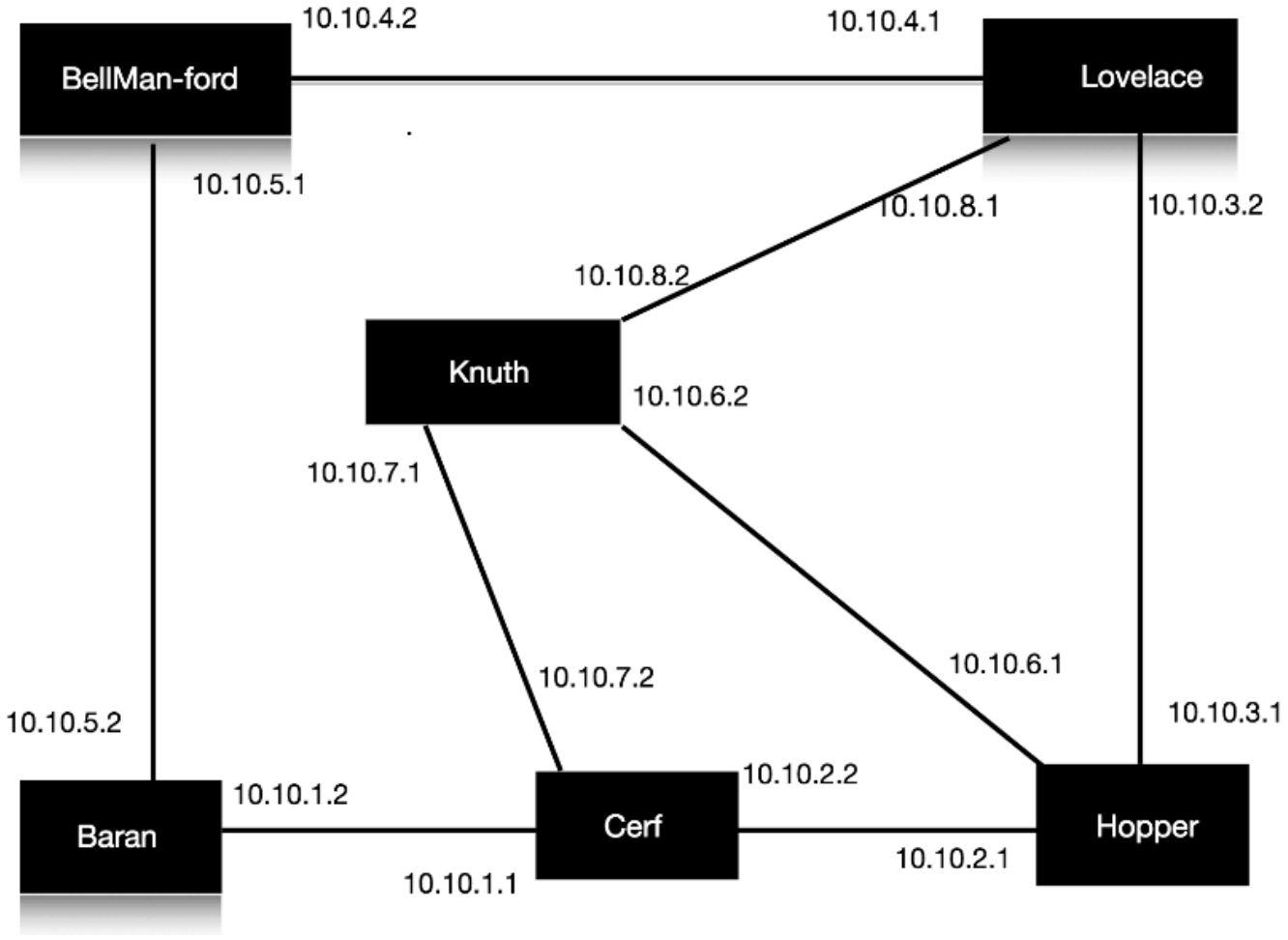
Pearl Leff

CSCI 39954: Final Project

5/27/18

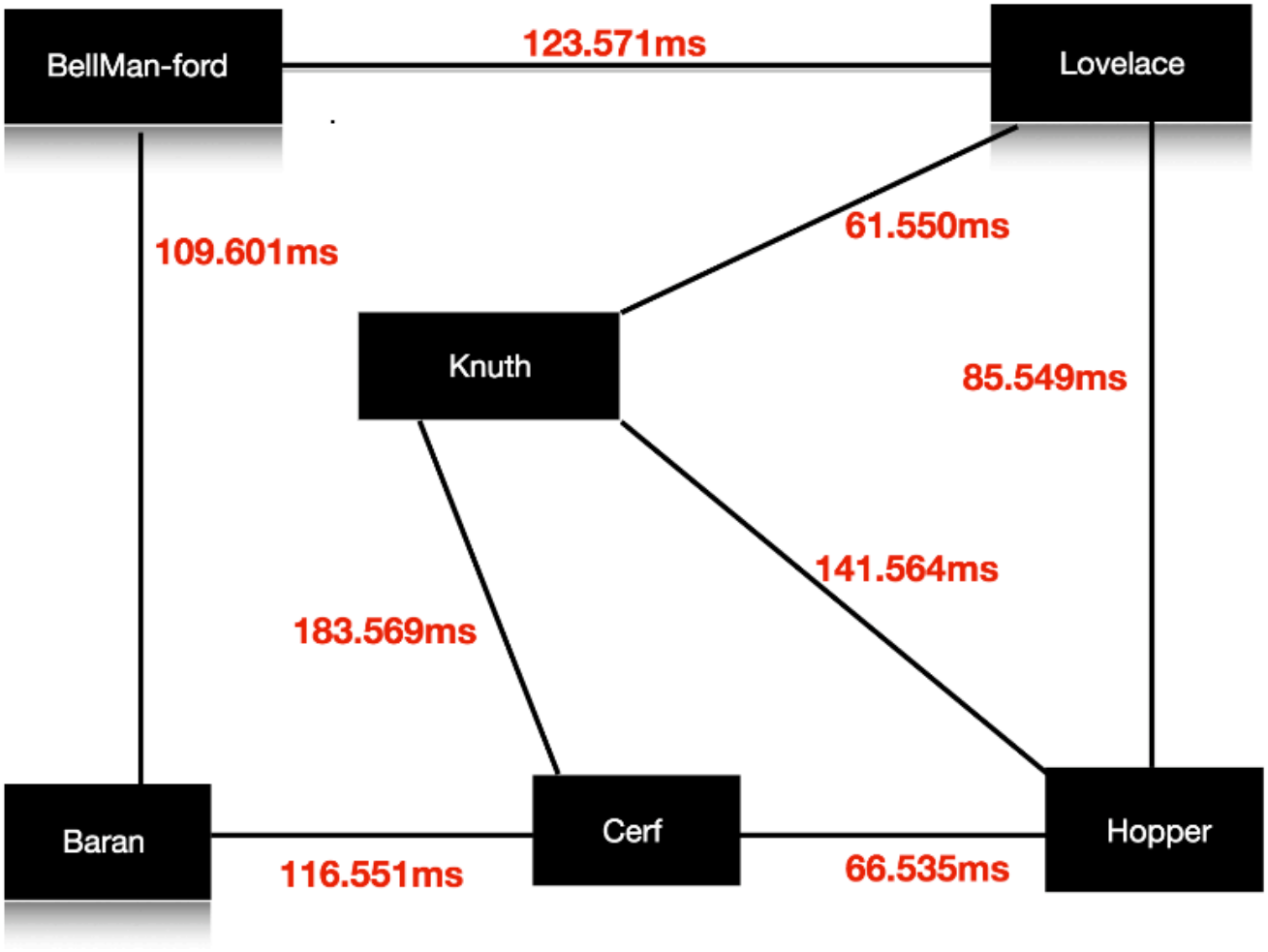
Networking Final Project

Graph of each node with *IP addresses*:



Above is the original topology of the experimental network for this project. Each node is depicted with its name, respective IP addresses, and the edges between them.

Graph of each node with *edge costs*:



The graph above highlights edge weights between each node as the traversal time in milliseconds. Whereas the shortest edge clocked an average cost of 61.550ms between `knuth` and `lovelace`, the longest edge has a cost of 183.569ms between `knuth` and `cerf.`

Output of Bellman-Ford algorithm:

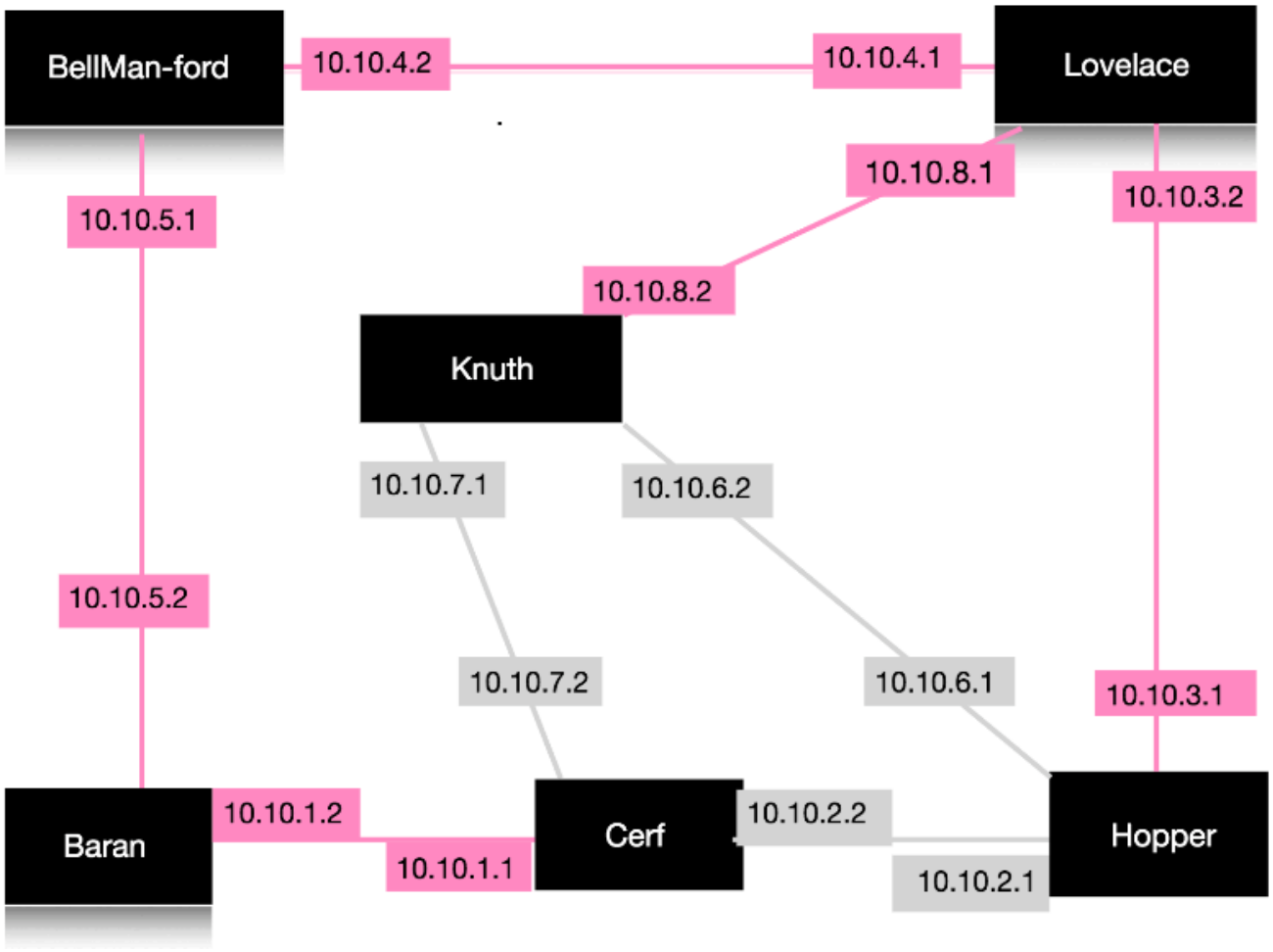
```
--Shortest Path Distance List Iterations--
lovelace: inf knuth: inf baran: inf bellman-ford: 0 cerf: inf hopper: inf
lovelace: 123.571 knuth: inf baran: 109.601 bellman-ford: 0 cerf: inf hopper: inf
lovelace: 123.571 knuth: inf baran: 109.601 bellman-ford: 0 cerf: 226.152 hopper: inf
lovelace: 123.571 knuth: inf baran: 109.601 bellman-ford: 0 cerf: 226.152 hopper: inf
lovelace: 123.571 knuth: 185.121 baran: 109.601 bellman-ford: 0 cerf: 226.152 hopper: 209.12

--Shortest Path Predecessor List--
lovelace preceeded by bellman-ford
knuth preceeded by lovelace
baran preceeded by bellman-ford
bellman-ford preceeded by NULL
cerf preceeded by baran
hopper preceeded by lovelace
```

The Bellman-Ford Algorithm, which is implemented in the `main.cpp` file, outputs a list of iterations, where each iteration reports the cost of the shortest *known* path from the source (`bellman-ford`) to each of the other nodes. Vertices that have not been visited will have a path cost of `inf` for infinite cost. Note that in the final iteration, each node has a computed shortest path. This output serves as a matrix where the rows represent the iterations and the columns are each of the vertices in the graph. The paths are computed using a text file (`bFtopology.txt`) that lists each vertex with its respective neighbors and edge weights.

The "Shortest Path Predecessor List" serves as the graph representation of the optimal routed network. Note that the `bellman-ford` node is preceeded by `NULL` because it is the source node for the purpose of this project.

Graph of Shortest Paths from `bellman-ford`:



The *shortest path* from `bellman-ford` to each of the other nodes are marked in pink, and the edges excluded from this path are marked in grey. The optimal path to each node is no greater than two hops from the source node.

Screenshot of interface script output:

```
rongjunc@bellman-ford:~$ ./destination.bash
[Enter name of destination node: cerf
[Start: Sun May 27 15:27:05 2018
HOST: bellman-ford.finalproject-c Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.10.5.2                 0.0%    10  109.5 109.6 109.5 109.7   0.0
  2.|-- 10.10.1.1                 0.0%    10  226.1 226.1 226.0 226.4   0.0
rongjunc@bellman-ford:~$ ./destination.bash
[Enter name of destination node: hopper
[Start: Sun May 27 15:27:34 2018
HOST: bellman-ford.finalproject-c Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.10.4.1                 0.0%    10  123.7 123.6 123.5 123.7   0.0
  2.|-- 10.10.3.1                 0.0%    10  209.1 209.1 209.0 209.3   0.0
rongjunc@bellman-ford:~$ ./destination.bash
[Enter name of destination node: knuth
[Start: Sun May 27 15:28:07 2018
HOST: bellman-ford.finalproject-c Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.10.4.1                 0.0%    10  123.5 123.6 123.5 123.7   0.0
  2.|-- 10.10.8.2                 0.0%    10  185.1 185.0 184.9 185.1   0.0
rongjunc@bellman-ford:~$ ./destination.bash
[Enter name of destination node: baran
[Start: Sun May 27 15:28:50 2018
HOST: bellman-ford.finalproject-c Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.10.5.2                 0.0%    10  109.6 109.6 109.6 109.7   0.0
rongjunc@bellman-ford:~$ ./destination.bash
[Enter name of destination node: lovelace
[Start: Sun May 27 15:29:11 2018
HOST: bellman-ford.finalproject-c Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.10.4.1                 0.0%    10  123.6 123.6 123.5 123.7   0.0
```

In the above image, the interface script, named `destination.bash` is run. The script prompts the user for a destination node, and then the user enters one of the other nodes in the graph. The script proceeds to output a network trace from `bellman-ford` to the given destination node. The above image captures the script being run for each node other than `bellman-ford`. Please note that the average time from the source to each destination node matches the shortest path cost that is depicted in the *output* of the Bellman-Ford algorithm above.

Instruction for Bellman-Ford Algorithm

As mentioned above, the Bellman-Ford algorithm is implemented in the included `main.cpp` file and must be run with the plain-text file `bFtopology.txt`.

To run the code on the command-line-interface, there are only three steps.

1. On the command line, access the directory where both `main.cpp` and `bFtopology.txt` live.
2. Compile the code by running the following command:

```
g++ -std=c++11 main.cpp
```

3. Execute the code by running the following command:

```
./a.out bFtopology.txt
```

Instruction for interface script

`destination.bash` is a script to print the routing trace from the source to a desired destination node. To use the script, it must be uploaded to the `bellman-ford` node in the project topology after the network has had all appropriate routing rules added. Below are the steps to get this script to work with the GENI project.

1. Upload the script onto the "bellman-ford" node. I recommend this GENI link for reference.
2. SSH onto the `bellman-ford` node.
3. Run the following on the command-line-interface to change the permissions of the script file:

```
chmod 755 destination.bash
```

4. Run the script:

```
./destination.bash
```

5. The user will then be prompted to enter a destination node. Once the destination is inputted the script will print the desired output.