

Go-Back-N Protocol (Java Implementation): This README is for a Java implementation of the Go-Back-N (GBN) protocol. The GBN protocol is a sliding window protocol that allows for reliable data transfer over an unreliable network. It provides an error recovery mechanism by retransmitting lost or damaged packets.

Table of Contents

- File Structure
- Classes
 - Gbnsender
 - Gbnnode
 - Packet
 - AckPacket
- Functions
 - sendPacket
 - sendAckPacket
 - decodeObject
 - decodePacket
 - decodeAckPacket

File Structure

The entire code is written in a single file, `gbnsender.java`, which contains the main class `gbnsender`, along with several inner classes and functions. The initialization code is in `gbnnode.java`

Classes

`gbnsender`

The `gbnsender` class is the main class that contains the implementation of the GBN protocol. The sender maintains a window of packets that have been sent but not yet acknowledged.

`gbnnode`

The `gbnnode` class is the main class that deals with initializing the parameters.

Packet

The `Packet` class represents a packet in the GBN protocol. Each packet has a sequence number and data. This class represents a data packet with the following attributes:

- `seqNum`: Sequence number of the packet
- `data`: Data payload of the packet

AckPacket

The `AckPacket` class represents an acknowledgment packet in the GBN protocol. Each acknowledgment packet has a sequence number. This class represents an acknowledgment (ACK) packet with the following attribute:

- `seqNum`: Sequence number of the packet being acknowledged

Functions

sendPacket

The `sendPacket` function takes a `Packet` object, the receiver's address, receiver's port, and the sender's socket as arguments. It serializes the packet and sends it to the receiver.

sendAckPacket

The `sendAckPacket` function takes an `AckPacket` object, the sender's address, sender's port, and the receiver's socket as arguments. It serializes the acknowledgment packet and sends it to the sender.

decodeObject

The `decodeObject` function takes a `DatagramPacket` object as an argument and returns a deserialized `Packet` object contained in the datagram.

decodeAckPacket

The `decodeAckPacket` function takes a `DatagramPacket` object as an argument and returns a deserialized `AckPacket` object contained in the datagram.

clientListen

This function listens for incoming packets and ACKs from the receiver, and performs the following tasks:

- Decodes and processes received packets and ACKs
- Handles packet loss and retransmissions
- Maintains a count of dropped packets and calculates loss rate

main

This is the main entry point of the program, where the sender and receiver interact. It initializes the required variables, spawns a thread for listening to incoming packets and ACKs, and enters a loop to send messages to the receiver.

The user can input commands in the format "send <message>", and the message will be sent to the receiver using the Go-Back-N protocol.

Usage

To run this program, compile it using a Java compiler, and then execute it with the appropriate arguments for the sending and receiving ports, window size, drop rate, and drop probability:

```
java gbnnode <sendingPort> <receivingPort> <>windowSize> [optional: -d <dropPackets>] [optional: -p <dropProbability>]
```

For example:

```
java gbnnode 5000 5001 4 -d 5 -p 0.1
```

This will run the program with a sending port of 5000, a receiving port of 5001, a window size of 4, a drop rate of 5, and a drop probability of 0.1.

When the program is running, enter commands in the format "send <message>" to send messages to the receiver using the Go-Back-N protocol.

Distance Vector Protocol (Java Implementation): This README is for a Java implementation of the Distance Vector Protocol (DVP) algorithm. The DVP algorithm is a dynamic routing protocol that calculates the best paths for sending packets between network nodes. It uses the Bellman-Ford algorithm to compute the shortest path and relies on periodic exchange of routing tables between neighboring nodes to update the routing information.

Table of Contents

- File Structure
- Classes
 - dvnode
 - Packet
 - Tuple
- Functions
 - sendPacket
 - decodePacket
 - clientListen
 - clientMode
- Usage

File Structure

The entire code is written in a single file, `d vnode.java`, which contains the main class `d vnode`, along with several inner classes and functions.

Classes

`d vnode`

The `d vnode` class is the main class that contains the implementation of the Distance Vector Protocol. It maintains the local routing table, a set of visited ports, and a set of neighboring ports.

Packet

The `Packet` class represents a packet of data that is sent between nodes. It contains a routing table and the source port of the packet.

Tuple

The `Tuple` class is a helper class that represents an entry in the routing table. It contains the next hop port and the loss rate for that route.

Functions

sendPacket

`sendPacket(Packet packet, InetAddress receiverAddress, int receiverPort, DatagramSocket socket)` is a function that sends a packet to the specified receiver address and port using the provided socket.

decodePacket

`decodePacket(DatagramPacket datagram)` is a function that decodes a received datagram packet and returns the corresponding `Packet` object.

clientListen

`clientListen(InetAddress address, int sourcePort)` is a function that listens for incoming packets and updates the local routing table accordingly. It also prints the updated routing table and sends it to neighboring nodes if there's a change.

clientMode

`clientMode(int sourcePort, boolean beginSendingPackets)` is a function that sets up the node in client mode. It creates a socket and starts a listener thread to process incoming packets. If `beginSendingPackets` is true, it sends the initial routing table to all neighbors.

Usage

To run the program, compile and execute the `d vnode.java` file using the following command:

```
java d vnode <local-port> <neighbor1-port> <loss-rate-1> <neighbor2-port> <loss-rate-2>
... [last]
```

`<local-port>`: The port number for the current node (must be between 1024 and 65534).

`<neighborX-port>`: The port number for each neighboring node (must be between 1024 and 65534).

`<loss-rate-X>`: The corresponding loss rate for each neighboring node (must be between 0 and 1).

`[last]`: An optional argument that indicates whether the current node is the last one in the network. If present, the current node will start sending packets to its neighbors.

For example, `java d vnode 1111 2222 .1 3333 .5 4444 .2`

CNNode (Java Implementation): CNNode is a Java-based network routing application that uses the Distance Vector Protocol (DVP) to maintain and update routing tables, and Go-Back-N protocol for reliable data transmission. The application simulates a network node and handles packet transmission and routing in a multi-node network environment.

Main features:

- Distance Vector Protocol for routing table maintenance
- Go-Back-N protocol for reliable data transmission
- Packet loss simulation to test routing algorithm robustness
- Multithreading to handle packet sending, receiving, and processing

How to run the application:

1. Compile the Java source code
2. Run the application with the following command-line arguments:

```
java cnnode <local-port> receive <neighbor1-port> <loss-rate-1> <neighbor2-port>  
<loss-rate-2> ... <neighborM-port> <loss-rate-M> send <neighbor(M+1)-port>  
<neighbor(M+2)-port> ... <neighborN-port> [last]
```

- `local-port`: The port number for the local node
- `receive`: Keyword to specify receiving neighbors
- `neighbor1-port, neighbor2-port, ..., neighborM-port`: Port numbers of the receiving neighbors
- `loss-rate-1, loss-rate-2, ..., loss-rate-M`: Packet loss rates for the receiving neighbors
- `send`: Keyword to specify sending neighbors
- `neighbor (M+1) -port, neighbor (M+2) -port, ..., neighborN-port`: Port numbers of the sending neighbors
- `last`: Optional keyword to specify that the current node is the last node to start

Classes and main methods:

- `CNNNode`: Main class with the `main()` method to run the application
- `clientMode()`: Sets up the client mode, initializing the sender and listener
- `clientListen()`: Handles listening for incoming packets and processes them
- `main()`: Sets up the parameters and starts the application

Note:

The code provided assumes that all nodes run on the same machine, using `InetAddress.getLoopbackAddress()` for the IP address. If you want to run the nodes on different machines, you will need to update the IP address accordingly.

Programming Part 1

Message 1

```
mas2545@instance-1:~/project2/p1$ java gbnnode 2222 1111 5 -p 0.1
Enter command (send <message>): send abcdefgh
[Wed May 03 22:38:12 UTC 2023] packet0 a sent
[Wed May 03 22:38:12 UTC 2023] packet0 b sent
[Wed May 03 22:38:12 UTC 2023] packet0 c sent
[Wed May 03 22:38:12 UTC 2023] packet0 d sent
[Wed May 03 22:38:12 UTC 2023] packet0 e sent
[Wed May 03 22:38:12 UTC 2023] ACK0 received, window moves to 1
[Wed May 03 22:38:12 UTC 2023] ACK1 received, window moves to 2
[Wed May 03 22:38:12 UTC 2023] ACK2 received, window moves to 3
[Wed May 03 22:38:12 UTC 2023] ACK3 received, window moves to 4
[Wed May 03 22:38:12 UTC 2023] ACK4 discarded
[Wed May 03 22:38:13 UTC 2023] packet4 e sent
[Wed May 03 22:38:13 UTC 2023] packet4 f sent
[Wed May 03 22:38:13 UTC 2023] packet4 g sent
[Wed May 03 22:38:13 UTC 2023] packet4 h sent
[Wed May 03 22:38:13 UTC 2023] ACK4 received, window moves to 5
[Wed May 03 22:38:13 UTC 2023] Duplicate ACK4 received.
[Wed May 03 22:38:13 UTC 2023] Duplicate ACK4 received.
[Wed May 03 22:38:13 UTC 2023] packet5 f sent
[Wed May 03 22:38:13 UTC 2023] packet5 g sent
[Wed May 03 22:38:13 UTC 2023] packet5 h sent
[Wed May 03 22:38:13 UTC 2023] ACK5 received, window moves to 6
[Wed May 03 22:38:13 UTC 2023] ACK6 received, window moves to 7
[Wed May 03 22:38:13 UTC 2023] ACK7 received, window moves to 8
[Summary] 1/11 total packets discarded, loss rate = 0.09090909
Enter command (send <message>): []
```

```
mas2545@instance-1:~/project2/p1$ java gbnnode 1111 2222 5 -p 0.1
Enter command (send <message>): [Wed May 03 22:38:12 UTC 2023] packet0 a received
[Wed May 03 22:38:12 UTC 2023] ACK0 sent, expecting packet1
[Wed May 03 22:38:12 UTC 2023] packet1 b received
[Wed May 03 22:38:12 UTC 2023] ACK1 sent, expecting packet2
[Wed May 03 22:38:12 UTC 2023] packet2 c received
[Wed May 03 22:38:12 UTC 2023] ACK2 sent, expecting packet3
[Wed May 03 22:38:12 UTC 2023] packet3 d received
[Wed May 03 22:38:12 UTC 2023] ACK3 sent, expecting packet4
[Wed May 03 22:38:12 UTC 2023] packet4 e received
[Wed May 03 22:38:12 UTC 2023] ACK4 sent, expecting packet5
[Wed May 03 22:38:13 UTC 2023] packet4 e received
[Wed May 03 22:38:13 UTC 2023] Received duplicate packet 4 and sent ACK 4
[Wed May 03 22:38:13 UTC 2023] packet5 f discarded
[Wed May 03 22:38:13 UTC 2023] packet6 g received
[Wed May 03 22:38:13 UTC 2023] Received out-of-sequence packet 6 and sent ACK 4
[Wed May 03 22:38:13 UTC 2023] packet7 h received
[Wed May 03 22:38:13 UTC 2023] Received out-of-sequence packet 7 and sent ACK 4
[Wed May 03 22:38:13 UTC 2023] packet5 f received
[Wed May 03 22:38:13 UTC 2023] ACK5 sent, expecting packet6
[Wed May 03 22:38:13 UTC 2023] packet6 g received
[Wed May 03 22:38:13 UTC 2023] ACK6 sent, expecting packet7
[Wed May 03 22:38:13 UTC 2023] packet7 h received
[Wed May 03 22:38:13 UTC 2023] ACK7 sent, expecting packet8
[Summary] 1/12 total packets discarded, loss rate = 0.083333336
[]
```

Message 2

```
mas2545@instance-1:~/project2/p1$ java gbnnode 2222 1111 2 -d 2
Enter command (send <message>): send cat
[Wed May 03 22:43:43 UTC 2023] packet0 c sent
[Wed May 03 22:43:43 UTC 2023] packet0 a sent
[Wed May 03 22:43:43 UTC 2023] ACK0 received, window moves to 1
[Wed May 03 22:43:43 UTC 2023] packet1 a sent
[Wed May 03 22:43:43 UTC 2023] packet1 t sent
[Wed May 03 22:43:43 UTC 2023] ACK1 discarded
[Wed May 03 22:43:43 UTC 2023] packet1 timeout
[Wed May 03 22:43:44 UTC 2023] packet1 a sent
[Wed May 03 22:43:44 UTC 2023] packet1 t sent
[Wed May 03 22:43:44 UTC 2023] ACK1 received, window moves to 2
[Wed May 03 22:43:44 UTC 2023] packet2 t sent
[Wed May 03 22:43:44 UTC 2023] ACK2 discarded
[Wed May 03 22:43:44 UTC 2023] packet2 timeout
[Wed May 03 22:43:45 UTC 2023] packet2 t sent
[Wed May 03 22:43:45 UTC 2023] packet2 timeout
[Wed May 03 22:43:45 UTC 2023] packet2 t sent
[Wed May 03 22:43:45 UTC 2023] ACK2 received, window moves to 3
[Summary] 2/5 total packets discarded, loss rate = 0.4
Enter command (send <message>): []
```

```
mas2545@instance-1:~/project2/p1$ java gbnnode 1111 2222 2 -d 2
Enter command (send <message>): [Wed May 03 22:43:43 UTC 2023] packet0 c received
[Wed May 03 22:43:43 UTC 2023] ACK0 sent, expecting packet1
[Wed May 03 22:43:43 UTC 2023] packet1 a discarded
[Wed May 03 22:43:43 UTC 2023] packet1 a received
[Wed May 03 22:43:43 UTC 2023] ACK1 sent, expecting packet2
[Wed May 03 22:43:43 UTC 2023] packet2 t discarded
[Wed May 03 22:43:44 UTC 2023] packet1 a received
[Wed May 03 22:43:44 UTC 2023] Received duplicate packet 1 and sent ACK 1
[Wed May 03 22:43:44 UTC 2023] packet2 t discarded
[Wed May 03 22:43:44 UTC 2023] packet2 t received
[Wed May 03 22:43:44 UTC 2023] ACK2 sent, expecting packet3
[Wed May 03 22:43:45 UTC 2023] packet2 t discarded
[Wed May 03 22:43:45 UTC 2023] packet2 t received
[Wed May 03 22:43:45 UTC 2023] Received duplicate packet 2 and sent ACK 2
[Summary] 4/9 total packets discarded, loss rate = 0.44444445
[]
```

Message 3

```
mas2545@instance-1:~/project2/p1$ java gbnnode 2222 1111 2
Enter command (send <message>): [Wed May 03 22:48:37 UTC 2023] packet0 m received
[Wed May 03 22:48:37 UTC 2023] ACK0 sent, expecting packet1
[Wed May 03 22:48:37 UTC 2023] packet1 o received
[Wed May 03 22:48:37 UTC 2023] ACK1 sent, expecting packet2
[Wed May 03 22:48:37 UTC 2023] packet2 u received
[Wed May 03 22:48:37 UTC 2023] ACK2 sent, expecting packet3
[Wed May 03 22:48:37 UTC 2023] packet3 s received
[Wed May 03 22:48:37 UTC 2023] ACK3 sent, expecting packet4
[Wed May 03 22:48:37 UTC 2023] packet4 e received
[Wed May 03 22:48:37 UTC 2023] ACK4 sent, expecting packet5
[Summary] 0/5 total packets discarded, loss rate = 0.0
send cow
[Wed May 03 22:48:43 UTC 2023] packet0 c sent
[Wed May 03 22:48:43 UTC 2023] packet0 o sent
[Wed May 03 22:48:43 UTC 2023] ACK0 received, window moves to 1
[Wed May 03 22:48:43 UTC 2023] ACK1 received, window moves to 2
[Wed May 03 22:48:44 UTC 2023] packet2 w sent
[Wed May 03 22:48:44 UTC 2023] ACK2 received, window moves to 3
[Summary] 0/3 total packets discarded, loss rate = 0.0
Enter command (send <message>): []
```

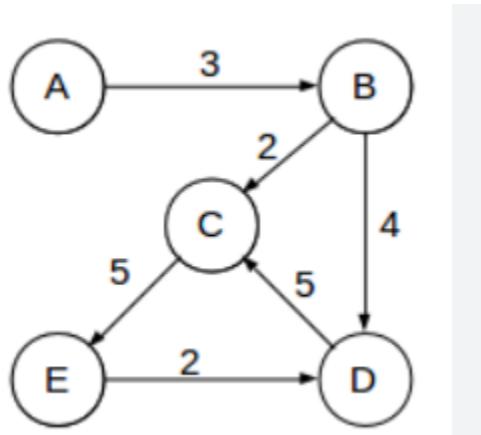
```

mas2545@instance-1:~/project2/p1$ java gbnnode 1111 2222 5
Enter command (send <message>): send mouse
[Wed May 03 22:48:37 UTC 2023] packet0 m sent
[Wed May 03 22:48:37 UTC 2023] packet0 o sent
[Wed May 03 22:48:37 UTC 2023] packet0 u sent
[Wed May 03 22:48:37 UTC 2023] packet0 s sent
[Wed May 03 22:48:37 UTC 2023] packet0 e sent
[Wed May 03 22:48:37 UTC 2023] ACK0 received, window moves to 1
[Wed May 03 22:48:37 UTC 2023] ACK1 received, window moves to 2
[Wed May 03 22:48:37 UTC 2023] ACK2 received, window moves to 3
[Wed May 03 22:48:37 UTC 2023] ACK3 received, window moves to 4
[Wed May 03 22:48:37 UTC 2023] ACK4 received, window moves to 5
[Summary] 0/5 total packets discarded, loss rate = 0.0
Enter command (send <message>): [Wed May 03 22:48:43 UTC 2023] packet0 c received
[Wed May 03 22:48:43 UTC 2023] ACK0 sent, expecting packet1
[Wed May 03 22:48:43 UTC 2023] packet1 o received
[Wed May 03 22:48:43 UTC 2023] ACK1 sent, expecting packet2
[Wed May 03 22:48:44 UTC 2023] packet2 w received
[Wed May 03 22:48:44 UTC 2023] ACK2 sent, expecting packet3
[Summary] 0/3 total packets discarded, loss rate = 0.0

```

Programming Part 2

Graph 1



Notation for ports: 1111 = 1, 2222 = 2 etc.

```

mas2545@instance-1:~/project2/p2$ java dvnode 1111 2222 .3
[Wed May 03 22:28:00 UTC 2023] Node 1111 Routing Table
- (0.90000004) -> Node 5555; Next hop -> Node 2222
- (0.5) -> Node 3333; Next hop -> Node 2222
- (0.70000005) -> Node 4444; Next hop -> Node 2222
- (0.3) -> Node 2222

```

```
Last login: Wed May  3 22:26:27 2023 from 172.253.216.57
mas2545@instance-1:~$ cd project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 2222 1111 .3 3333 .2 4444 .4
[Wed May 03 22:27:59 UTC 2023] Node 2222 Routing Table
- (0.6) -> Node 5555; Next hop -> Node 4444
- (0.2) -> Node 3333
- (0.3) -> Node 1111
- (0.4) -> Node 4444
```

```
mas2545@instance-1:~$ cd project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 3333 2222 .2 4444 .5 5555 .5
[Wed May 03 22:27:59 UTC 2023] Node 3333 Routing Table
- (0.5) -> Node 5555
- (0.5) -> Node 4444
- (0.2) -> Node 2222

[Wed May 03 22:28:00 UTC 2023] Node 3333 Routing Table
- (0.5) -> Node 5555
- (0.5) -> Node 1111; Next hop -> Node 2222
- (0.5) -> Node 4444
- (0.2) -> Node 2222
```

```
mas2545@instance-1:~/project2/p2$ java dvnode 4444 2222 .4 3333 .5 5555 .2
[Wed May 03 22:27:59 UTC 2023] Node 4444 Routing Table
- (0.2) -> Node 5555
- (0.5) -> Node 3333
- (0.4) -> Node 2222

[Wed May 03 22:28:00 UTC 2023] Node 4444 Routing Table
- (0.2) -> Node 5555
- (0.5) -> Node 3333
- (1.2) -> Node 1111; Next hop -> Node 5555
- (0.4) -> Node 2222

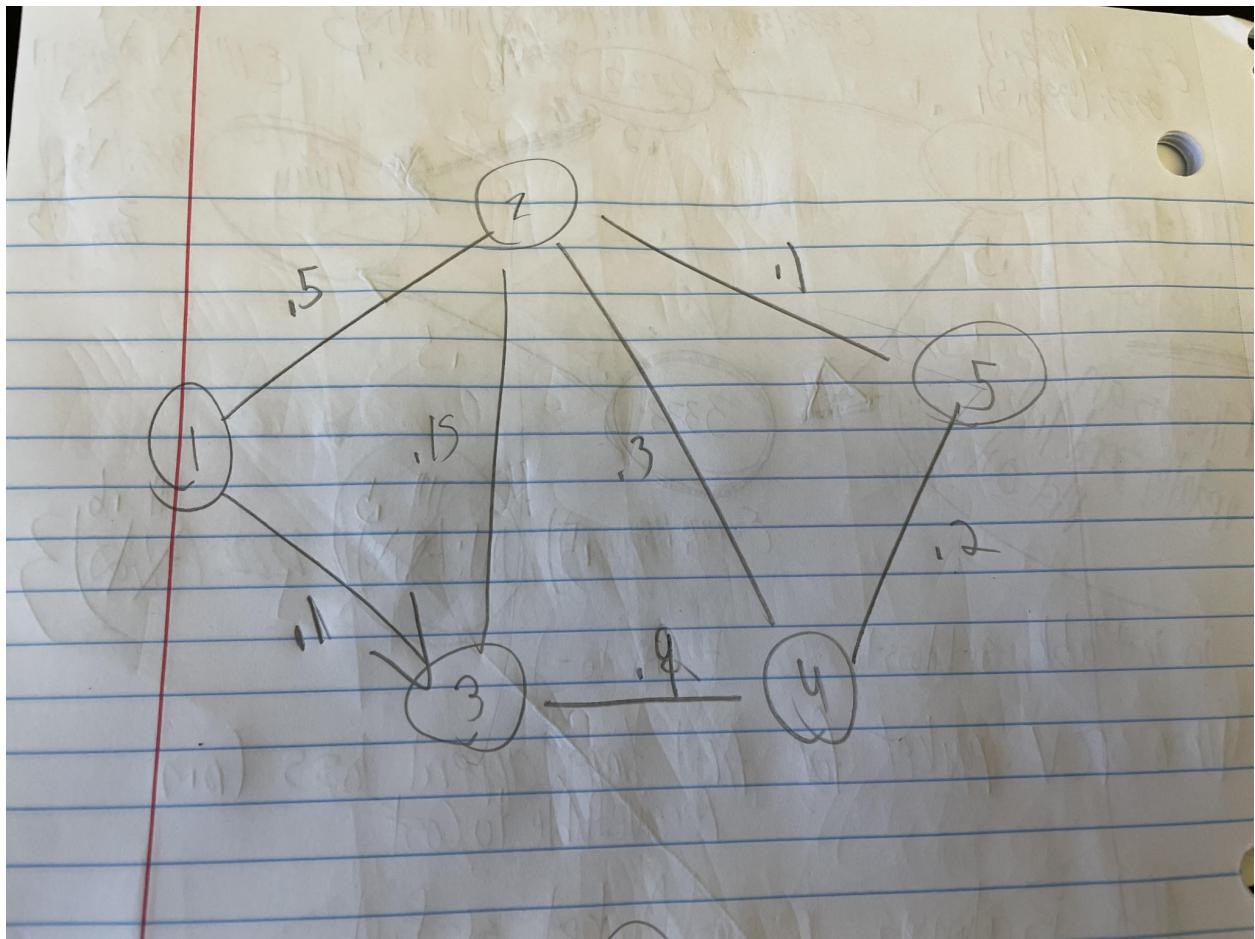
[Wed May 03 22:28:00 UTC 2023] Node 4444 Routing Table
- (0.2) -> Node 5555
- (0.5) -> Node 3333
- (0.70000005) -> Node 1111; Next hop -> Node 2222
- (0.4) -> Node 2222
```

```
Last login: Wed May  3 22:26:55 2023 from 35.235.240.2
mas2545@instance-1:~$ cd project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 5555 3333 .5 4444 .2 last
[Wed May 03 22:27:59 UTC 2023] Node 5555 Routing Table
- (0.5) -> Node 3333
- (0.2) -> Node 4444
- (0.6) -> Node 2222; Next hop -> Node 4444

[Wed May 03 22:28:00 UTC 2023] Node 5555 Routing Table
- (0.5) -> Node 3333
- (1.0) -> Node 1111; Next hop -> Node 3333
- (0.2) -> Node 4444
- (0.6) -> Node 2222; Next hop -> Node 4444

[Wed May 03 22:28:00 UTC 2023] Node 5555 Routing Table
- (0.5) -> Node 3333
- (0.90000004) -> Node 1111; Next hop -> Node 4444
- (0.2) -> Node 4444
- (0.6) -> Node 2222; Next hop -> Node 4444
```

Graph 2



Notation for ports: 1111 = 1, 2222 = 2 etc.

```
Last login: Tue May  2 22:00:50 2023 from 35.235.244.33
mas2545@instance-1:~$ cd project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 1111 2222 .5 3333 .1
[Wed May 03 00:47:18 UTC 2023] Node 1111 Routing Table
- (0.6) -> Node 5555; Next hop -> Node 2222
- (0.1) -> Node 3333
- (0.8) -> Node 4444; Next hop -> Node 2222
- (0.5) -> Node 2222

[Wed May 03 00:47:19 UTC 2023] Node 1111 Routing Table
- (0.6) -> Node 5555; Next hop -> Node 2222
- (0.1) -> Node 3333
- (0.8) -> Node 4444; Next hop -> Node 2222
- (0.25) -> Node 2222; Next hop -> Node 3333

[Wed May 03 00:47:19 UTC 2023] Node 1111 Routing Table
- (0.35) -> Node 5555; Next hop -> Node 3333
- (0.1) -> Node 3333
- (0.55) -> Node 4444; Next hop -> Node 3333
- (0.25) -> Node 2222; Next hop -> Node 3333
```

```
Last login: Wed May  3 00:45:44 2023 from 35.235.244.32
mas2545@instance-1:~$ cd project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 2222 1111 .5 3333 .15 4444 .3 5555 .1
[Wed May 03 00:47:18 UTC 2023] Node 2222 Routing Table
- (0.1) -> Node 5555
- (0.15) -> Node 3333
- (0.5) -> Node 1111
- (0.3) -> Node 4444

[Wed May 03 00:47:19 UTC 2023] Node 2222 Routing Table
- (0.1) -> Node 5555
- (0.15) -> Node 3333
- (0.25) -> Node 1111; Next hop -> Node 3333
- (0.3) -> Node 4444
```

```
Last login: Wed May  3 00:45:45 2023 from 35.235.244.32
mas2545@instance-1:~$ cd project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 3333 1111 .1 2222 .15 4444 .9
[Wed May 03 00:47:18 UTC 2023] Node 3333 Routing Table
- (1.1) -> Node 5555; Next hop -> Node 4444
- (0.1) -> Node 1111
- (0.9) -> Node 4444
- (0.15) -> Node 2222

[Wed May 03 00:47:19 UTC 2023] Node 3333 Routing Table
- (0.25) -> Node 5555; Next hop -> Node 2222
- (0.1) -> Node 1111
- (0.45000002) -> Node 4444; Next hop -> Node 2222
- (0.15) -> Node 2222
```

```
Last login: Wed May  3 00:45:40 2023 from 172.253.216.57
mas2545@instance-1:~$ cd project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 4444 3333 .9 2222 .3 5555 .2
[Wed May 03 00:47:18 UTC 2023] Node 4444 Routing Table
- (0.2) -> Node 5555
- (0.9) -> Node 3333
- (0.3) -> Node 2222

[Wed May 03 00:47:18 UTC 2023] Node 4444 Routing Table
- (0.2) -> Node 5555
- (0.45000002) -> Node 3333; Next hop -> Node 2222
- (0.8) -> Node 1111; Next hop -> Node 2222
- (0.3) -> Node 2222

[Wed May 03 00:47:19 UTC 2023] Node 4444 Routing Table
- (0.2) -> Node 5555
- (0.45) -> Node 3333; Next hop -> Node 5555
- (0.8) -> Node 1111; Next hop -> Node 2222
- (0.3) -> Node 2222

[Wed May 03 00:47:19 UTC 2023] Node 4444 Routing Table
- (0.2) -> Node 5555
- (0.45) -> Node 3333; Next hop -> Node 5555
- (0.55) -> Node 1111; Next hop -> Node 5555
- (0.3) -> Node 2222
```

```
Last login: Wed May  3 00:46:02 2023 from 172.253.216.57
mas2545@instance-1:~/project2/p2/
mas2545@instance-1:~/project2/p2$ java dvnode 5555 2222 .1 4444 .2 last
[Wed May 03 00:47:18 UTC 2023] Node 5555 Routing Table
- (0.25) -> Node 3333; Next hop -> Node 2222
- (0.6) -> Node 1111; Next hop -> Node 2222
- (0.2) -> Node 4444
- (0.1) -> Node 2222

[Wed May 03 00:47:19 UTC 2023] Node 5555 Routing Table
- (0.25) -> Node 3333; Next hop -> Node 2222
- (0.35) -> Node 1111; Next hop -> Node 2222
- (0.2) -> Node 4444
- (0.1) -> Node 2222

^Cmas2545@instance-1:~/project2/p2$ []
```

Graph 3

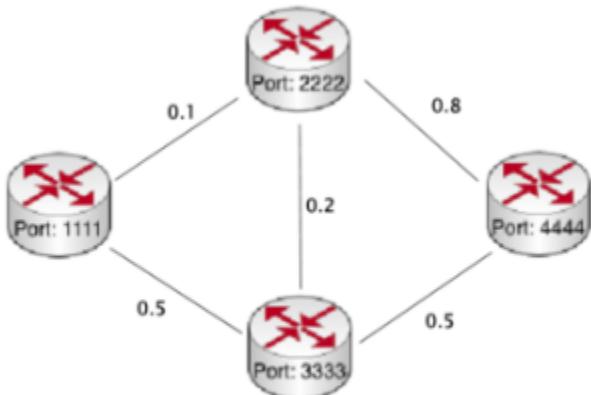


Figure 3: Network Configuration Example

```
^Cmas2545@instance-1:~/project2/p2$ java dvnode 1111 2222 .1 3333 .5
[Wed May 03 22:31:32 UTC 2023] Node 1111 Routing Table
- (0.3) -> Node 3333; Next hop -> Node 2222
- (0.90000004) -> Node 4444; Next hop -> Node 2222
- (0.1) -> Node 2222

[Wed May 03 22:31:33 UTC 2023] Node 1111 Routing Table
- (0.3) -> Node 3333; Next hop -> Node 2222
- (0.8) -> Node 4444; Next hop -> Node 2222
- (0.1) -> Node 2222
```

```
^Cmas2545@instance-1:~/project2/p2$ java dvnode 2222 1111 .1 3333 .2 4444 .8  
[Wed May 03 22:31:32 UTC 2023] Node 2222 Routing Table  
- (0.2) -> Node 3333  
- (0.1) -> Node 1111  
- (0.8) -> Node 4444  
  
[Wed May 03 22:31:32 UTC 2023] Node 2222 Routing Table  
- (0.2) -> Node 3333  
- (0.1) -> Node 1111  
- (0.7) -> Node 4444; Next hop -> Node 3333
```

```
^Cmas2545@instance-1:~/project2/p2$ java dvnode 3333 1111 .5 2222 .2 4444 .5  
[Wed May 03 22:31:32 UTC 2023] Node 3333 Routing Table  
- (0.5) -> Node 1111  
- (0.5) -> Node 4444  
- (0.2) -> Node 2222  
  
[Wed May 03 22:31:32 UTC 2023] Node 3333 Routing Table  
- (0.3) -> Node 1111; Next hop -> Node 2222  
- (0.5) -> Node 4444  
- (0.2) -> Node 2222
```

```
^Cmas2545@instance-1:~/project2/p2$ java dvnode 4444 3333 .5 2222 .8 last  
[Wed May 03 22:31:32 UTC 2023] Node 4444 Routing Table  
- (0.5) -> Node 3333  
- (0.90000004) -> Node 1111; Next hop -> Node 2222  
- (0.8) -> Node 2222  
  
[Wed May 03 22:31:33 UTC 2023] Node 4444 Routing Table  
- (0.5) -> Node 3333  
- (0.90000004) -> Node 1111; Next hop -> Node 2222  
- (0.7) -> Node 2222; Next hop -> Node 3333  
  
[Wed May 03 22:31:33 UTC 2023] Node 4444 Routing Table  
- (0.5) -> Node 3333  
- (0.8) -> Node 1111; Next hop -> Node 3333  
- (0.7) -> Node 2222; Next hop -> Node 3333
```

Programming Part 3

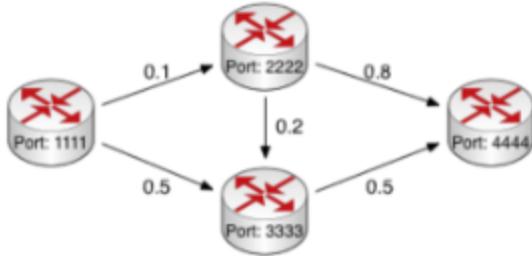


Figure 4: Network Configuration Example

```

[Sun May 21 21:24:56 UTC 2023] Link to 3333: 363 packets sent, 107 packets lost, loss rate 0.51
[Sun May 21 21:24:56 UTC 2023] Link to 2222: 194 packets sent, 19 packets lost, loss rate 0.09
[Sun May 21 21:24:57 UTC 2023] Link to 3333: 365 packets sent, 189 packets lost, loss rate 0.51
[Sun May 21 21:24:57 UTC 2023] Link to 2222: 194 packets sent, 19 packets lost, loss rate 0.09
[Sun May 21 21:24:58 UTC 2023] Link to 3333: 366 packets sent, 190 packets lost, loss rate 0.51
[Sun May 21 21:24:58 UTC 2023] Link to 2222: 195 packets sent, 19 packets lost, loss rate 0.09
[Sun May 21 21:24:59 UTC 2023] Link to 3333: 367 packets sent, 190 packets lost, loss rate 0.51
[Sun May 21 21:24:59 UTC 2023] Link to 2222: 196 packets sent, 19 packets lost, loss rate 0.09
[Sun May 21 21:25:00 UTC 2023] Link to 3333: 368 packets sent, 190 packets lost, loss rate 0.51
[Sun May 21 21:25:00 UTC 2023] Link to 2222: 197 packets sent, 19 packets lost, loss rate 0.09
[Sun May 21 21:25:01 UTC 2023] Node 1111 Routing Table
- (0.26) -> Node 3333; Next hop -> Node 2222
- (0.76) -> Node 4444; Next hop -> Node 2222
- (0.09) -> Node 2222

```

```

[Sun May 21 21:25:02 UTC 2023] Link to 3333: 106 packets sent, 19 packets lost, loss rate 0.17
[Sun May 21 21:25:02 UTC 2023] Link to 4444: 465 packets sent, 378 packets lost, loss rate 0.81
[Sun May 21 21:25:03 UTC 2023] Link to 3333: 106 packets sent, 19 packets lost, loss rate 0.17
[Sun May 21 21:25:03 UTC 2023] Link to 4444: 467 packets sent, 380 packets lost, loss rate 0.81
[Sun May 21 21:25:04 UTC 2023] Link to 3333: 106 packets sent, 19 packets lost, loss rate 0.17
[Sun May 21 21:25:04 UTC 2023] Link to 4444: 469 packets sent, 382 packets lost, loss rate 0.81
[Sun May 21 21:25:05 UTC 2023] Link to 3333: 107 packets sent, 19 packets lost, loss rate 0.17
[Sun May 21 21:25:05 UTC 2023] Link to 4444: 470 packets sent, 383 packets lost, loss rate 0.81
[Sun May 21 21:25:06 UTC 2023] Link to 3333: 107 packets sent, 19 packets lost, loss rate 0.17
[Sun May 21 21:25:06 UTC 2023] Link to 4444: 472 packets sent, 384 packets lost, loss rate 0.81
[Sun May 21 21:25:06 UTC 2023] Node 2222 Routing Table
- (0.17) -> Node 3333
- (0.09) -> Node 1111
- (0.67) -> Node 4444; Next hop -> Node 3333

```

```

[Sun May 21 21:25:02 UTC 2023] Link to 4444: 571 packets sent, 288 packets lost, loss rate 0.5
[Sun May 21 21:25:03 UTC 2023] Link to 4444: 573 packets sent, 290 packets lost, loss rate 0.5
[Sun May 21 21:25:04 UTC 2023] Link to 4444: 575 packets sent, 291 packets lost, loss rate 0.5
[Sun May 21 21:25:05 UTC 2023] Link to 4444: 577 packets sent, 292 packets lost, loss rate 0.5
[Sun May 21 21:25:06 UTC 2023] Link to 4444: 579 packets sent, 294 packets lost, loss rate 0.5
[Sun May 21 21:25:06 UTC 2023] Node 3333 Routing Table
- (0.26) -> Node 1111; Next hop -> Node 2222
- (0.5) -> Node 4444
- (0.17) -> Node 2222

```

```
[Sun May 21 21:24:46 UTC 2023] Node 4444 Routing Table
- (0.49) -> Node 3333
- (0.74) -> Node 1111; Next hop -> Node 3333
- (0.66) -> Node 2222; Next hop -> Node 3333

[Sun May 21 21:24:51 UTC 2023] Node 4444 Routing Table
- (0.5) -> Node 3333
- (0.74) -> Node 1111; Next hop -> Node 3333
- (0.67) -> Node 2222; Next hop -> Node 3333

[Sun May 21 21:24:56 UTC 2023] Node 4444 Routing Table
- (0.5) -> Node 3333
- (0.75) -> Node 1111; Next hop -> Node 3333
- (0.65999997) -> Node 2222; Next hop -> Node 3333

[Sun May 21 21:25:01 UTC 2023] Node 4444 Routing Table
- (0.5) -> Node 3333
- (0.77) -> Node 1111; Next hop -> Node 3333
- (0.68) -> Node 2222; Next hop -> Node 3333
```