2015 Spring CS118 Project 2
Simple DV Router

Team Info:

Yichen Pan 604152505
Haonan Zhou 104180068
Yueyang Mi 504169832

Implementation Description:

- Structures:
    - Each router maintains the following information as a variable (some subtle members are not listed here):

Data Member	**Descripttion**
char id	ID of the router (A, B, C, D, E, F)
int port	port number of the router
udp::socket socket	Socket descriptor of the router. Definition of this class is provided by the BOOST library. This variable is needed when invoking asynchronous sending methods of BOOST.
boost::asio::deadline_timer* periodic_timer	A pointer to a timer for sending periodic DV updates to neighbors.
boost::asio::deadline_timer* timeout_timer	A pointer to a timer for detecting timeouts when the router does not hear from the neighbor for a long time.
char data_buffer[MAX_LENGTH]	Buffer for storage of data before the router sends or after it receives a message.
FTEntry FT[6]	Forwarding table of the router.
int DV[6][6]	Distance vector maintained by each router.

Some key methods of the DVRouter class is listed and explained in the chart below (all arguments are omitted):

Method	**Description**
handle_receive()	Invoked when the socket receives a packet. This function handles each packet differently, based on the packet type.

handle_data_pkt() handle_control_pkt() handle_rrep_pkt()	These three methods are invoked by handle_received(), based on the received packet type.
send_to()	This method uses the asynchronous send utility from the BOOST library to send contents of the data buffer to the destination port
prepare_dv_send() prepare_rrep_send()	Called before an DV update or RREP message is sent. These two methods format the message and store it in the data buffer.
update()	Update DV and Forwarding Table based on the received DV update from a neighbor.

- Forwarding Table
  - 6 Forward Table entries, one for each neighbor.
  - FT[i].cost stores the direct cost to Router i. (infinity if not a neighbor)
  - Each entry FT[i] also records Next hop name on the shortest path to i, and the port number of Next hop(0 if not neighbor).
  - FT[i]'s next hop is updated if a new path is found by the algorithm.
  - If the destination is a neighbor and is not alive, FT[i].alive is false.
    Else, FT[i].alive is true.

- Distance Vector Table
  - A 6*6 Table recording the known shortest path cost for every combination of source and destination.
  - DV[x][y] records the shortest distance known from router x to y.
  - Every 5 seconds router x broadcasts (by sending control message) its own row of DV, DV[x] to all its neighbors. Receiver Router y would update its own DV according to the row of DV received.

- Message format:
  - First line of each message starts with "Type: " followed by the message type. A message can be of type "Data", "Control", or "RREP".
  - Second line of each message starts with "Src: ", followed by the ID of the source router that generated the message. If it is a data or RREP message, there is a string that indicates the destination router in the format of "Dest: " followed by the ID of router where the message is being sent to.
  - Third line of data or RREP messages contains the path that the message has traveled. It is in the format of "Path: " followed by the IDs of routers that the message has visited. In a control message, however, such a path would be unnecessary, since it consists of only two nodes and can be easily identified with source and destination on the second line.
  - In Data and RREP messages, there is an extra header line that identifies the packet ID they are referring to.
  - After that comes the actual content of the message. In a data message, this would be the data sent by the user. In a control message, this part contains a single line of the sender's updated DV. Numbers are separated by a single comma with no space.

- A typical control message looks like the following:

```
Type: Control
Src ID: B
3,0,2,4,2,1
```

- A typical data message looks like the following:

```
Type: Data
Src ID: A, Dest ID: D
Path: ABFCD
Pkt ID: 2
Aha, this is only a test file
Aha, this is only a test file
Aha, this is only a test file
Aha, this is only a test file
Aha, this is only a test file
```

- A typical RREP message looks like the following:

```
Type: RREP
Src ID: D, Dest ID: A
Path: DCFB
Pkt ID: 2
```

Difficulties:

- As we didn't fully understand the shortest path distance and the direct cost to neighbor at first, our DV's entries became smaller than expected. This is solved by storing shortest path cost and direct distance to neighbor separately in DV and FT.
- Setting up Boost library was a big trouble for us. It took us a lot of time to define the path to link the boost library.
- The loop routing problem:
  Non-neighbor routers do not learn about a router's death.

  Assuming the following topology:
  A - B - C
  After DV converges, we kill router C.
  B didn't get contacted from C for a while, timeout, noticed that C is dead.
  However, A tells B that A can reach C via some path, but A really cannot because C died.
  B would then think that C is reachable via A, updating C to be alive.
  Both A and B think that C is reachable.

  Solution: Split Horizon
  Basically, a router does not advertise learned routes back to who told him about them.
  In the easy example, (A would not tell B how to reach C) since (B told A how to reach C).
  Then, both A and B knows that C is unreachable.

DV and Forwarding Table after Converging:

```
+---Forwarding Table of Router A---+
A | C=0, outport=10000, destport=0, alive
B | C=3, outport=10000, destport=10001, alive
B | C=-, outport=10000, destport=10001, alive
B | C=-, outport=10000, destport=10001, alive
E | C=1, outport=10000, destport=10004, alive
B | C=-, outport=10000, destport=10001, alive
+--------------------------------+

+------DV Table of Router A---+
    A | B | C | D | E | F |
A | 0 | 3 | 5 | 7 | 1 | 4 |
B | - | 0 | 2 | 4 | 2 | 1 |
C | - | - | - | - | - | - |
D | - | - | - | - | - | - |
E | - | 2 | 4 | 6 | 0 | 3 |
F | - | - | - | - | - | - |
+---------------------------+
```

```
+---Forwarding Table of Router B---+
A | C=3, outport=10001, destport=10000, alive
B | C=0, outport=10001, destport=0, alive
F | C=3, outport=10001, destport=10005, alive
F | C=-, outport=10001, destport=10005, alive
E | C=2, outport=10001, destport=10004, alive
F | C=1, outport=10001, destport=10005, alive
+--------------------------------+

+------DV Table of Router B---+
    A | B | C | D | E | F |
A | 0 | - | - | - | 1 | - |
B | 3 | 0 | 2 | 4 | 2 | 1 |
C | 5 | 2 | 0 | 2 | 4 | 1 |
D | - | - | - | - | - | - |
E | 1 | - | - | - | 0 | - |
F | - | - | 1 | 3 | - | 0 |
+---------------------------+
```

```
+---Forwarding Table of Router C---+
F | C=-, outport=10002, destport=10005, alive
F | C=3, outport=10002, destport=10005, alive
C | C=0, outport=10002, destport=0, alive
D | C=2, outport=10002, destport=10003, alive
F | C=-, outport=10002, destport=10005, alive
F | C=1, outport=10002, destport=10005, alive
+--------------------------------+

+------DV Table of Router C---+
    A | B | C | D | E | F |
A | - | - | - | - | - | - |
B | 3 | 0 | 2 | 4 | 2 | 1 |
C | 5 | 2 | 0 | 2 | 4 | 1 |
D | - | - | - | 0 | - | - |
E | - | - | - | - | - | - |
F | 4 | 1 | - | - | 3 | 0 |
+---------------------------+
```

```
+---Forwarding Table of Router D---+
C | C=-, outport=10003, destport=10002, alive
C | C=-, outport=10003, destport=10002, alive
C | C=2, outport=10003, destport=10002, alive
D | C=0, outport=10003, destport=0, alive
C | C=-, outport=10003, destport=10002, alive
C | C=3, outport=10003, destport=10002, alive
+--------------------------------+

+------DV Table of Router D---+
    A | B | C | D | E | F |
A | - | - | - | - | - | - |
B | - | - | - | - | - | - |
C | 5 | 2 | 0 | - | 4 | 1 |
D | 7 | 4 | 2 | 0 | 6 | 3 |
E | - | - | - | - | - | - |
F | 4 | 1 | 1 | 3 | 3 | 0 |
+---------------------------+
```

```
+---Forwarding Table of Router E---+
A | C=1, outport=10004, destport=10000, alive
B | C=2, outport=10004, destport=10001, alive
B | C=-, outport=10004, destport=10001, alive
B | C=-, outport=10004, destport=10001, alive
E | C=0, outport=10004, destport=0, alive
B | C=3, outport=10004, destport=10001, alive
+--------------------------------+

+------DV Table of Router E---+
    A | B | C | D | E | F |
A | 0 | 3 | 5 | 7 | - | 4 |
B | 3 | 0 | 2 | 4 | - | 1 |
C | - | - | - | - | - | - |
D | - | - | - | - | - | - |
E | 1 | 2 | 4 | 6 | 0 | 3 |
F | 4 | 1 | 1 | 3 | 3 | 0 |
+---------------------------+
```

```
+---Forwarding Table of Router F---+
B | C=-, outport=10005, destport=10001, alive
B | C=1, outport=10005, destport=10001, alive
C | C=1, outport=10005, destport=10002, alive
C | C=3, outport=10005, destport=10002, alive
B | C=3, outport=10005, destport=10001, alive
F | C=0, outport=10005, destport=0, alive
+--------------------------------+

+------DV Table of Router F---+
    A | B | C | D | E | F |
A | - | - | - | - | - | - |
B | 3 | 0 | - | - | 2 | - |
C | - | - | 0 | 2 | - | - |
D | 7 | 4 | 2 | 0 | 6 | 3 |
E | 1 | 2 | 4 | 6 | 0 | 3 |
F | 4 | 1 | 1 | 3 | 3 | 0 |
+---------------------------+
```