CSE 489/589
Programming Assignment 3

Project Objective

- Implement a simplified version of the Distance Vector routing protocol over simulated routers.

- Basic idea:
  - When x receives new DV estimate from neighbor, it updates its own DV using Bellman-Ford equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

Programming environment

- Use 5 CSE student servers as *routers*
  - stones
  - euston
  - highgate
  - embankment
  - underground

Distance vector algorithm

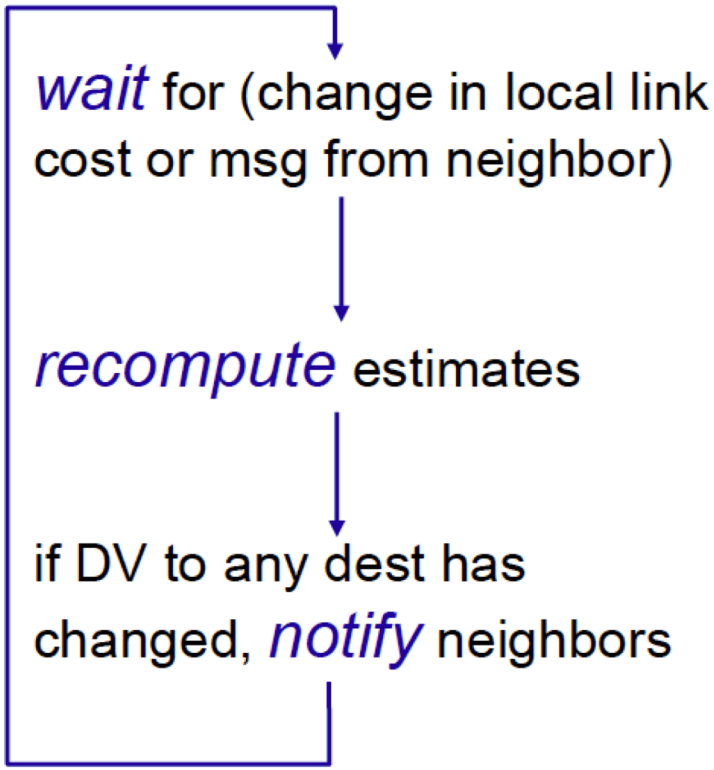
iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

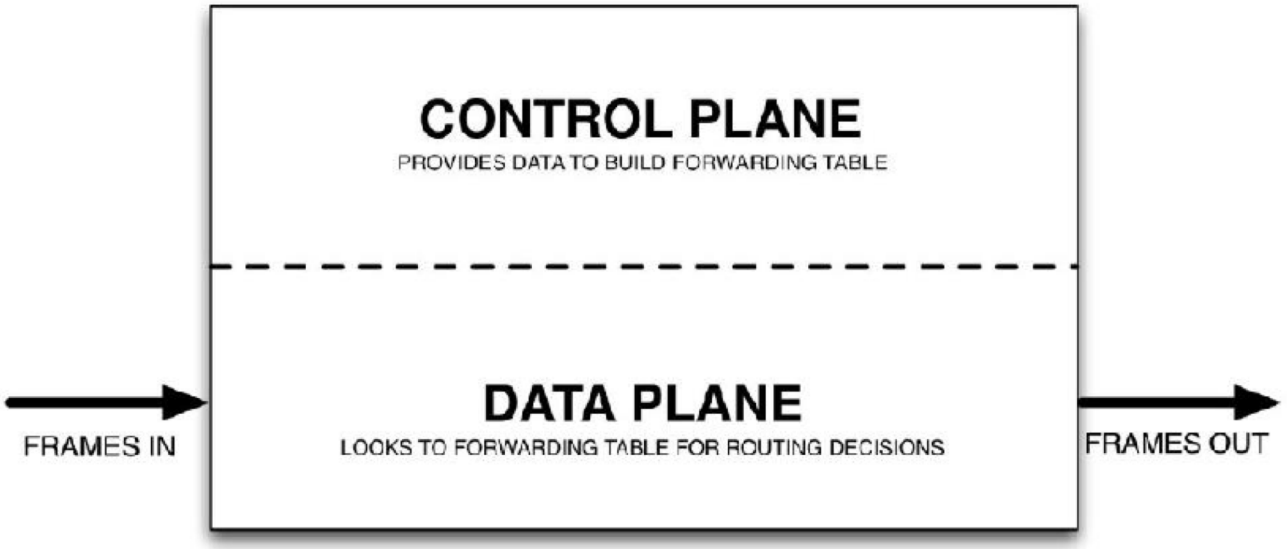
- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



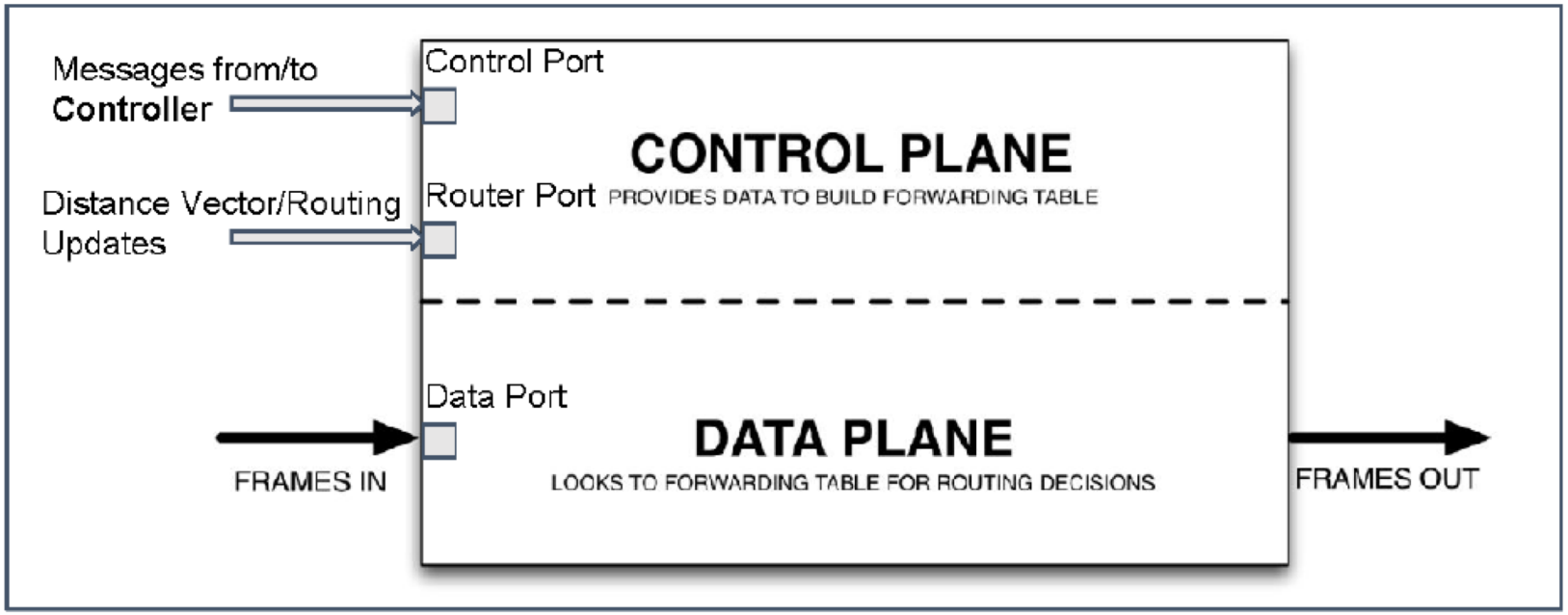
Routers

- Backbone of the internet



Routers: PA3

- Backbone of the internet



ROUTER APPLICATION

$ ./router <control port>

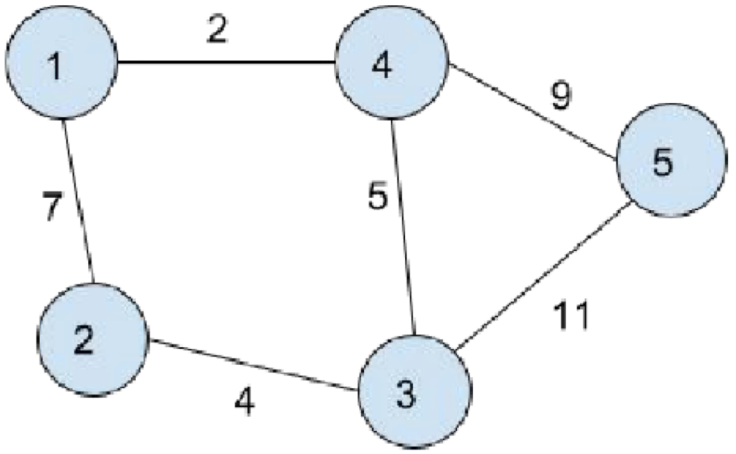
Controller

- A separate application
  - Generates *control* messages for routers
  - Expects response within a specified time limit
- Included in the template. No need to implement
- Router application needs to act and generate response packets to control messages

Topology

- Established by controller using INIT messages
- Reads a *topology file*
- Topology file
  - Snapshot of the network
  - Number of routers in the network and ID
  - IP address and all three port numbers
  - Links and their cost

Topology: Example



Line #	Line entry
1	5
2	1 128.205.36.8 4091 3452 2344
3	2 128.205.35.24 4094 4562 2345
4	3 128.205.36.24 4096 8356 1635
5	4 128.205.36.4 7091 4573 1678
6	5 128.205.36.25 7864 3456 1946
7	1 2 7
8	4 5 9
9	1 4 2
10	3 4 5
11	3 2 4
12	3 5 11

<ID> <IP address> <control port> <router port> <data port>

<router ID 1> <router ID 2> <cost>

Difference from text-book…

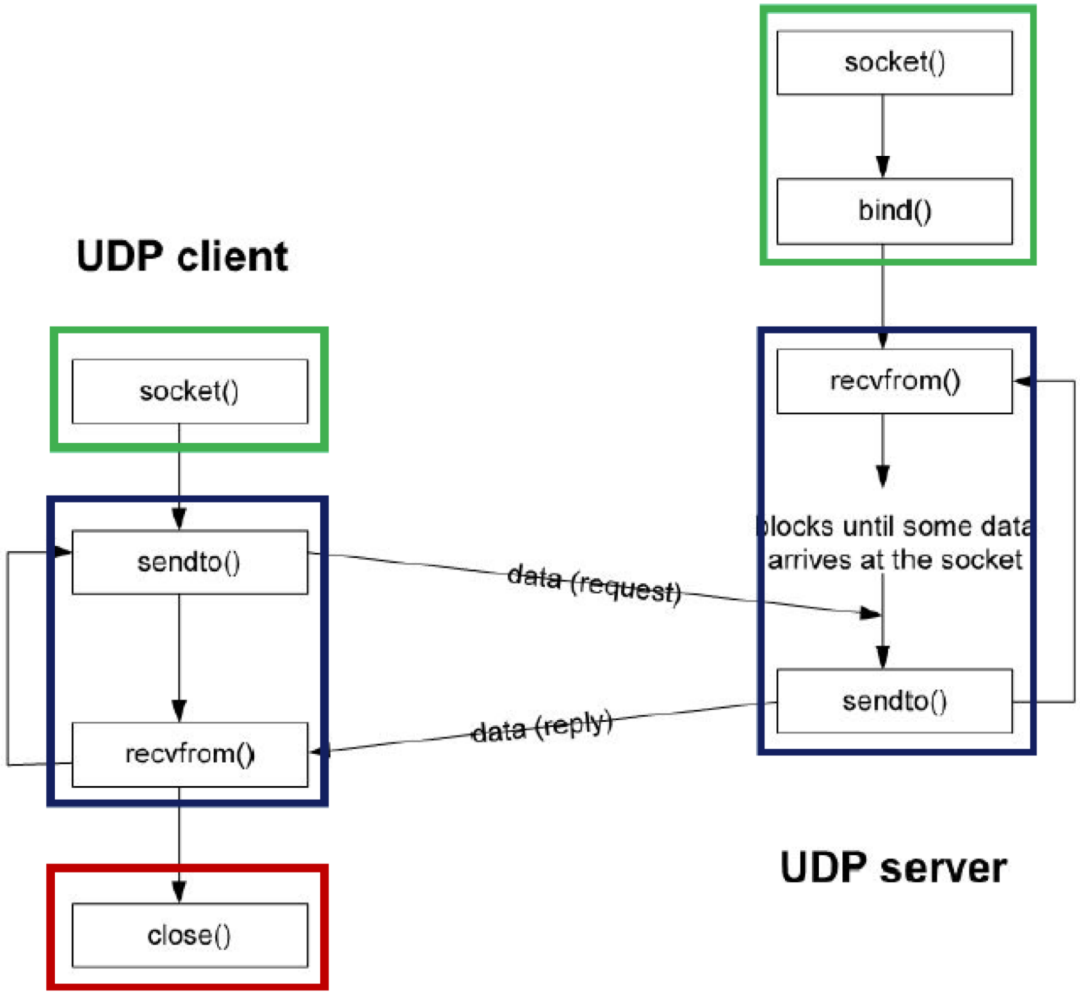
- Routing Updates: Periodic exchange Only
  - Not on DV changes

- To solve count-to-infinity?
  - No!

UDP socket overview

- Client Init.(socket)
- Server Init. (socket, bind)
  - No connection establishment
- Data transfer
  - Client sendto - Server recvfrom
  - Server sendto - Client recvfrom

UDP socket overview



UDP server socket overview

```
server_socket = socket(AF_INET, SOCK_DGRAM, 0);
if(server_socket < 0)
    return err_msg_ERR("Cannot create socket");
```

```
bzero(&server_addr, sizeof(server_addr));

server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(port);

if(bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0 )
    return err_msg_ERR("Bind failed");
```

Multiplexing

- Control Port
  - **TCP**
  - Controller connections/messages
- Router Port
  - **UDP**
  - Distance vector updates
- Data Port
  - **TCP**
  - Actual data packets

Select() again

- I/O multiplexing with select
  - Control, router and data ports
  - Data on any connections established on these ports
- Timeout
  - To implement periodic distance vector broadcast

```
int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
```

  - Multiple timers needed!
    - Each router may be on a different schedule

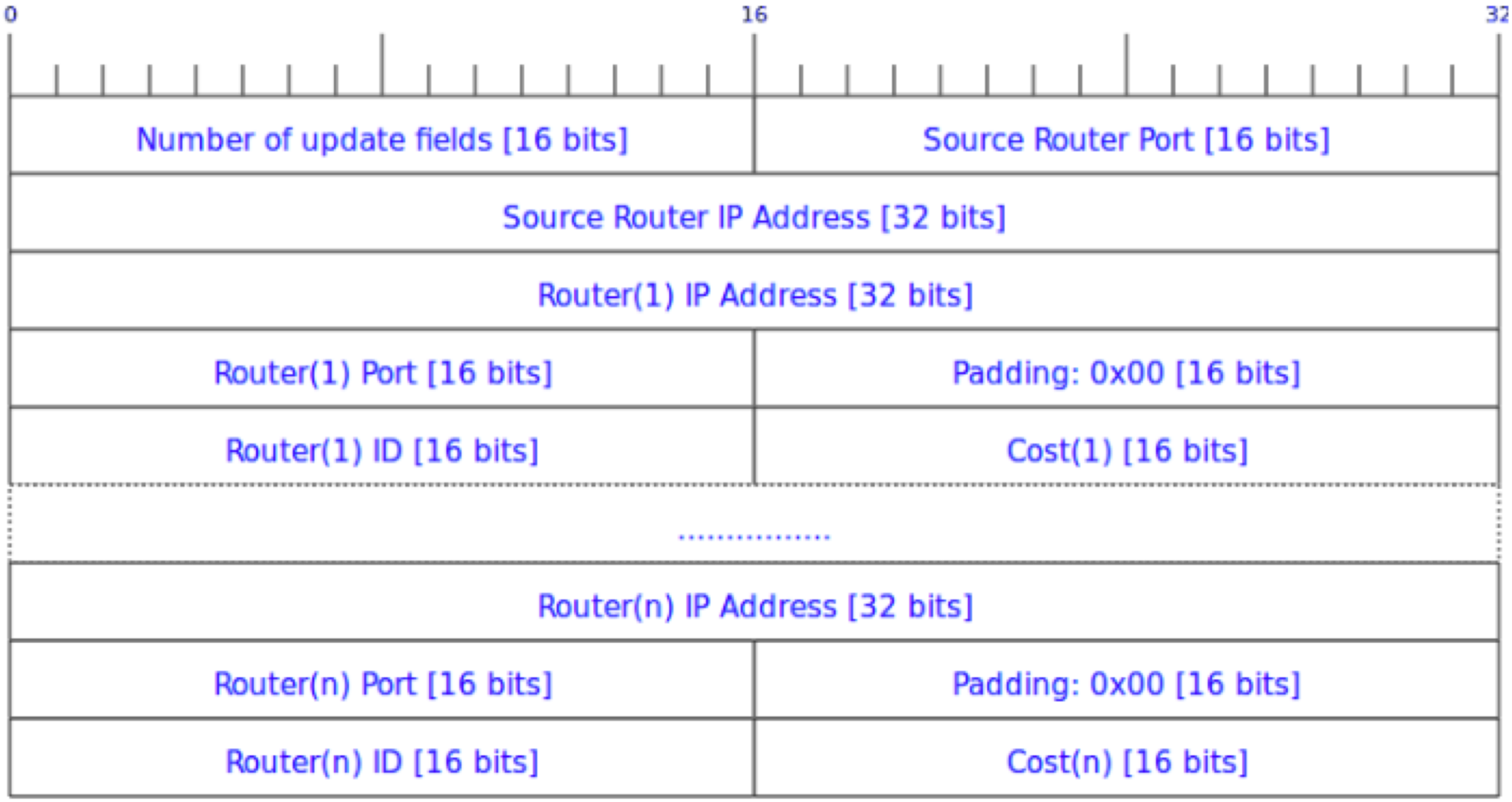
Packet formats

- Multiple packet formats
  - Routing updates
  - Control
  - Control-Response
  - Data

- Need to be followed exactly!

- Example: Routing Update Packet

Packet Formats: Example

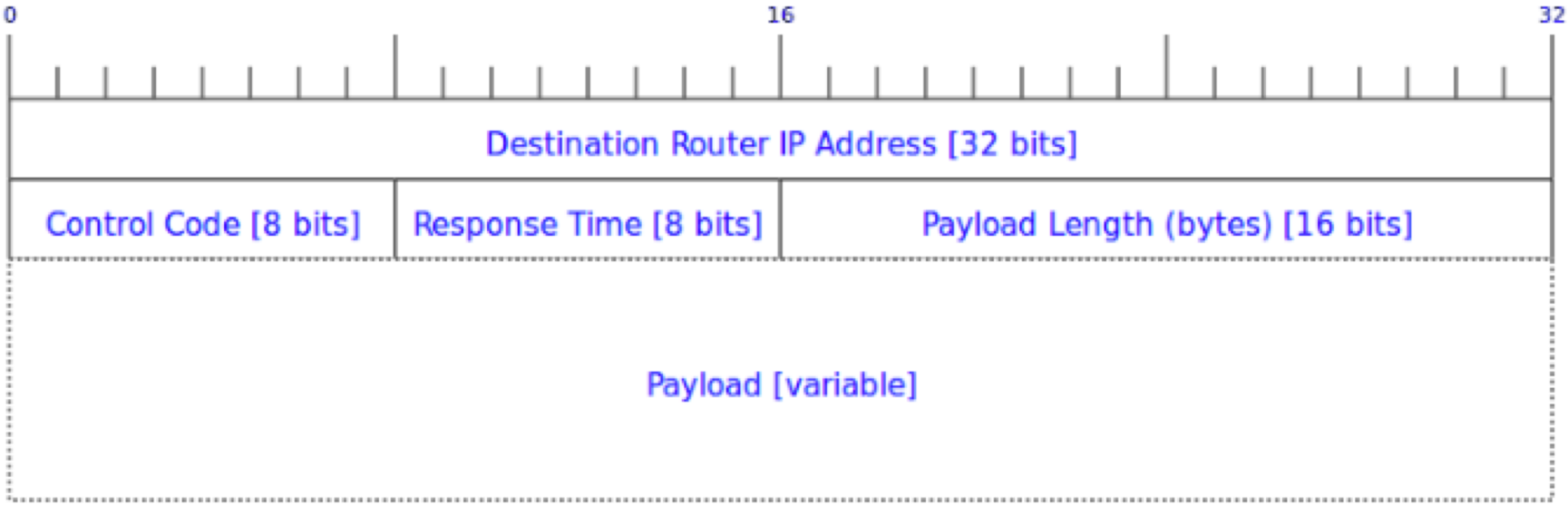


Numeric Types

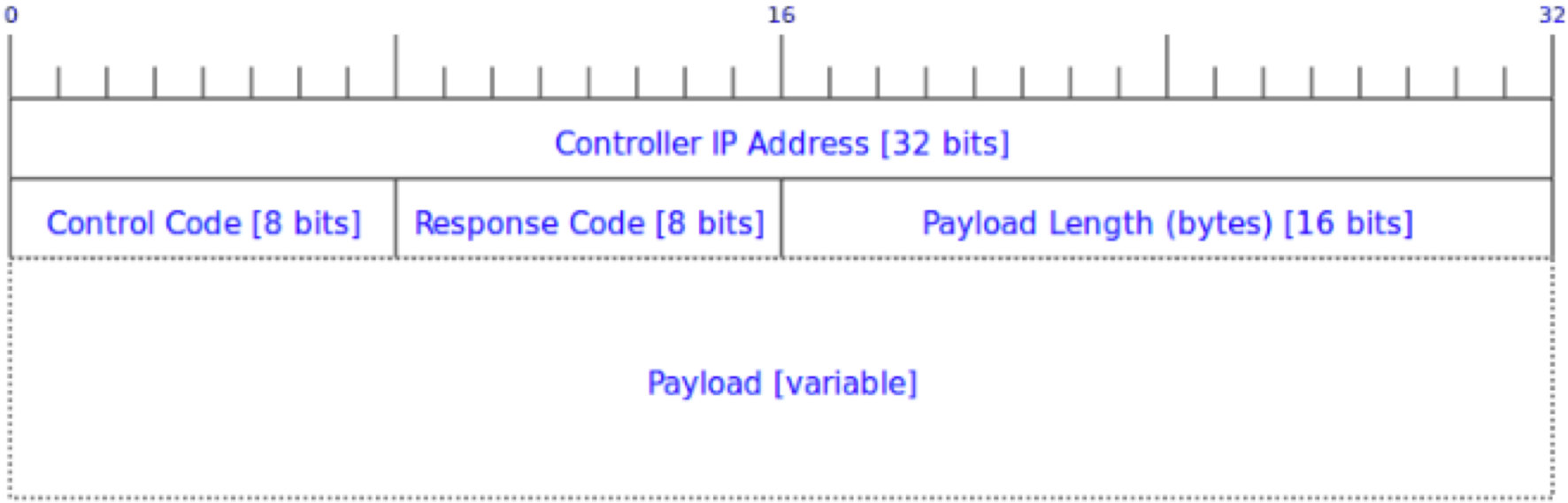
- All numeric fields, unless mentioned otherwise, are represented/encoded as their unsigned versions

- How to represent infinity (INF)?
  - Largest 2-byte unsigned integer (65535)

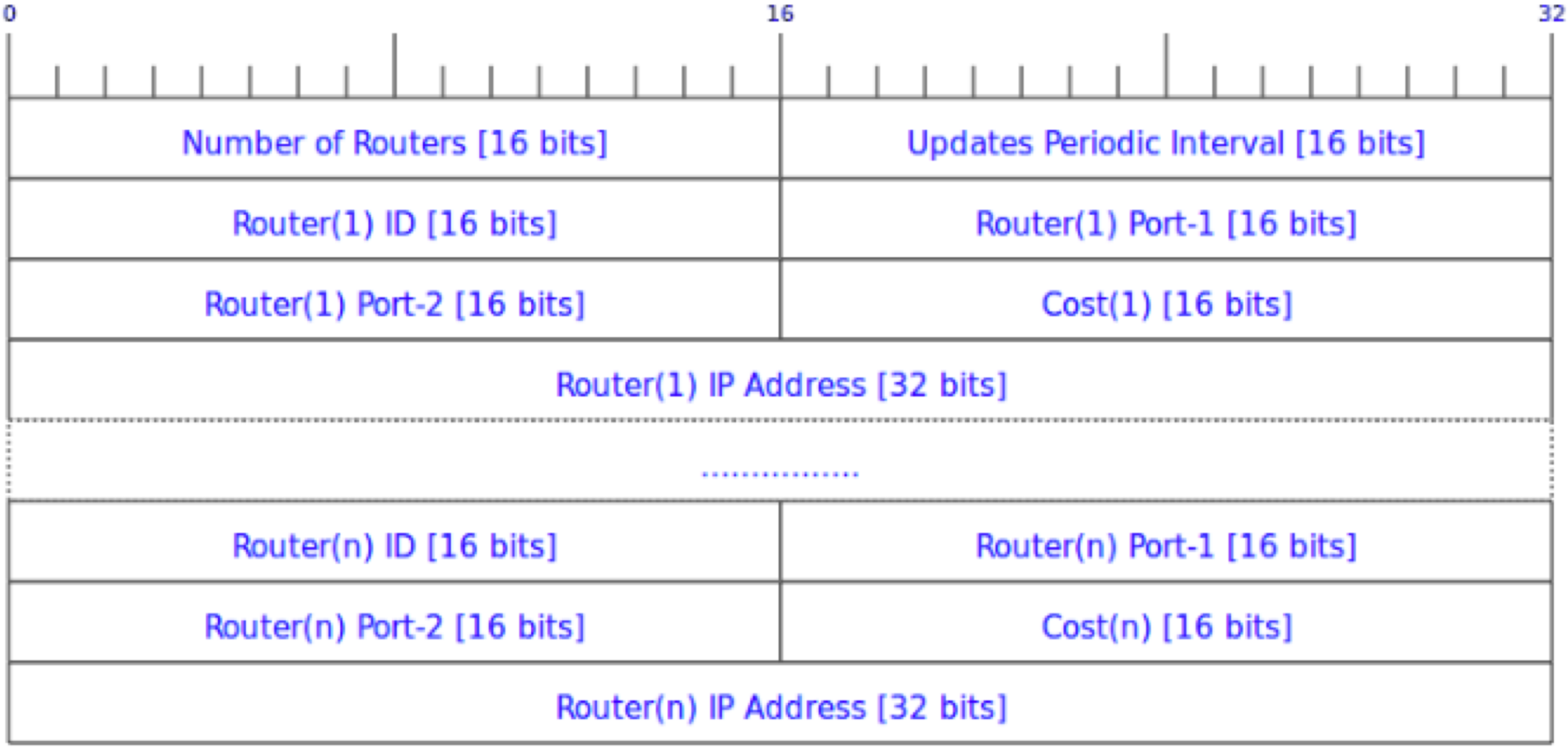
Control message header



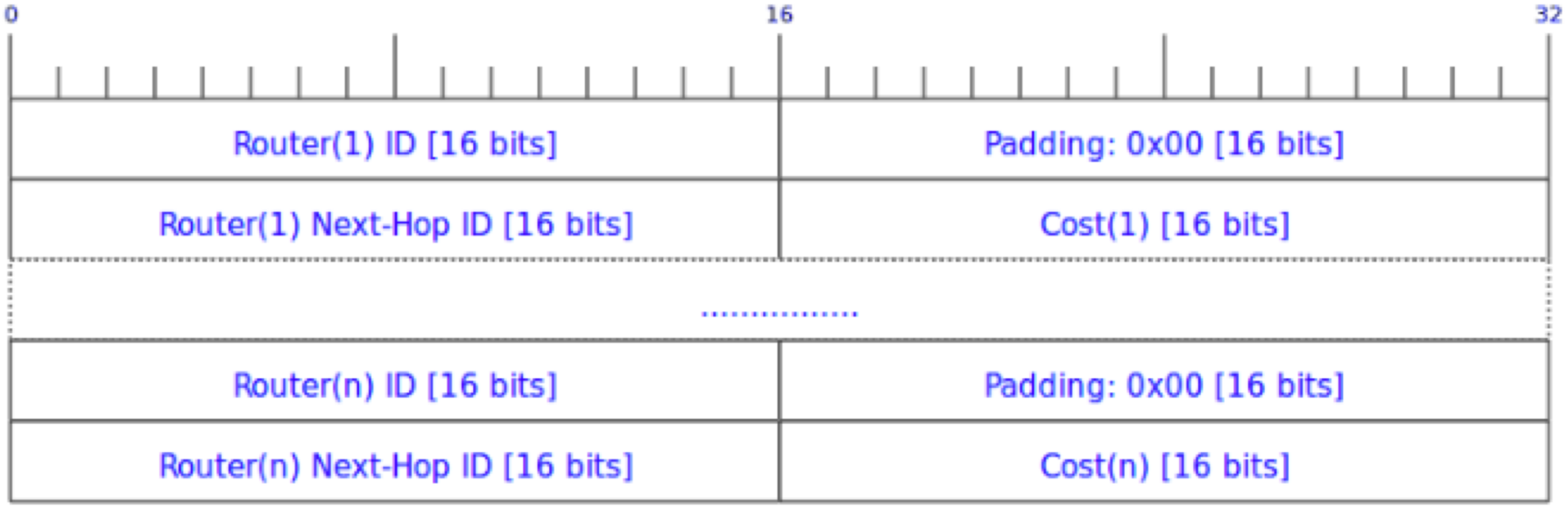
Control-Response message header



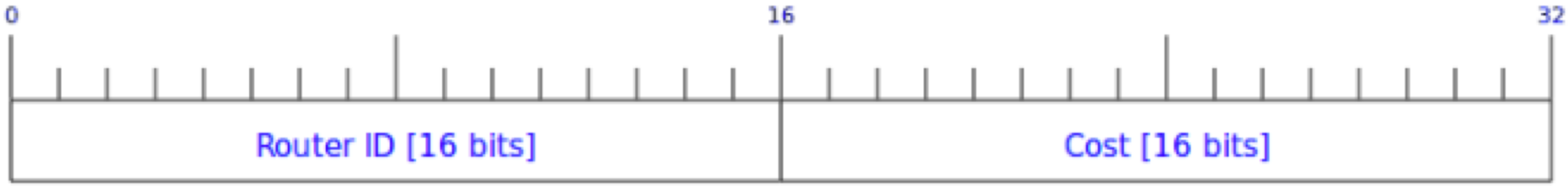
INIT (Control)



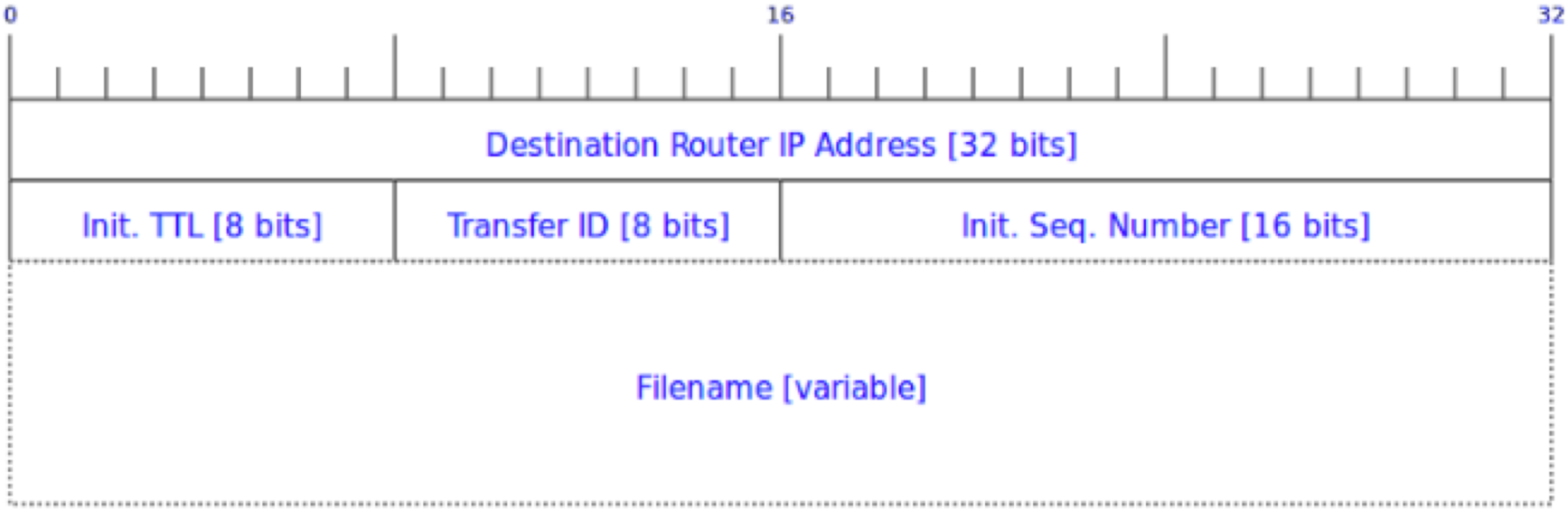
ROUTING-TABLE (Control-Response)



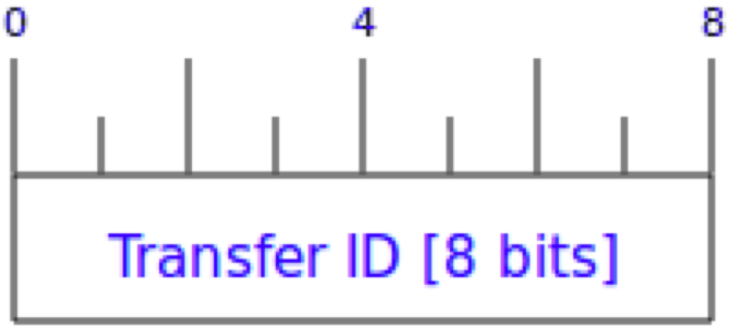
UPDATE (Control)



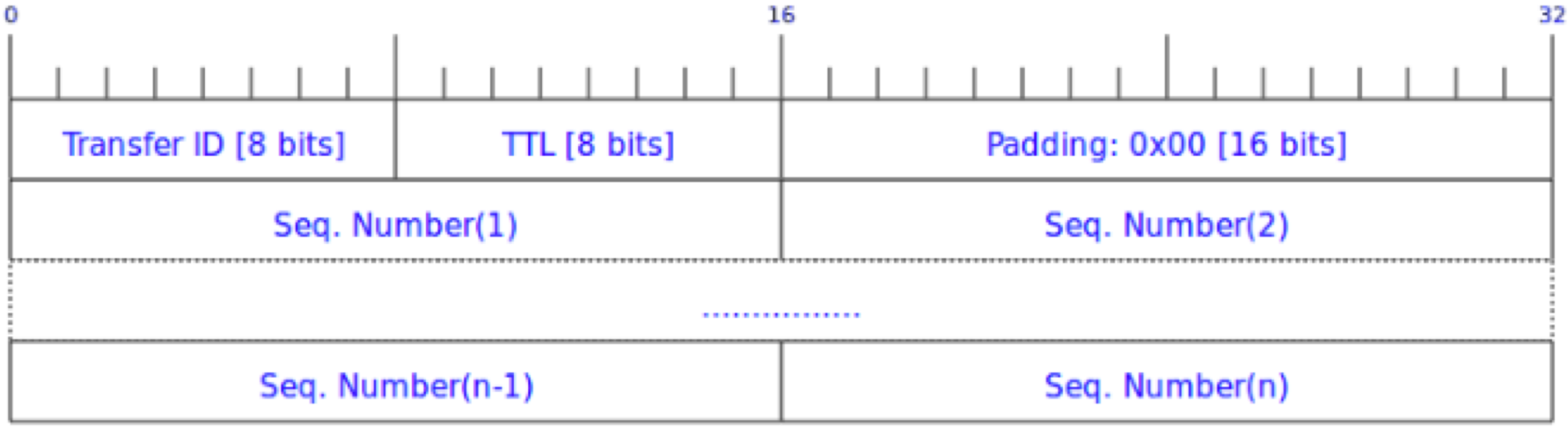
SENDFILE (Control)



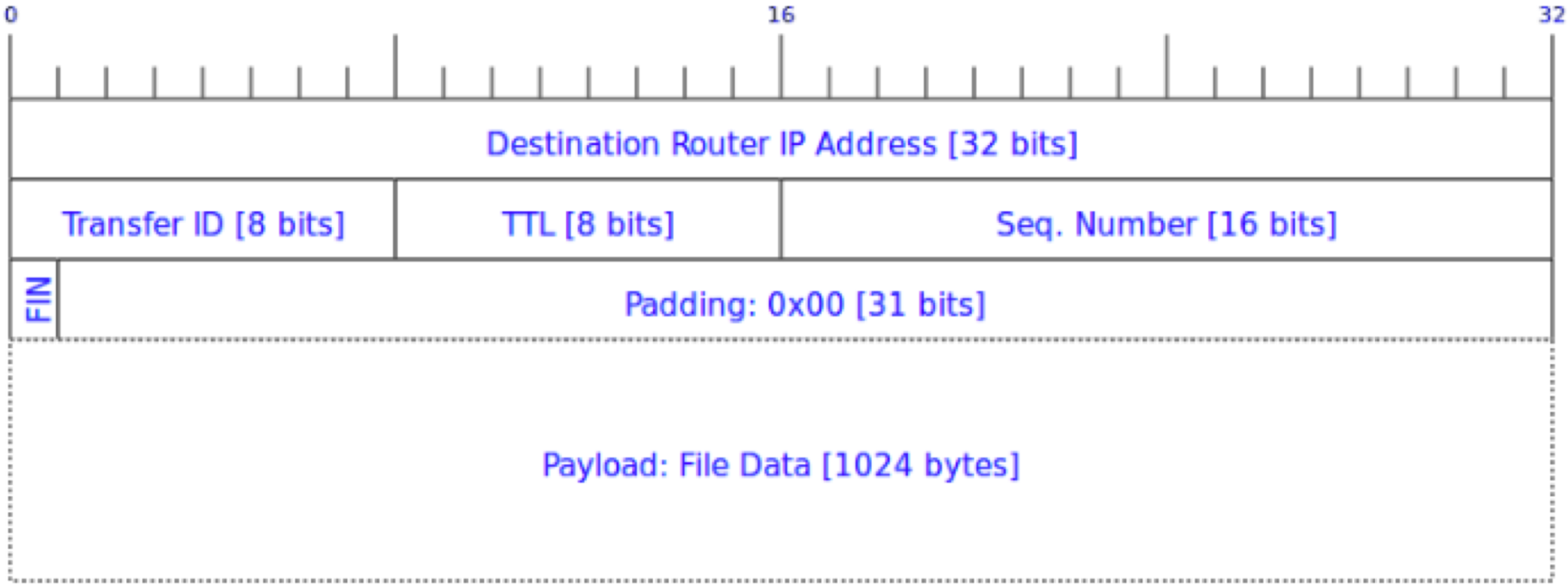
SENDFILE-STATS (Control)



SENDFILE-STATS (Control-Response)



Data Packet Format



Questions ?