RUTGERS

16:332:544: COMMUNICATION NETWORKS II

FINAL PROJECT REPORT

Design and Implementation of Content Routing Protocol

Prasad Netalkar & Mareesh Kumar Issar

Department of Electrical and Computer Engineering

Rutgers, State University of New Jersey

Piscataway, NJ-08854

1. OBJECTIVE

The goal of this project is to create a content routing scheme by routing contents using their unique content identifiers (CIDs). The design philosophy employs the use of content identifiers when requesting a piece of content to find the closest content to the user that is being requested and when the content is found at the closest host the content identifier along with host ids are deployed to fetch the contents from nearest location. The reason for using such hybrid *(HOST and CONTENT)* identifiers between the data packets and the ACK packets is to be able to employ a more standard routing scheme until the full content is delivered and acked. For routing we use reliable flooding approach where each router maintains two tables *(HOST and CONTENT)* which gets updated either through routing update process *(between routers)* or host update process *(HOST to ROUTER).* This creates a routing scheme where content identifiers or host numbers serve as the addresses.

2. DEMO

I. Network Topology

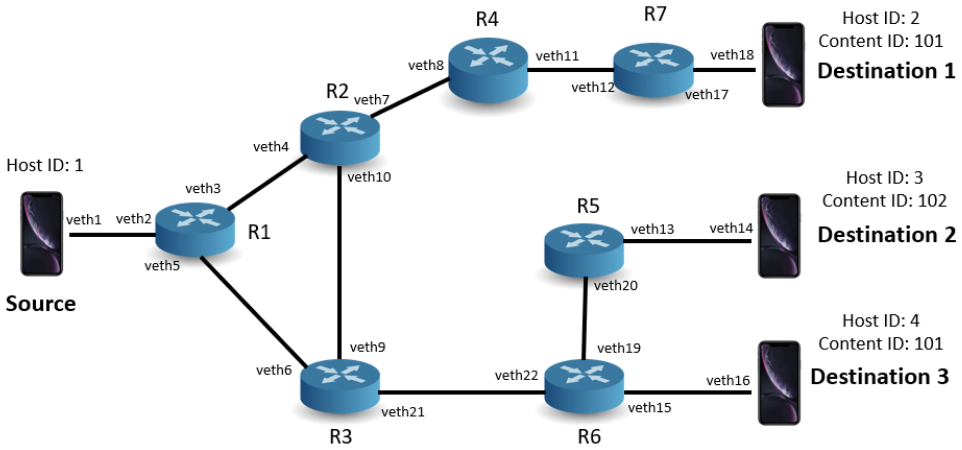


Fig 1: Sample Test Topology

II. Screenshots:

i. Content and Host Routing Table

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
-----CONTENT TABLE 501-------
CONTENT: 101 | HOP: 2 | PORT: 2
CONTENT: 102 | HOP: 3 | PORT: 2
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
-----HOST TABLE 501-------
Host: 1 | HOP: 0 | PORT: 0
Host: 3 | HOP: 3 | PORT: 2
Host: 4 | HOP: 2 | PORT: 2
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Fig 2: Sample Routing table at Router 1

ii. Packet Forwarding

```
Sending data request packet from SRC 1 for 101
Sending update packet from SRC 1 having content id
```

Fig 3: Sample Data Request Packet

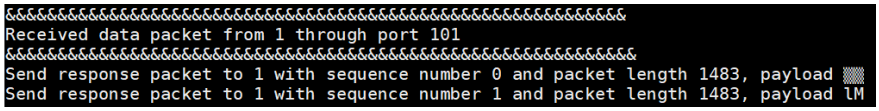


Fig 4: Sample Data Response Packet

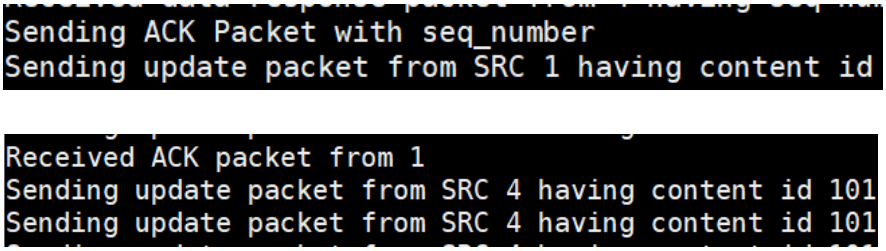


Fig 5: ACK Packets

3. PACKET STRUCTURE

I. Update Packet and Delete Packet

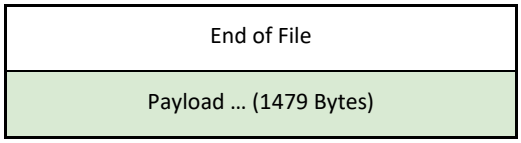
0	1	2	3	4	5	6	7
Type 0 or Type 1							
Source Address							
Length							
Content IDs...							
...							

II. Data Request Packet

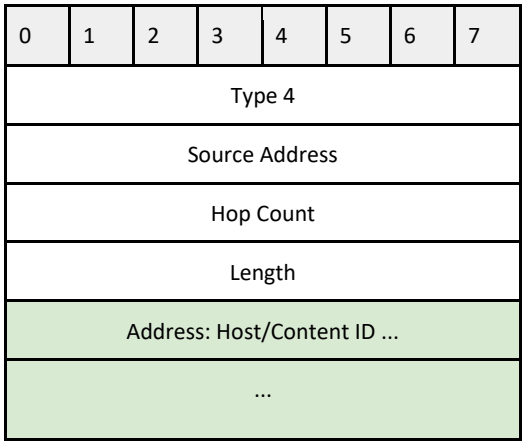
0	1	2	3	4	5	6	7
Type 2							
Source Address							
Content ID							

III. Data Response Packet

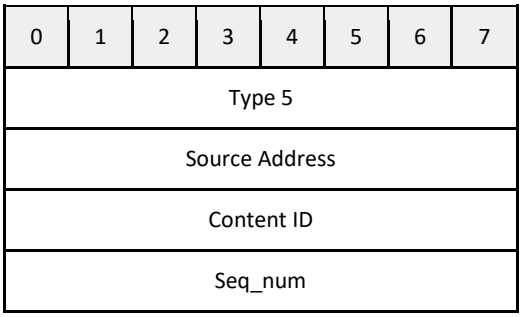
0	1	2	3	4	5	6	7
Type 3							
Source Address							
Dest Address							
Seq_num							



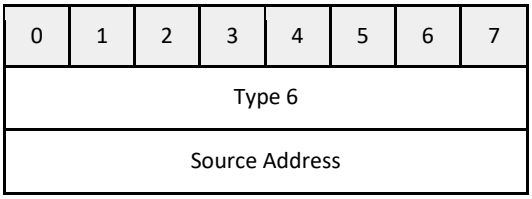
IV. Routing Update Packet



V. ACK Packet



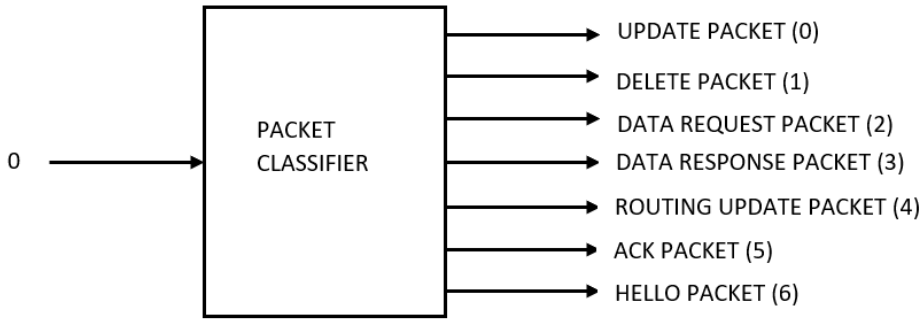
VI. Hello Packet



4. IMPLEMENTATION

I. Click Elements:

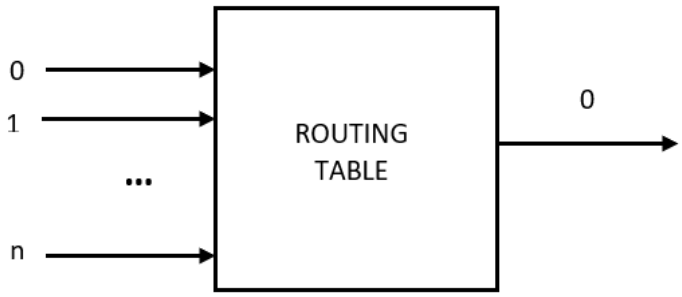
Packet Classifier:



Packet classifier element has 1 input port and 7 output ports. It receives packets from router port element and classifies it based on type field. This element is used at both host and router endpoints.

In .click file we can instantiate packet classifier using ***pc1 :: PacketClassifier().*** Based on the packet type we discard routing update packet, hello packet at host and in routers all the packets are necessary based on element configuration.

Routing Table:

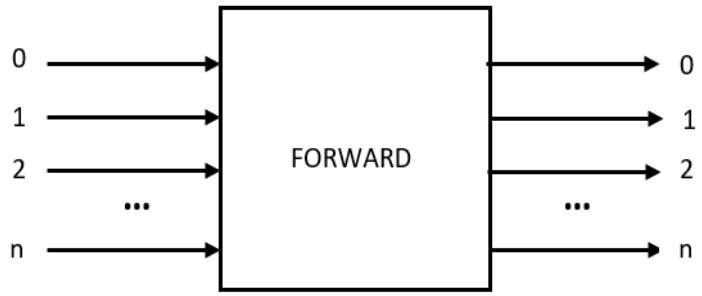


Routing table element has n input port based on topology and 1 output port. Each of the input port is connected to the respective packet classifier output ports. To begin with router port is attached to each virtual interface, from router port the packet is sent to packet classifier to classify the packets. Since routing table element builds host and content routing table only packets like update, delete, and routing update packet are necessary into this element. The update packet received by router from host is required to update zero hop neighbours (directly attached host/content). This information is propagated to the entire network using routing update packet. Also, it will receive routing update packet from neighbouring routers and the table gets updated accordingly.

In .click file the table is the routing table element is called using ***rtable :: RoutingTable (MY_ADDRESS 501).*** Here the element takes just router address as its input and its builds two hash table- HOST and CONTENT (Format: Destination, hop count, port number). The output

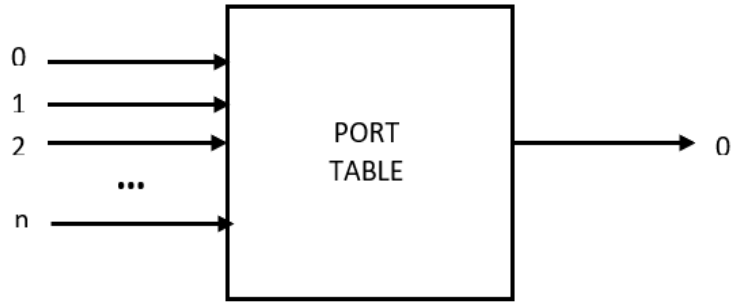
port of routing table is sent to processor element which is nothing but a broadcaster and floods routing update packet to all its interfaces.

Forward:



The forward element has n input port and n output port based on topology. It helps in forwarding of data request/response and ACK packets to the respective destination. In .click file we can initiate forward using ***forwarder :: Forward(ROUTING_ELEMENT rtable, MY_ADDRESS 501).*** This element has routing table and the router address as input. The output of packet classifier with packet type 2, 3 and 4 (data request/response and ack respectively) are send to forwarder and based on table lookup the packet is sent to the destination through respective ports. Finally, the output ports of forward element are sent to respective router ports.

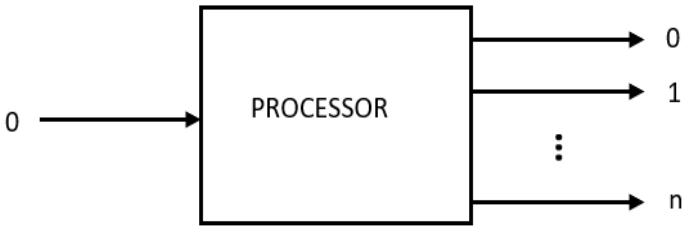
Port Table:



Port table element has n input ports and 1 output port. It is used for the routers to store information direct neighbouring routers (DESTINATION, PORT). This information is updated at each router whenever the router receives a hello packet. Also, port table element is responsible for the periodic generation of hello message.

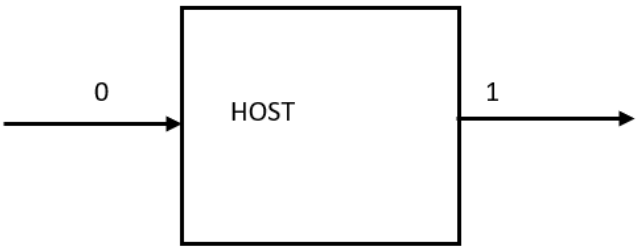
In click file, we instantiate the port table element as ***ptable :: PortTable(MY_ADDRESS 501).*** Here, 501 is the address of the router in which the element is initialized. With the help of port table generated by this element, the router becomes aware of the of its directly connected neighbours and the ports to which they are attached.

Processor:



Processor element has one input port and multiple output ports. The number of output ports depends on the interfaces attached to the host/router element. This element broadcasts the packet to all its directly attached ports. In .click files we instantiate the processor element as ***broadcast::Processor().*** It uses the function ***noutputs()*** to find the number of directly connected interfaces.

Host:



Host element has one input port and one output port. The messages that go from the host into the network pass through a router port and the messages that the host receives through the router port are passed through a packet classifier. The packet classifier filters the packets based upon its type and passes only the data request, data response and ack packet to the host.

In .click files we instantiate the host element as ***host::Host(FILENAME 101.dat, MY_ADDRESS 1,GET 101,LENGTH 1,CONTENT_ID 101).*** Here, ***MY_ADDRESS*** refers to the unique host ID, ***GET 101*** implies that the host is requesting for content 101, ***LENGTH*** signifies that number of contents that are present in that host, and ***CONTENT_ID 101*** denotes that a content with content ID 101 is present within the host with ***FILENAME*** 101.dat.

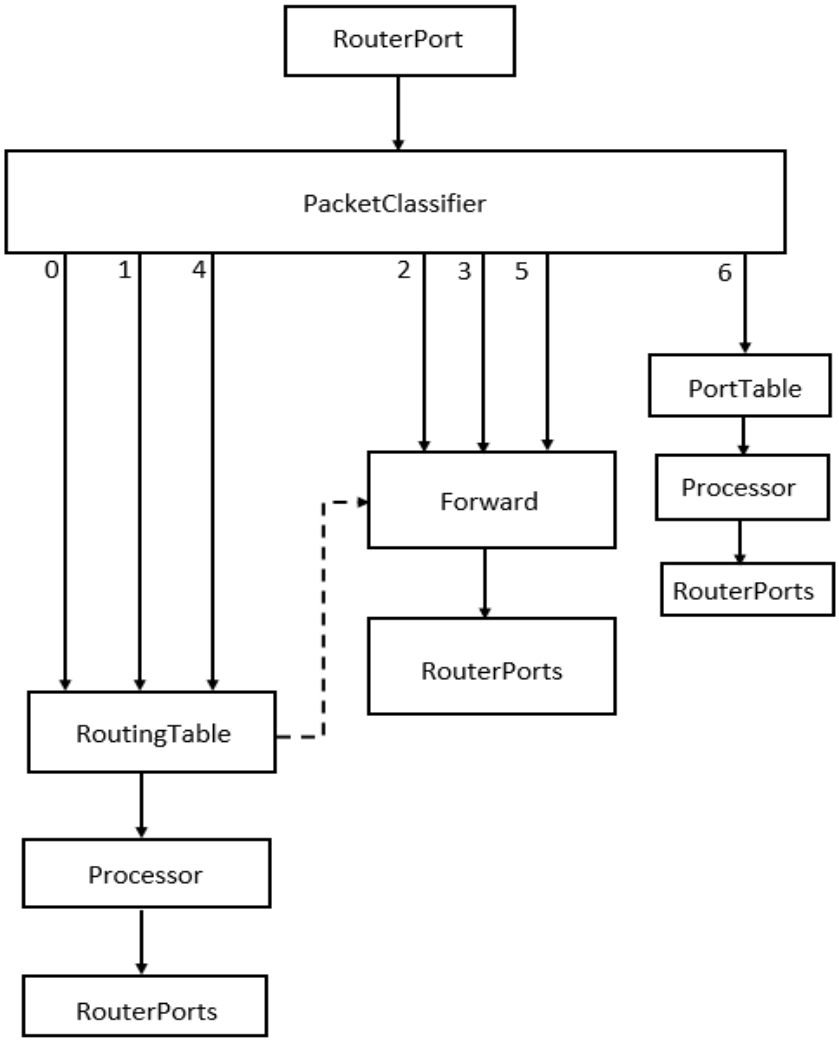
II. Click Router Flow Chart

The flowchart describes the route taken by a packet starting from when it is received at the router port. After being received at the router port, the packet is sent to packet classifier. The packet classifier then sends the update, delete, and routing update packet to the routing table element. The routing table element updates the routing table present at the router based on the information present in the packet and sends its output to Processor element. This element then broadcasts the packets on all its interfaces (by sending them to router port associated with each link).

Date request, data response, and ack packets are sent from the classifier to the Forward element (takes routing table as input), directs the packets to their destination based on the

information present in their host/content tables (towards the appropriate router port associated with the interface).

The hello packets are sent from the classifier towards the port table element. The port table at each router is updated, these packets are generated and then broadcasted to all the directly attached interfaces with the help of Processor element (which in turn sends them towards their respective router ports).



5. PERFORMANCE ANALYSIS

I. Latency

The following table gives End to End latency when host 1 requests for respective contents individually.

SRC	***DST***	***E2E Latency***
1	2	61ms
1	3	64ms
1	4	37ms

6. IMPLEMENTATION DRAWBACKS

I. No dynamic content ID update

The current implementation does not update the content id list at the host after it receives files from other hosts present in the network. This can be mitigated by updating the length and content ids field based on the event (when the last packet is received).

II. No reliability for data packet

Currently, our implementation has reliability checks implemented for the data request packet and sends a cumulative acknowledgment after the last packet has been received by the requester. But, there are no procedures implemented to deal with intermediate packet losses. This can be mitigated by implementing ack for each packet that is received at the receiver, but a major drawback of this approach is that it leads to network congestion. Instead of sending ack for each packet we could send ack for every 5 packets (like sliding window protocol).

7. CONCLUSION

The project gave us an introduction to click environment and also gave us an holistic overview of the functioning of basic network elements. Future extension of this project involves having some features of NDN network like pending interest table, content caching etc.

8. ACKNOWLEDGEMENT

We would like to thank Professor Dipankar Raychaudhuri for his valuable guidance throughout the course and his insights during the standards committee meeting. A special thanks to Sumit Maheshwari and Jiachen Chen for helping us out in implementation.