**Contents**

[1](#_30j0zll) INTRODUCTION 2

[1.1](#_1fob9te) Purpose 2

[1.2](#_3znysh7) Definitions 2

[1.2.1](#_2et92p0) Router 2

[1.2.2](#_tyjcwt) Node 2

[1.2.3](#_3dy6vkm) Content Routing 2

[1.2.4](#_3dy6vkm) Content Identifier 2

[1.2.5](#_4d34og8) Routing Table 2

[1.2.6](#_1t3h5sf) Reliability 3

[1.3](#_17dp8vu) Assumptions 3

[2](#_3rdcrjn) BOOTSTRAP AND DISCOVERY 4

[2.1](#_26in1rg) Addressing Scheme 4

[2.2](#_35nkun2) Packet Format 4

[2.3](#_lnxbz9) Procedure 5

[3](#_1ksv4uv) CONTENT ROUTING 5

[3.1](#_44sinio) Routing 6

[3.1.1](#_2jxsxqh) Routing Packet Format 6

[3.2](#_z337ya) Content Retrieval 7

[3.2.1](#_3j2qqm3) Algorithms for Content Routing 7

[3.2.2](#_1y810tw) Algorithms for Packet Forwarding 11

[3.2.3](#_4i7ojhp) Data Packet Format 11

[3.3](#_2xcytpi) State Diagram 12

[4](#_1ci93xb) RELIABILITY SCHEME [1](#_2xcytpi)4

[APPENDIX 1](#_qsh70q)5

# Introduction

## Purpose

The design goal is to create a content routing scheme by routing contents using their 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 then when the content is found at the closest host the content identifier along with content based routing scheme will be deployed to fetch contents. Afterwards, the content that is retrieved from the host it was stored and is routed back to the host that requested the content and the ACK packet is sent using the host identifiers rather than the CIDs. The reason for using the host identifiers between the data response packets and the ACK packets is to be able to employ a more standard routing scheme until the full content is delivered and acked. Routing is more like a reliable flooding approach with each routers keeping track of number of hops the content or host are located and is done using routing tables that contain CIDs, host identifiers, costs and port numbers. CID and Host information is sent to edge routers and then these routers send information to their nearest routers. This creates a routing scheme where content identifiers or host numbers serve as the addresses.

## Definitions

### Router

A network device that passes data packets between networks according to some sort of routing scheme.

### Node

A node is a point on a network to which data packets are sent or received. A node can either be a host or router.

### Content Routing

Content routing is the act of routing through a network with a predefined routing scheme that employs the use of “content identifiers” to locate and transmit content located within a network back to a user.

### Content Identifier

A content identifier also referred to as a CID is a unique marker to distinguish one piece of content from another. If two pieces of content are the same (they have the exact same data) then they will hold the same CID.

### Routing Table

A routing table is a data table stored within a router that holds information that is vital for the routing scheme to function. The exact contents of the routing table is dependant on the routing scheme employed. The routing table fills up with information from packets that are intercepted by the router containing information that it requires such as control packets.

### Reliability

A reliable network sends information across itself with successful delivery of data to a recipient. Reliability in the network can be improved by protocols (such as ARQ) or by improving hardware for less data packet drops.

## Assumptions

* Unreliable network with packet loss prob. per link of p
* Each end node is attached to only one router
* All links have same characteristics (i.e. hop cost 1 and same MTU 1500 Bytes)
* Small network having maximum number of nodes (i.e. 50)
* Packet either lost or received completely: No CRC check
* Reliable mode of transmission through ARQ
* No temporary storage at routers

**Network Topology**

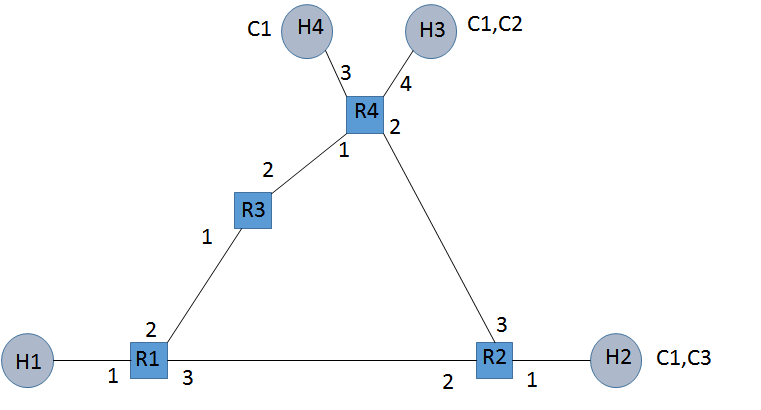


Figure:2 Network topology

## 1. 4 Packet Type/Structure

We basically have 7 different packet type structures. The syntax for these packets will be explained in the individual section below but schematics almost remains same for these packets-

**7 packet types:**

* Update/Delete content packet - control
* Routing update packet - control
* Data request packet - control
* Data response packet - data
* ACK packet – control
* Hello Packet - Control

The general semantics can be explained as under:

●**Type- 1 Byte**

* 1-Update packet
* 2-Delete packet
* 3 -Routing update packet
* 4 -Request data packet
* 5 -Response data packet
* 6 -ACK packet
* 7-Hello packet

●**SRC/DST and Router Address- 2 Byte**

* Host/Content/Router ID

●**Length-1 Byte**

* Number of addresses in payload

# BOOTSTRAP AND DISCOVERY

## Addressing Scheme

2 Byte Address Space, first 4 bits is reserved exclusively for Host or router and remaining 12 bits for Contents.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | H | H | H | C | C | C | C | C | C | C | C |

16 bits

## Packet Format

## Syntax of Bootstrap packet

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

## Semantics

**Type:** Packet type, which in this case it’s type 1 for update packet and type 2

for delete packet.

**Source Add:** Host who is generating this packet

**Length:** Number of unique contents attached to it.

**Content IDs:** The payload. Is a list of all the content ids that are present at the host generating this message.

## Procedure

During the bootstrap phase, all the hosts send the following message to their directly connected routers i.e. all the ‘edge’ routers receive the message. This message essentially consists of the content ID’s present in each host during the bootstrap phase. For the topology shown in Figure 2, the following packets will be received at the connected routers.

**At Router 1**

Messages exchanged between H1 and Router1

|  |  |  |  |
| --- | --- | --- | --- |
| Type-1 | Source H1 | Length 0 | NULL |

Since source H1 doesn’t have any content during the bootstrap phase, so its payload is empty.

**At Router 2**

Messages exchanged between H2 and Router2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type-1 | Source-H2 | Length 2 | C1 | C3 |

**At Router 3**

No packet is received at R3 during the bootstrap phase as there are no hosts/contents connected to it.

**At Router 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Type-1 | Src-H4 | Length 1 | C1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type-1 | Src-H3 | Length 2 | C1 | C2 |

Since R4 has two hosts (H3 and H4) connected to it, therefore R4 receives two packets during the bootstrap phase.

# CONTENT ROUTING

Since each content and host will have a unique id, the request for content can be based on its id and the response can be based on who is requesting the content. We are using few concepts from path vector routing where each router will generate a ‘reachability packet’ about its directly attached contents and hosts and this will be forwarded with sequence ID ‘0’ to all of its outgoing ports. We also plan to maintain two separate routing tables - host and content routing table to simplify the lookup in the table. The lookup into the table happens based on the ID structure of host as well as content. The intermediate routers having received this packet will update their routing table based on shortest hop count algorithm to both host as well as content table. Additionally, the sequence number will also be incremented (the sequence field) and then then forwarded to port other than where it was received. During this phase we also have a loop prevention mechanism which will be explained along with detailed representation of routing algorithm and content retrieval below.

## Routing

Routing table - Host and Content will have a similar structure which consist of host/content id, hop count and port number. Host and content table could have been merged (like hash table) , but implementation wise it will be much simpler to have two separate tables. A separate entry for expiration timeout is requirement to make sure we always have best anycast routes (basically takes care or deletion/update). The Host and Content tables have fixed length to simplify implementation and usage.

**Host Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Host ID | Hop Count | Port No | Timeout |
| H1 |  |  |  |
| H2 |  |  |  |
| .. |  |  |  |

**Content Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Content ID | Hop Count | Port No | Timeout |
| C1 |  |  |  |
| C2 |  |  |  |
| .. |  |  |  |

### Routing Packet Format

### Syntax of Routing Packet

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Type 3 | | | | | | | |
| Source Address | | | | | | | |
| Hops | | | | | | | |
| Length | | | | | | | |
| Payload: Host/Content ID ... | | | | | | | |
| ... | | | | | | | |

### 3.1.1.2 Semantics

**Type:** Packet type, which in this case it’s type 3

**Source Add:** Router who is generating this packet

**Hops:** Initially starts with 0, but incremented at every hop

**Length:** How many unique host or content in payload

**Host/Content:** Append all the list of unique host/content ids

## Content Retrieval

### Algorithms for Content/Host Routing

The basic algorithm used is “path vector” based forwarding where each router sends the reachability information to all the neighbouring routers. The neighbouring router will in turn check its routing table before updating the information from new packet. Here if a host/content id already exist in the table and if the hop count is lower in the table then no updating takes place, but loop prevention techniques like pop out the content/host ID from routing packet having higher hop count from packet received and then forward having sequence number incremented by one. This process continues and eventually all routers will have updated information about host or content ID location. The pseudo code is given below:

**Pseudo code:**

Router receives a type 3 packet:

num\_ids = length

increment sequence number in packet by 1

**for** (i=1 to *length*){

**if** the CID/Host ID exists in the CID/Host routing table {

**if** ((hops for CID in table) > (hops for CID in pkt)||(hops for host in table) > (hops for host in pkt)){

Replace the entry in respective table with new entry for host or content

}

**else** {

Pop out that content or host from the packet

num\_ids =num\_ids - 1 (from the packet) //loop prevention

}

}

**else** {

Add CID/Host ID with hop count along with port number to respective table

}

} //end of for loop

**if** (num\_ids == 0){ // No IDs were updated

Drop packet }//loop prevention

**else** {

Forward the updated packet to all output ports

}

### Generation of Routing Update Packet

Example initial packet generation by routers during step after bootstrap following the same topology as before-

**At Router 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type-3 | Src- R1 | Hops-0 | Length-1 | H1 |

**At Router 2:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type-3 | Src-R2 | Hops-0 | Length-3 | H2 | C1 | C3 |

**At Router 3:**

No generation

**At Router 4:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Type-3 | Src-R4 | Hops-0 | Length-4 | H3 | H4 | C1 | C2 |

The same packet is modified by intermediate routers and also new packets are generated according to some timer events.

### Content and Host Routing Table Generation

The following colour convention is used throughout the host and content routing tables:-

**STEP 1**: In the bootstrap phase, the routers receive information from their directly attached hosts.

**STEP 2:** After one time click, the routers receive packets from their one hop neighbouring routers.

**STEP 3:** After two time clicks, the routers receive packets from their two hop neighbouring routers.

**At Router 1**

**Host Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Host ID | Hop Count | Port No. | Timeout |
| H1 | 0 | 1 | 180s |
| H2 | 1 | 3 | 180s |
| H3 | 2 | 3 | 180s |
| H4 | 2 | 3 | 180s |

**Content Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Content ID | Hop Count | Port No. | Timeout |
| C1 | 1 | 3 | 180s |
| C3 | 1 | 3 | 180s |
| C2 | 2 | 3 | 180s |

In STEP 3 we assume that packet from port 3 was received at R1 before the packet from port 2 (this is done to break ties since both the packets are coming from two hops away). Also, since the router already had an entry for C1 with 1 hop, the info about C1 from the other ports will be ignored, since it already knows a closer location of C1

**At Router 2**

**Host Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Host ID | Hop Count | Port No. | Timeout |
| H2 | 0 | 1 | 180s |
| H1 | 1 | 2 | 180s |
| H3 | 1 | 3 | 180s |
| H4 | 1 | 3 | 180s |

**Content Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Content ID | Hop Count | Port No. | Timeout |
| C1 | 0 | 1 | 180s |
| C3 | 0 | 1 | 180s |
| C2 | 1 | 3 | 180s |

**At Router 3**

**Host Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Host ID | Hop Count | Port No. | Timeout |
| H1 | 1 | 1 | 180s |
| H3 | 1 | 2 | 180s |
| H4 | 1 | 2 | 180s |
| H2 | 2 | 1 | 180s |

**Content Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Content ID | Hop Count | Port No. | Timeout |
| C1 | 1 | 2 | 180s |
| C2 | 1 | 2 | 180s |
| C3 | 2 | 1 | 180s |

In STEP 3 we assume that packet from port 1 was received at R3 before the packet from port 2 (this is done to break ties since both the packets are coming from two hops away).

**At Router 4**

**Host Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Host ID | Hop Count | Port No. | Timeout |
| H3 | 0 | 4 | 180s |
| H4 | 0 | 3 | 180s |
| H2 | 1 | 2 | 180s |
| H1 | 2 | 1 | 180s |

In STEP 3 we assume that packet from port 1 was received at R4 before the packet from port 2 (this is done to break ties since both the packets are coming from two hops away).

**Content Routing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Content ID | Hop Count | Port No. | Timeout |
| C1 | 0 | 4 | 180s |
| C2 | 0 | 4 | 180s |
| C3 | 1 | 2 | 180s |

### Algorithms for Packet Forwarding

The data plane forwarding is very easy once you have routing tables built up at the router. We match the destination field and based on the address structure we do table lookups.

**Pseudo Code:**

Date request/response packet received

**if** (Type == 3 || Type == 4|| Type == 5){ //data request packet type

**if** (Dst address == Content ID){

Lookup next hop in Content Table and forward packet}

**else** {

Lookup next hop in Host Table and forward packet}

}

### Data Packet Format

Data Request:

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

Data Response:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Type 5 | | | | | | | |
| Source Address | | | | | | | |
| Dest Address | | | | | | | |
| Payload … (1495 Bytes) | | | | | | | |

Content Ack:

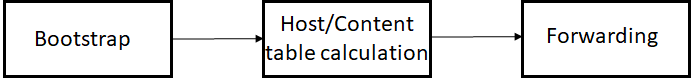
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Type 6 | | | | | | | |
| Source Address | | | | | | | |
| Content ID | | | | | | | |

### Loop Prevention Mechanism

### Any packet with src address matching to router own address - discard

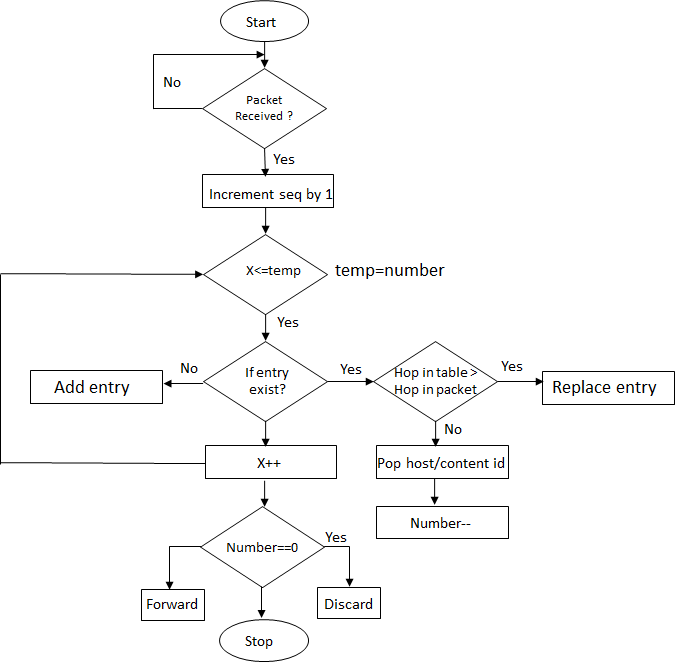
### Loop through IDs check if the hop in the packet is less than the hops stored in tables, forward ONLY the updated entries. If by the end of the loop, there was no updates, then drop the packet.

## State Diagram



# 3.3.1 Flow Chart

Routing Table Calculation:(x=0,initially)



# Keepalive / Hello

Since content delivery routing must consider the changing dynamics of the network, there must be mechanism for the routers to identify each other to build and update the routing table effectively. In the proposed scheme, we use hello packets that are frequently exchanged between routers to identify the changes in network (i.e. broken links), Each router would maintain a table of active neighbours.

PACKET FORMAT:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Type 7 | | | | | | | |
| Source Address | | | | | | | |
| Length | | | | | | | |
| “Hello” | | | | | | | |

Neighbour Router Table: The following represents a sample router table.

|  |  |
| --- | --- |
| Router ID | Port No. |
| R2 | 3 |
| R4 | 2 |

# 4. RELIABILITY SCHEME

In context of computer networking, a reliable protocol refers to a protocol which notifies the sender if the delivery of data to intended recipients was successful. It is a synonym for ‘assurance’, which is the term used by the ITU and ATM Forum. A reliable service is one that notifies the user if delivery fails, while an "unreliable" one does not notify the user if delivery fails. Reliable protocols typically incur more overhead than unreliable protocols. Thus, they function more slowly and provide less scalability. Often this isn’t an issue for unicast protocols, but they tend to become a problem for reliable multicast protocols.

We feel that our protocol design has the following merits:

1. It has some ‘load balancing’ as both ‘data request’ and ‘data response’ packet need not take the same route.
2. It is basically connectionless as there exists no end to end connectivity like that in ATM where if any router breaks down in between, paths have to be re-established.

Automatic Repeat request (ARQ), is an error-control method for data transmission which uses acknowledgements (messages sent by the receiver indicating that it has correctly received a data frame or packet) and timeouts (specified periods of time allowed to elapse before an acknowledgment is to be received) to achieve reliable data transmission over an unreliable service. If the sender does not receive an acknowledgment before the timeout, it usually re-transmits the frame/packet until the sender receives an acknowledgment or exceeds a predefined number of retransmissions. Three types of ARQ protocols are Stop-and-wait ARQ, Go Back-N ARQ and Selective Repeat ARQ.

The ARQ scheme that the protocol uses is the Stop and Wait / Go back N ARQ Scheme. There exist multiple timers in the protocol design which ensure smooth running of the protocol. The three timers used here are Update/Deletion Timer, Routing Update Timer and ARQ Timeout Timer. Each serve their own purpose as under. ‘Update/Deletion Timer’ deals with an updating required from the Host’s end to the edge router on a periodic basis. ‘Routing Update Timer’ also helps to keep track of the topological changes. In cases where a router goes down, periodical pings are sent from every router to the neighbouring routers which in turn helps to update the routing tables. ‘ARQ Timeout Timers’ are essential as all the packet transmissions, happening on an end to end basis, use the ARQ mechanism. The ARQ Timers come handy for such situations where we don’t want to wait for a response endlessly and want to take decision based on the expiration of a certain time frame.

# APPENDIX

**Timers**

Our protocol design involves three major timers, namely:

1. Update & Deletion Timer
2. Routing Update Timer
3. Route Entry Expiration Timer
4. ARQ Timeout Timer

‘Update/Deletion Timer’ deals with an updating required from the Host’s end to the edge router on a periodic basis. This is done in order to check if the host is still existing in the network. It keeps a check on the network topology and helps to update it in real time. The protocol advocates for the generation of a ‘Update/Deletion’ packet every 5 seconds.

‘Routing Update Timer’ also helps to keep track of the topological changes. In cases where a particular router goes down, periodical pings are sent from every router to the neighbouring routers which in turn helps to update the routing tables. These routing updates are also ‘event based’ depending on hello timer timeout. Also ‘Route Entry Expiration Timer’ will allow us to maintain latest route along with ‘Routing Update Timer’.

‘ARQ Timeout Timers’ are essential as all the packet transmissions, happening on an end to end basis, use the ARQ mechanism. The ARQ Timers come handy for such situations where we don’t want to wait for a response endlessly and want to take decision based on the expiration of a certain time frame. The timeout of these timers essentially depends on the nature of the network i.e. the host density, the sparseness of the network etc.

**Summary**

The proposed protocol model uses ‘Host/Content ID’ lookup approach. BGP style path routing has been used where every router notifies and exchanges information with immediate neighbours. Essentially, pairwise reachability exchange takes place. Eventually, these reachability exchanges help in defining the existing network topology at a given instant of time. At the bootstrap phase, the host/content attached to each router will update its directly connected neighbours.

Five types of packet structures are used in our protocol, namely: Update/Delete content packet, Routing update packet, Data request packet, Data response packet, Ack packet. Among these packets, all deal with the control plane except the ‘Data Response’ packet.

Each router maintains a set of two tables namely, ‘Host Routing Table’ and ‘Content Routing Table’. Some of the rows can go unused and now the information about host/content id, hop count and port number is all stored within the table. We build the host and content id table simultaneously. A plausible alternative to the two-table concept is the ‘Hash table’ implementation which has been ignored as the lookup is very expensive.

The algorithm used is the ‘Path Vector’ based forwarding where each router exchanges the reachability information with all the immediate neighbouring routers. Each router generates a reachability packet piggybacking the host and content id along with sequence number which starts from 0.

There exists a loop prevention scheme which operates as under:

1. Discard any packet with source address matching to the router’s own address
2. Discard any packet when the host or content is already in routing table with lower hop count. This is repeated this till the number ‘0’ is obtained.

The ARQ scheme that the protocol advocates are the Stop and Wait / Go back N ARQ Scheme. It has been used for reliable transmission. There exist multiple timers in the protocol design which ensure smooth running of the protocol. The three timers used here are Update/Deletion Timer, Routing Update Timer and ARQ Timeout Timer.