# INTRODUCTION

Multicast is the delivery of a message to a group of destination computers simultaneously in a single transmission from the source. The routers, if required by the topology create duplicates of the message automatically and forward multicast packets.

With Unicast traffic one copy of a packet goes to each client so if there is a large group of clients a large number of the same packets take up bandwidth on the same network, particularly if the traffic is multimedia. The multicast is appropriate in such a situation. It is basically ‘Replicated Unicast’. It is mostly implemented in IP multicast, which is a technique for [one-to-many](http://en.wikipedia.org/wiki/Point-to-multipoint_communication) communication over an IP infrastructure in a network.

Multicast allows each system to choose to participate in a given multicast thereby reducing the resource, as each computer must process each datagram.

## Purpose

The purpose of this project is to design and implement a multicast system. A message is to be sent to a maximum of three recipients and each packet will have 1, 2 or 3 destination addresses.

By implementing such a system, requirement of high bandwidth-intensive applications with its inherent ability to replicate single stream when and where necessary are fulfilled. It helps in reducing traffic and server loads.

## Definitions

### Router

It is a device used to forward data packets between computer networks. It is connected to two or more data lines and when packets arrive to these lines, the router reads the address and forwards the packet accordingly to the respective destination.

### Node

A node is a communication endpoint (terminal), which is an active electronic device capable of sending, receiving or forwarding information over communication channels.

### Multi-casting

Multicasting refers to the transmission of messages to a selected group of hosts present in the network. It uses network infrastructure efficiently by requiring the source to send a packet only once, even if it needs to be delivered to a large number of receivers. The nodes in the network take care of replicating the packet to reach multiple receivers only when necessary.

### Reliability

It is defined as the efficient delivery of all the data to all the group members. A multicast transmission is delivered to destination group members with the same “best-effort” reliability as a standard unicast IP transmission.

### Routing Table

Routers contain Routing Tables to store data information regarding the list of routes and hops along with their cost to particular network destinations. With the help of routing table, routers are able to know information stored in it to forward the packet to the desired destinations.

## Assumptions

### Reliability

The underlying network is and transmission is not guaranteed to reach all members of the group, or arrive in the same order relative to the transmission of other packets. The packet loss probability on each link is approximately p.

### Routers

Each end node or terminal is attached to only one router, which makes the transmission of message and implementation of algorithm simpler and easy.

### Characteristics

All links have the characteristics of hop cost = 1 and the MTU (Maximum Transmission Unit) of 1500 bytes.

### Nodes

The entire network consists of a maximum of 50 nodes.

# BOOTSTRAP AND DISCOVERY

## Addressing Scheme

In this project, flat addresses are used. Addresses are assigned manually, at the startup of each node.

A multicast address is defined as a sequence of unicast destination addresses.

## Packet Format

### Hello Packet

The Hello Protocol is responsible for establishing and maintaining neighbor relationships. Hello packets are sent periodically every 30 seconds out to all node interfaces until acknowledgements are received. Figure 1 below shows the packet format for Hello packet and the relative ACK packet.

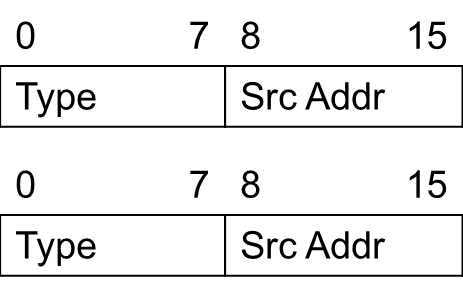


Figure 1: Hello packet and Hello ACK packet format

They consist of two fields. Type represents the type of packet and has value 0x00 for the Hello packet and 0x01 for the ACK. The Src Addr represents the address of the sending router for the Hello packet while is replaced by the replying Router with its own address.

## Procedure

Initially, each router broadcasts hello packets to its neighboring routers until it receives an ACK in return. Each host broadcasts hello packets to their neighboring router until it receives an ACK. After the connections are established, routers start periodically sending link-state update packets, and calculating their shortest paths using the Dijkstra algorithm.

# ROUTING AND MULTICASTING

## Routing

Routing protocol is link-state based. Each node constructs the map of the entire network. Upon constructing the map, the Dijkstra algorithm is performed in order to calculate shortest paths to all the other nodes in the network.

### Routing Packet Format

Routing information is exchanged with the neighboring routers in a way similar to the OSPF protocol, with some simplifications.

The link-state packet format is shown on Figure 2.

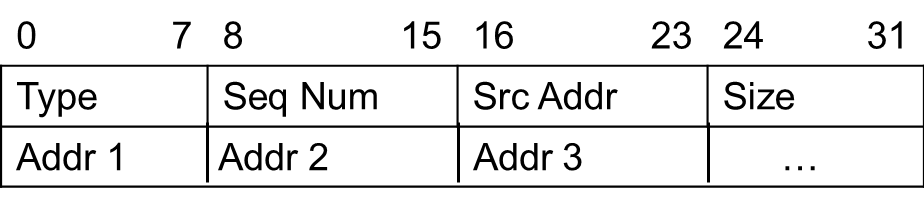


Figure 2: Link-state packet format

The Type field represents the type of the packet, in this case it is the link-state information packet that corresponds to 0x03. Seq Num is the sequence number, used as described earlier. The first packet sent has sequence number 0x00 and increases by 1 each new packet sent. Src Addr contains the address of the router that has sent the link-state packet. The last byte in the packet header contains the number of the link-state addresses contained in the message. Padding is added to align the packet to 32 bits.

Link-state messages are sent at start up, periodically every 2 minutes and in case of a topology change. Every time a new packet is sent the 2 minutes timeout is restarted. Link-state information is distributed throughout the network using reliable flooding. Each router knows which nodes it is directly connected to and through which ports. It then sends link-state messages to all of the neighboring routers, containing the list of its neighboring routers. Upon receiving a link-state message from its neighbor, each router sends it to all of the other neighbors excluding the one that it received the message from. In case the message that a router receives has already been received previously, the router discards the message. This procedure is performed throughout the network until all routers receive link-state messages from all other routers in the network. The routers then construct the network map and perform the Dijkstra algorithm for themselves and their neighbors to all the possible destinations in order to build their own and their neighbours’ forwarding tables.

In order to determine if a link-state message has already been received, a sequence number is added to each link-state message. It increments each time a new link-state message is generated.

### Forwarding Table

As it will be described later, each router needs to maintain a variable number of forwarding tables; in particular it needs to maintain a forwarding table containing the set of <*destination, cost, next hop*> triples for both itself and all its neighbors. Table 1 shows an example of a forwarding table.

No single forwarding table is stored in the system as next hops are calculated on the fly from the information obtained from the routing tables.

Table 1: Forwarding table

|  |  |  |
| --- | --- | --- |
| **Destination** | **Cost** | **<Port #, Next Hop>** |
| D1 | 5 | <1003, R3> |
| D2 | 4 | <1005, R5> |
| D3 | 5 | <1007, R7> |

## Multicast

The multicast used in this case is packet-oriented. There will be no multicast groups. Each packet can have arbitrary destinations (up to 3 of them). The multicast routing is based on unicast information available from the forwarding tables, as explained in the Routing section. Upon receiving a data packet, a router checks the packet’s time to live. If it’s zero the router discards the packet, otherwise decrements it and forwards it in the following manner:

* if the number of destinations is 1, the next hop address from its own forwarding table is chosen and the packet is forwarded to that next hop. If the destination is not present in the routing table, the packet is forwarded to a default router;
* if the number of destinations is 2 or 3, the multicast algorithm to determine the next hop(s) is performed. The algorithm will be described later in this part.

### Data Packet Format

The format of the data packet is given on Figure 3.

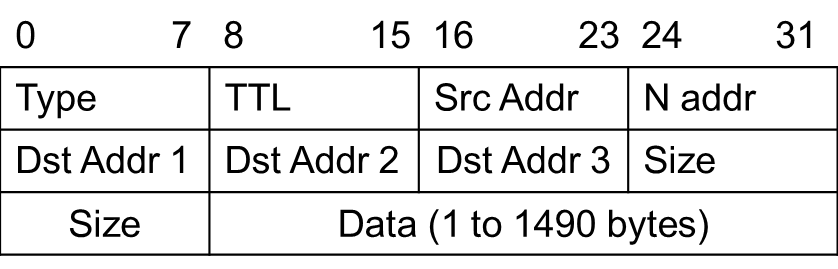


Figure 3: Data packet format

The maximum size of the whole packet is 1500 bytes and it is the same throughout the network. No fragmentation or reassembly is needed. Explanation of each field in the packet format:

* Type field contains the type of the packet, in this case it’s the data packet;
* TTL is the packet’s time to live and is decremented by each router before forwarding a packet;
* Src Addr represents the address of the original source of the packet, and it is not changed;
* N addr is the number of the following addresses, it can be 1, 2 or 3;
* List of N destination addresses follows;
* Size contains the number of bytes of the payload (data part of the packet);
* The data of is in the end of the packet.

### Algorithm for Multicast Forwarding

Upon receiving a packet with multiple destinations, the router starts considering its own and its neighbors forwarding tables to find the best combination in terms of number of packets generated and distance to the destinations.

First of all, it starts excluding all those combinations for which the cost of reaching the destination from the next hop is greater than the one from the considered router. From the remaining combinations the following formula is calculated:

*Cost = CA + CB,*

where CA is the number of links used in next hop and CB is the average cost to reach destination using unicast from next hop.

The following example is given to provide additional clarification. Consider the case shown in the following example in which router R1 has 3 neighbors. Forwarding tables that R1 contains are shown in Tables 2, 3, 4 and 5.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 2: Forwarding table for R1   |  |  |  | | --- | --- | --- | | **Destination** | **Cost** | **<Port #, Next Hop>** | | D1 | 3 | <1004, R4> | | D2 | 4 | <1002, R2> | | D3 | 3 | <1003, R3> | | Table 3: Forwarding table for R2   |  |  |  | | --- | --- | --- | | **Destination** | **Cost** | **<Port #, Next Hop>** | | D1 | 4 | <2003, R3> | | D2 | 3 | <2006, R6> | | D3 | 4 | <2001, R1> | |
| Table 4: Forwarding table for R3   |  |  |  | | --- | --- | --- | | **Destination** | **Cost** | **<Port #, Next Hop>** | | D1 | 3 | <3004, R4> | | D2 | 3 | <3006, R6> | | D3 | 2 | <3007, R7> | | Table 5: Forwarding table for R4   |  |  |  | | --- | --- | --- | | **Destination** | **Cost** | **<Port #, Next Hop>** | | D1 | 2 | <4008, R8> | | D2 | 5 | <4001, R1> | | D3 | 3 | <4003, R3> | |

### After excluding the combinations with greater distances from the next-hop from the 33 = 27 possibilities, R1 remains with the following possible combinations:

* D1 – through R3 or R4;
* D2 – through R2 or R3;
* D3 – through R3 or R4.

This gives a total of 8 combinations. The Cost is calculated for all of those combinations according to the formula given earlier:

*Cost = CA + CB.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D1 | D2 | D3 | CA | CB | Total Cost |
| R3 | R2 | R3 | 2 | (3+3+2)/3=8/3 | 4.67 |
| R3 | R2 | R4 | 3 | (3+3+3)/3=3 | 6 |
| R3 | R3 | R3 | 1 | (3+3+2)/3=8/3 | 3.67 |
| R3 | R3 | R4 | 3 | (3+3+3)/3=3 | 6 |
| R4 | R3 | R3 | 2 | (2+3+2)/3=7/3 | 4.33 |
| R4 | R3 | R4 | 2 | (2+3+3)/3=8/3 | 4.67 |
| R4 | R2 | R3 | 3 | (2+3+2)/3=7/3 | 5.33 |
| R4 | R2 | R4 | 2 | (2+3+3)/3=8/3 | 4.33 |

In this case, the combination of the lowest cost is if R3 is chosen for all 3 destinations. The result we get in this case is

*CA + CB* = 1 + 8/3.

Just for clarification, if R4 was chosen as the next hop for D1 instead of R3, *CB* would have a lower value of 7/3, but *CA* with its value of 2 (due to the packet being duplicated) would increase the total cost.

## State Diagram

The state diagram of an end node is given on Figure 4.

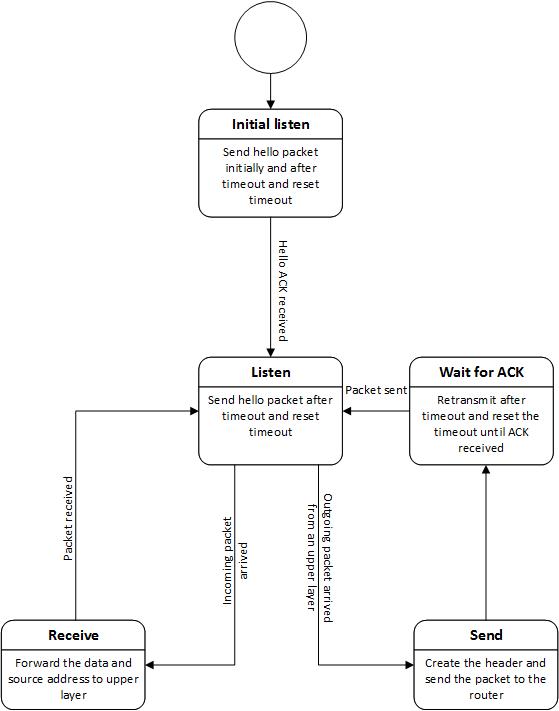


Figure 4: End node state diagram

When an end node is started, it sends hello packets until it gets an answer from a neighboring router. Hello packets are also sent after hello timeout occurs. In the listen state, it is listening to the incoming port for incoming packets and to the upper layer for outgoing packets. When an outgoing packet arrives, it creates the packet header and sends it via outgoing port. It then waits for an acknowledgement, and retransmits after a timeout until the ACK arrives.

Router, as seen on Figure 5, enters the listen state immediately. It sends its link-state update packets after every 2 minutes and hello messages every 30 seconds. If it receives a data packet, it forwards it, waits for an acknowledgement and retransmits after a timeout if necessary, until an ACK arrives.

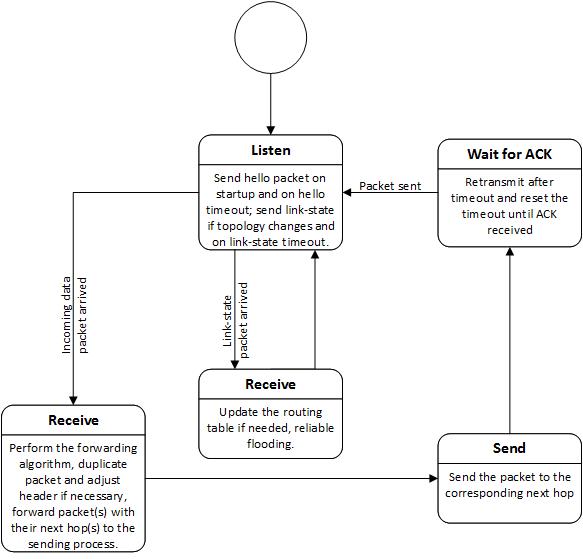


Figure 5: Router state diagram

# RELIABILITY SCHEME

Hop-by-hop transport principle is used throughout the network. It is accompanied by stop & wait ARQ scheme.

## Link Layer Packet Format

To implement the hop by hop reliability the stop & wait protocol is applied at layer 2. Each data packet received from the upper layer is encapsulated as shown in Table 6.

Table 6: Link layer packet

|  |  |
| --- | --- |
| 0 | 7 |
| Sequence Number | |
| Payload (2 to 1499 bytes) | |

Each packet has a sequence number with the value of either 0x00 or 0x01, and it switches with each outgoing packet, no matter the type of the packet. The first sent packet always has sequence number 0x00. ACK packet includes the sequence number of the acknowledged packet. An outgoing packet goes to an outgoing buffer when it’s being sent. If ACK is not received within a given time, the packet is retransmitted. When the ACK is received, the packet is removed from the outgoing buffer. The ACK packet is shown in Table 7. It is the only packet that can be of the size of 1 byte at the link layer. It contains the sequence number of the packet it acknowledges. The type of the packet that is being acknowledged does not affect the ACK.

Table 7: ACK packet format

|  |  |
| --- | --- |
| 0 | 7 |
| Sequence Number | |

The timeout for retransmission is 0.3 seconds.

# IMPLEMENTATION GUIDELINES

This section gives some guidelines on how to implement the needed software components of the system. There are three major objects that will need to be implemented: Routers, Senders and Receivers.

To complete the task it is required to have a good knowledge of the following programming techniques:

* multi threaded programming;
* semaphores management;
* network programming.

If possible, reuse the [library](http://www.winlab.rutgers.edu/comnet2/Projects/example3.tar) provided for previous projects.

## Routers

Routers are going to be the most complex elements of the system. Figure 6 shows a general description of the system.

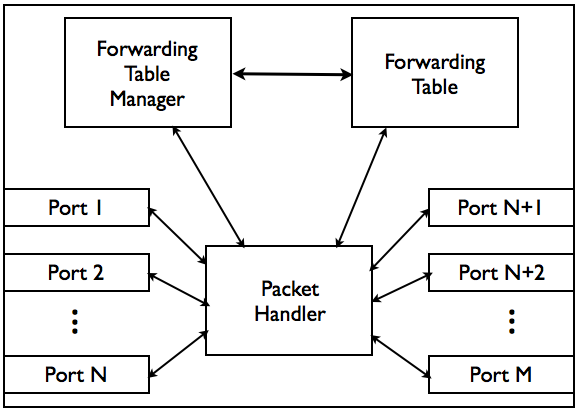


Figure 6: Router objects composition

### Ports

Ports are the objects that are in charge of exchanging packets with other nodes. They implement an outgoing queue that caches the packets that the Packet Handler has assigned to them. Moreover they listen for incoming packets and once they are received they dispatch them to the Packet Handler.

In order not to limit the system in terms of multitasking, each port needs to be run on one or more threads.

The logic for the Stop & Wait protocol should be implemented inside the port (or on top of it). It is used for the hop by hop reliability (remember that the sequence number is related only to the actual link).

Finally, it could be preferable to have some logic that provides support in case of a link failure.

### Forwarding Table Manager

The Forwarding Table Manager is in charge of updating the forwarding table and handling the timeouts for sending the update messages to the neighboring routers. Once an update message is received it needs to update the Forwarding Table and if there are changes it should propagate them passing a new message to the Packet Handler.

The logic to maintain the status of the connection among routers (HELLO packets) should be implemented inside this class.

### Forwarding Table

The forwarding table contains the most updated forwarding table. This object is not in charge of executing any action so it does not need to be run on a separated thread, but as it is a shared resource it requires particular attention on the modalities used to access the information contained in it (i.e. use semaphores).

### Packet Handler

The Packet Handler is the most crucial element of the system as it is in charge of dispatching all the packets that are circulating in the system.

There are two possible ways of implementing it: as a single threaded event list manager or as a multi-threaded manager. Even though the second possibility could increase the performances of the system we suggest implementing it as a single threaded element. In this case each packet received could be appropriately dispatched depending on its type (i.e. HELLO and UPDATE packets to the Forwarding Table Manager and Data Packets to the relative Port).

## Sender & Receiver

Senders and the Receivers are much simpler than the Routers and in particular they are limited to controlling the outgoing and incoming flows of packets. Figures 7 and 8 show the main objects that constitute them.

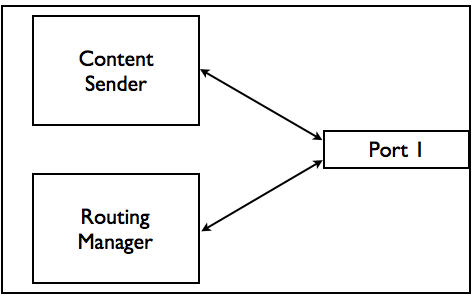


Figure 7: Sender objects composition

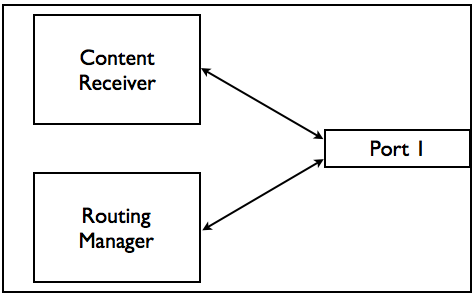


Figure 8: Receiver objects composition

### 

### Ports

Ports basically have the same behavior as the ones of the Routers; they maintain the queues of incoming and outgoing packets. Moreover they implement the logic for the link layer data link controls.

### Routing Manager

The Routing Manager is the element that maintains the status of the connection of the node with the rest of the network. In our particular case, the Routing Manager is responsible of handling the HELLO messages and the timeouts necessary to detect possible errors in the connection to the Router.

### Content Sender & Receiver

The main logic of the two network elements is located in this class. As we are considering a network with hop by hop reliability, these two components do not need to provide systems for retransmission or acknowledgment of packets; they do need to provide some logic to reassemble the transmitted data in case the delivery happens out of order.

## Final Comments

These guidelines provide a way of implementing the system described in this document but are not the only way for realizing it. Possible additions to what is written here could be:

* implementing a priority queue in the ports to prioritize routing packets that could be more important in the proper functioning of the network;
* having multiple packet handlers connect to the different ports to have an increased parallelization of the operations.

# Appendix:

End nodes send “Hello” packets every 30 seconds.

Routers send LS updates every 2 minutes.

Packet type values are given in the table below.

|  |  |
| --- | --- |
| **Packet type** | **Value** |
| Hello | 0x00 |
| Hello ACK | 0x01 |
| Data | 0x02 |
| LS update | 0x03 |