Message Dispatch/Relay within Network

Inter-Mediate: Message Store   5->5'Mailbox
Noice that the current node will detect it self have a matching id as the message's destination, and hence will add the message into the mailbox (AKA: Receiev Queue) awaiting for client or test program to pick up

RECEIVE

5

Termination : Message Receive
Test program sends out a receive call on the target node of the network (destination router), and wait there for a short period of time. Before the time run-out, the routers in the network should've relay the message and successfully put it into 'mailbox' (receive queue) of this target node.

6

Remainder of the network

SEND

1

Initiation : Message Send
Test program sends out a message at a node of the network (a router) with a pre-specified destination And the network have been running for a short period of time, which allows the routing table to be established

Inter-Mediate: Message Relay 1->2

4

Inter-Mediate: Message Relay   4->5

I am not the destination to arrive 4 ... first patch message to Router_4

2

Inter-Mediate: Message Relay   2->3

3

Look Up

Router_3's Routing Table

Router_...	.....	......
Router_4	Distance=1	NextHop=4
Router_5	**Distance=2**	**NextHop=4**
Router_6	Distance=3	NextHop=2
Router_...	.....	......

Relay(ing) Message

Sender: Router_1
Receiver: Router_5
The_Message: "Hello"
Hop_Count: int(2)

Inter-Mediate: Message Relay   3->4
Each node within the network get the message either through the client (test program) or other node, and will check if the message belongs to itself:
- If 'yes', then put it into the pending mailbox (receive queue) for later pickup from another client
- if 'no',   then using its local routing table to find the best router the message should be sent to (in order for it to arrive at the destination node), and relay the message to that router.

Legend

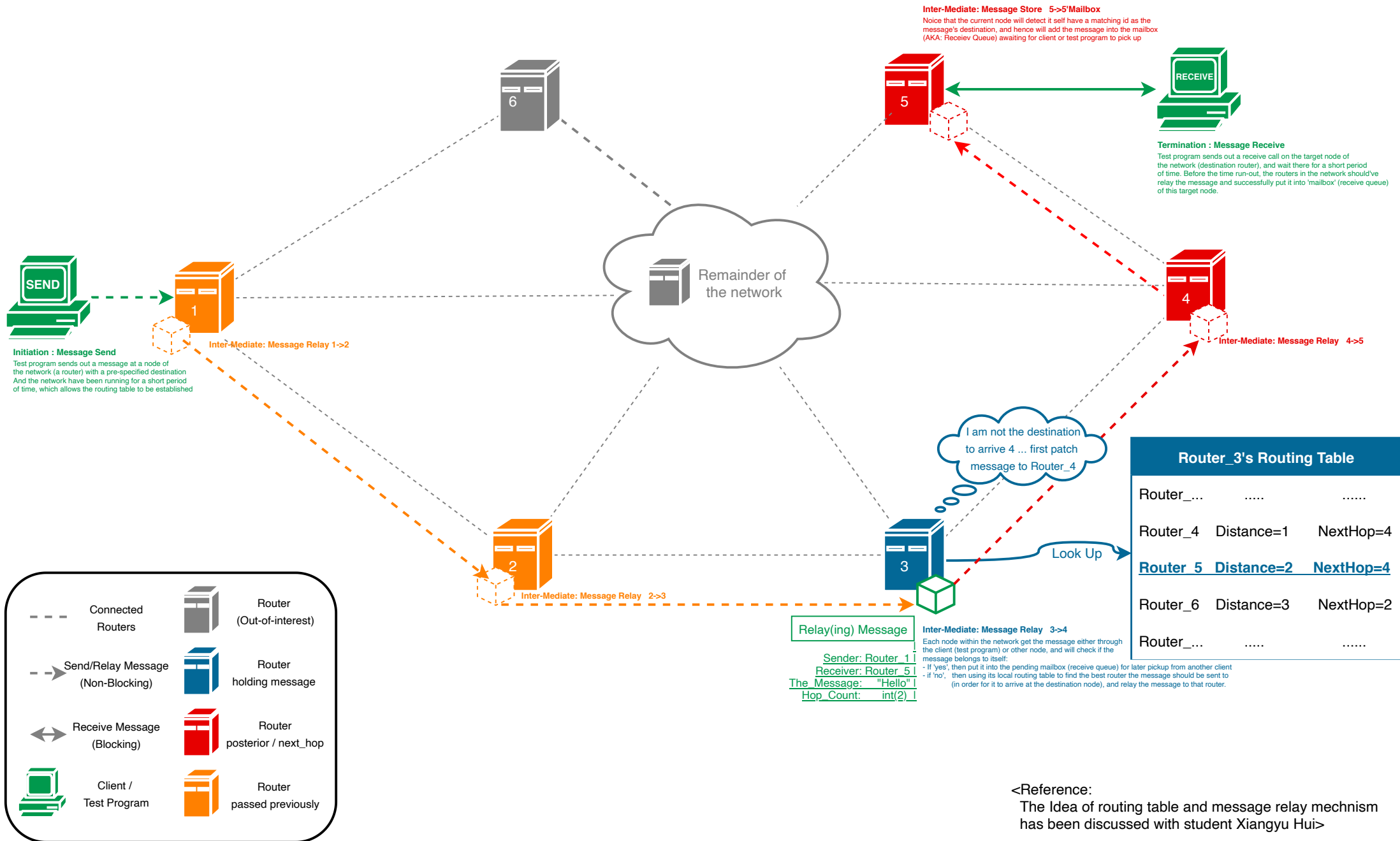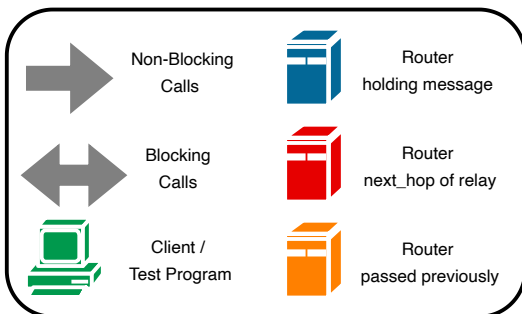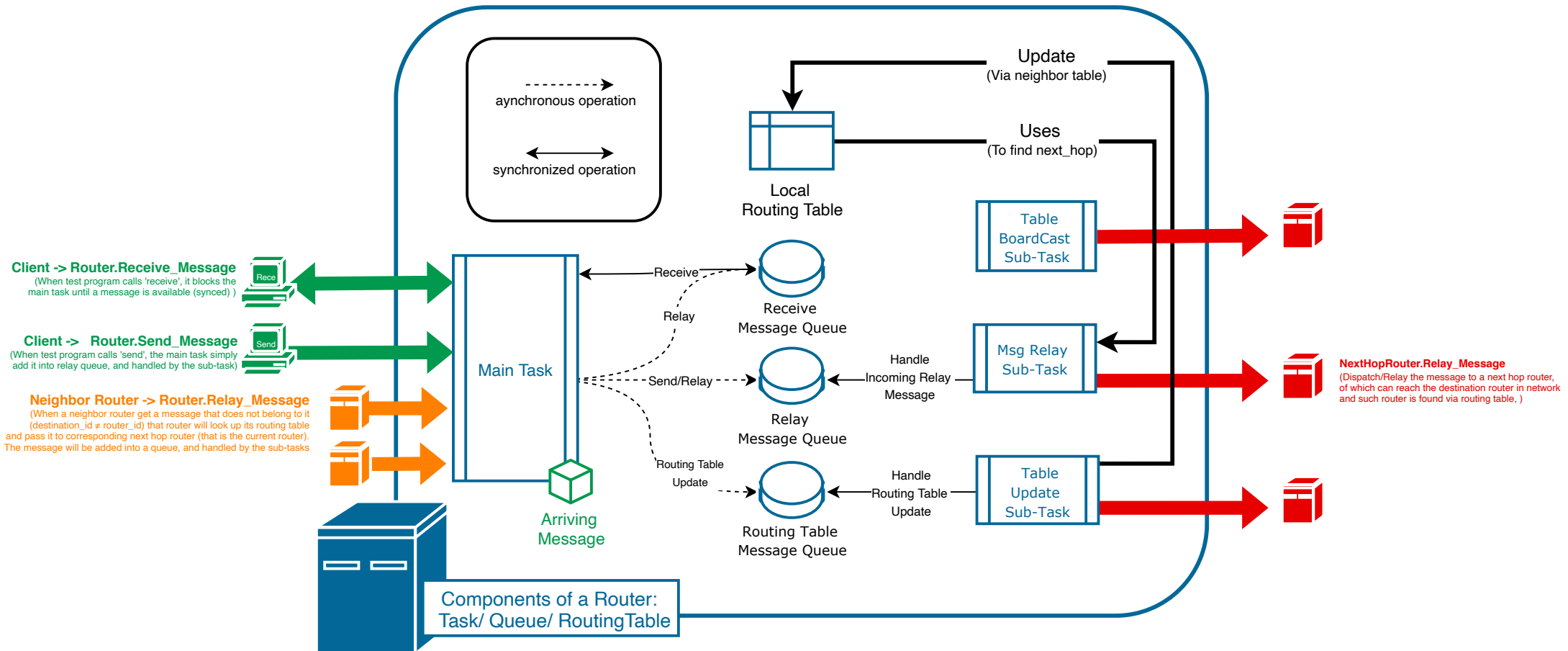|			
- - - -	Connected Routers		Router (Out-of-interest)
- -▷	Send/Relay Message (Non-Blocking)		Router holding message
◄──►	Receive Message (Blocking)		Router posterior / next_hop
	Client / Test Program		Router passed previously

<Reference:
   The Idea of routing table and message relay mechnism
   has been discussed with student Xiangyu Hui>

Process Calls & Router Components

ayncronous operation

synchronized operation

Update
(Via neighbor table)

Uses
(To find next_hop)

Local
Routing Table

Table
BoardCast
Sub-Task

Client -> Router.Receive_Message
(When test program calls 'receive', it blocks the
main task until a message is available (synced))

Receive

Receive
Message Queue

Client ->   Router.Send_Message
(When test program calls 'send', the main task simply
add it into relay queue, and handled by the sub-task)

Send

Main Task

Receive

Relay

Send/Relay

Msg Relay
Sub-Task

Handle
Incoming Relay
Message

Relay
Message Queue

NextHopRouter.Relay_Message
(Dispatch/Relay the message to a next hop router,
of which can reach the destination router in network
and such router is found via routing table,)

Neighbor Router -> Router.Relay_Message
(When a neighbor router get a message that does not belong to it
(destination_id ≠ router_id) that router will look up its routing table
and pass it to corresponding next hop router (that is the current router).
The message will be added into a queue, and handled by the sub-tasks)

Arriving
Message

Routing Table
Update

Table
Update
Sub-Task

Handle
Routing Table
Update

Routing Table
Message Queue

Components of a Router:
Task/ Queue/ RoutingTable

Non-Blocking
Calls

Router
holding message

Blocking
Calls

Router
next_hop of relay

Client /
Test Program

Router
passed previously

<Reference:
The Idea of multi-queue, multi-subtask is inspired
and has been discussed with student Kent Leung >

Routing Table Algorithm

Time-0: Initialization

To begin with, each table is only consisting of the router itself, as well as its neighboring node. Each role of the table uniquely represent the the neighboring router the current router should forward the message to, in order for the message to arrive at the destination node.
(An illustration on what each element in a row represent is attach at the bottom left conner)
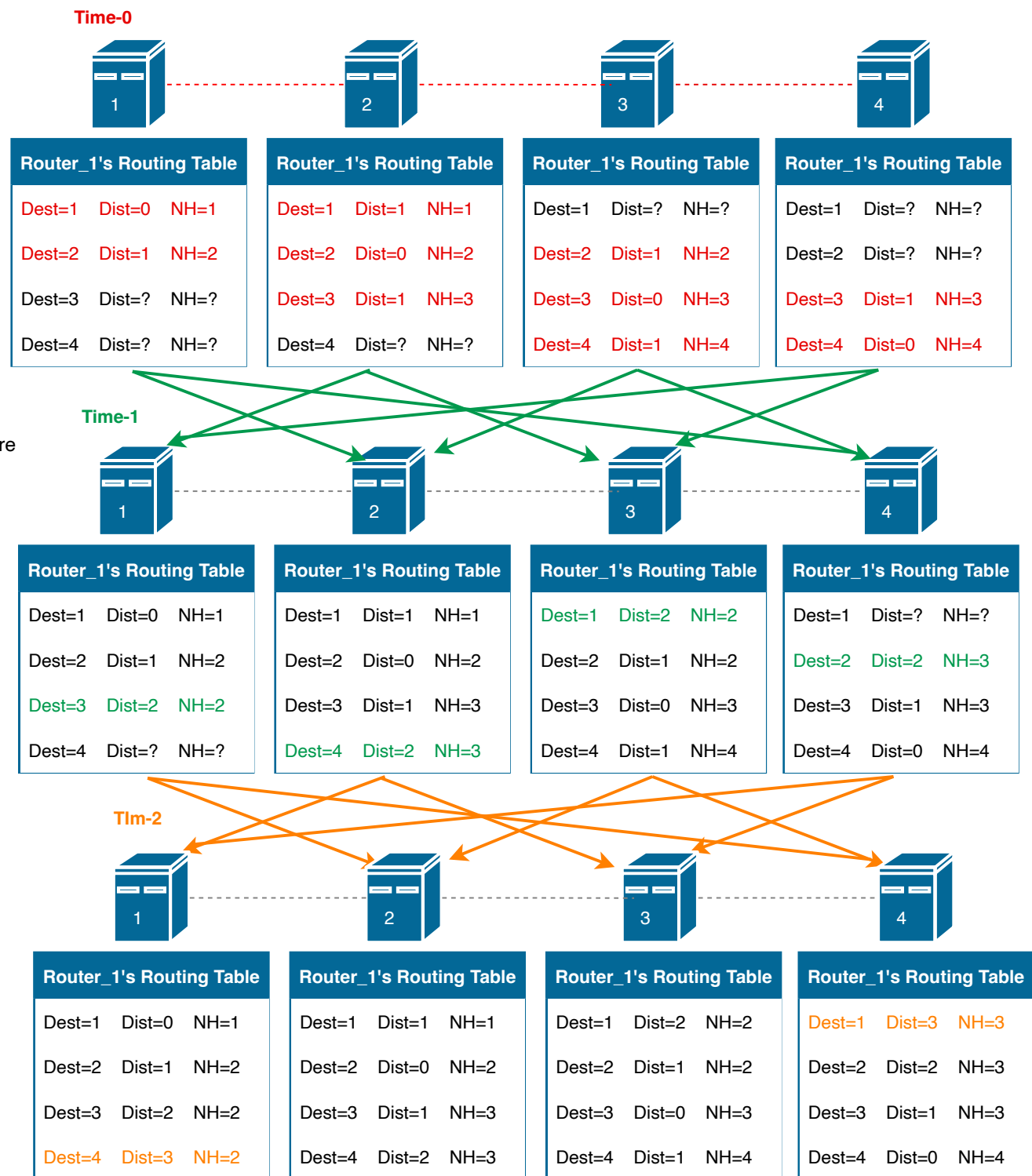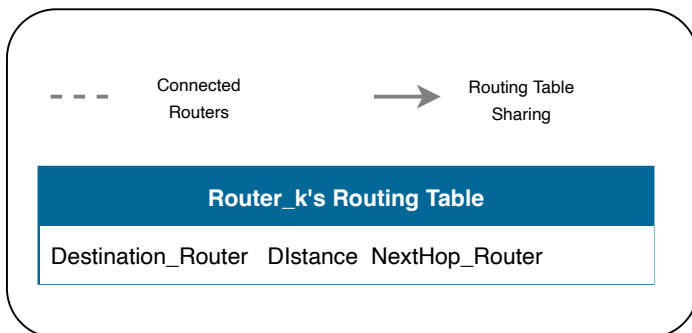
Time-1: BoardCast - Triggered by Timer

After initialization, each router will board cast its local routing table to all of its neighboring node, potentially brining new routing information to the neighbors. Each router receiving such shared routing table will compare it with the local table, and update any row that has unknown next hop, or is uses a longer route (larger hop distance).

Time-2: BoardCast - Triggered by Update

Every once the router update the table, it will again, board cast its new table. As you might have noticed, in each update, a router will gain info about another router in the network, that's 2 hops further away. By keep updating the routing table, eventually the tables will converge, and each router will have a "next hop" for every destination with shortest distance.

Usage

In order for a router (say Router_3) to send message to another router (Router_1), it will look up the corresponding row (Row_1) of the routing table, and find the next hop router, and send it.
(In special cases where the row does not have a valid next hop, the message relay will be postponed)

Legend:
- - - - Connected Routers
- → Routing Table Sharing

Router_k's Routing Table

Destination_Router	DIstance	NextHop_Router

Time-0

Router_1's Routing Table

Dest=1	Dist=0	NH=1
Dest=2	Dist=1	NH=2
Dest=3	Dist=?	NH=?
Dest=4	Dist=?	NH=?

Router_1's Routing Table

Dest=1	Dist=1	NH=1
Dest=2	Dist=0	NH=2
Dest=3	Dist=1	NH=3
Dest=4	Dist=?	NH=?

Router_1's Routing Table

Dest=1	Dist=?	NH=?
Dest=2	Dist=1	NH=2
Dest=3	Dist=0	NH=3
Dest=4	Dist=1	NH=4

Router_1's Routing Table

Dest=1	Dist=?	NH=?
Dest=2	Dist=?	NH=?
Dest=3	Dist=1	NH=3
Dest=4	Dist=0	NH=4

Time-1

Router_1's Routing Table

Dest=1	Dist=0	NH=1
Dest=2	Dist=1	NH=2
Dest=3	Dist=2	NH=2
Dest=4	Dist=?	NH=?

Router_1's Routing Table

Dest=1	Dist=1	NH=1
Dest=2	Dist=0	NH=2
Dest=3	Dist=1	NH=3
Dest=4	Dist=2	NH=3

Router_1's Routing Table

Dest=1	Dist=2	NH=2
Dest=2	Dist=1	NH=2
Dest=3	Dist=0	NH=3
Dest=4	Dist=1	NH=4

Router_1's Routing Table

Dest=1	Dist=?	NH=?
Dest=2	Dist=2	NH=3
Dest=3	Dist=1	NH=3
Dest=4	Dist=0	NH=4

TIm-2

Router_1's Routing Table

Dest=1	Dist=0	NH=1
Dest=2	Dist=1	NH=2
Dest=3	Dist=2	NH=2
Dest=4	Dist=3	NH=2

Router_1's Routing Table

Dest=1	Dist=1	NH=1
Dest=2	Dist=0	NH=2
Dest=3	Dist=1	NH=3
Dest=4	Dist=2	NH=3

Router_1's Routing Table

Dest=1	Dist=2	NH=2
Dest=2	Dist=1	NH=2
Dest=3	Dist=0	NH=3
Dest=4	Dist=1	NH=4

Router_1's Routing Table

Dest=1	Dist=3	NH=3
Dest=2	Dist=2	NH=3
Dest=3	Dist=1	NH=3
Dest=4	Dist=0	NH=4

Task Flow Diagram

(RouterTask_TableUpdate)

Table Update Sub-Task

Start

Loop

Routing Table Message Queue

Table_Queue.Dequeue (Synchronous)

Variable "UpdMade" = False

For Each "Row" in Table

Compare Distance

Found Closer Distance — False → Continue

True

Update Next Hop Update Distance

Local Routing Table

Variable "UpdMade" = True

Continue

UpdMade == True — False

True

Board Cast Local Routing Table

Abort

(RouterTask_TableBoardCast)

Table BoardCast Sub-Task

Start

Loop

For neighbor router in port_list

Update local routing table using port list

Board cast local routing table

Local Routing Table

delay

Abort

(RouterTask_Relay)

Msg Relay Sub-Task

Start

Loop

Relay Message Queue

RelayMessage_Queue.Dequeue (Synchronous)

Find next_hop using Routing Table

Next hop found — False → (Re) Enqueue

Next_hop.(Main).Relay(MSG)

Abort

(begin in RouterTask)

Main Task

Start

Loop

Else

remote entry call — Shutdown → Abort all sub-tasks → Terminate

TableUpdate_Message

Send_Message

Relay_Message

Receive_Message

Routing Table Message Queue

Enqueue Table Queue (Synchronous)

Dequeue Mailbox (Synchronous)

Current router == Destination

True

False

Enqueue Relay Queue (Synchronous)

Relay Message Queue

Enqueue Relay Queue (Synchronous)

Enqueue Mailbox (Synchronous)

Receive Message Queue