2. D. Estrin, S. Deering, Dino Farinacci, Van Jacobson, Ching-gung Liu and Liming Wei “Protocol Independent Multicast (PIM): Protocol Specification”, working draft, Oct 1994.

3. C. Alaettinoglu, A.U. Shankar, K. Dussa-Zieger and I. Mata, “Design and Implementation of MaRS: A Routing Testbed”, Technical Report, CS-TR-2964, Department of Computer Science, University of Maryland, Sept 1992

3. C. Alaettinoglu, K. Dussa-Zieger, I Matta, A. U. Shankar, “MaRS (Maryland Routing Simulator - Version 1.0 User’s Manual”, Technical Report, CS-TR-2687, Department of Computer Science, University of Maryland. June 1991

4. C. Alaettinoglu, K. Dussa-Zieger, I Matta, O. Gudmundsson and A. U. Shankar, “MaRS (Maryland Routing Simulator) - Version 1.0 Programmer’s Manual”, Technical Report, CS-TR-2723, Department of Computer Science, University of Maryland. July 1991

4.0 The implementation status

The current simulator has implemented almost all PIM protocol mechanisms. At this moment, there are still two protocol related areas needing updates:

1. Dense-mode outgoing interface state handling. The current simulator only simulates two states of an outgoing interface for dense mode: pruned or in-oif-list. An interface is added to the outgoing interface list when the first data packet is flooded, and remains there until either (1) it is pruned off and turns into 'pruned' state; or (2) the whole entry is timed out after data pakcets stopped arriving.

   Since the specification also specifies that an outgoing interface in the oif list can be timed out when there is no data traffic, and each interface can potentially have different timers due to the scalable timer mechanisms, each outgoing interface in the oif list may be timed out at a different time. Hence in a dense mode entry, an interface may be in one of the following 3 states: Pruned, in-oif-list, Timed out. A pruned interface will time out and change to in-oif-list state. an interface in the oif-list will time out when there is no data packets being forwarded for a certain period of time.

   I am waiting for this part of the specification to stablize before changing the bi-state oif into tri-state oif for dense mode groups.
2. We will also update the rules used for asserts on multiaccess LANs.

The following is a tentative list of functionalities to be added to a future pimsim release:

3. Dynamic change of network topology. Add the instruments to allow modification of topology on the fly. This involves (a) rewriting of the global topology table; (b) recompute the unicast routing table for all nodes; (c) maintenance of existing network states;
4. GUI based pause action based on user-specified events. It was projected that most pimsim users would be programmers, this function is now performed in a C debugger such as gdb or dbx;
5. Automatic loop detection instruments. This involves addition of mechanisms to keep the state for all data packets passing-by and to recognize a packet going through the same interface twice or more times.

5.0 Acknowledgment

The author would like to thank the authors of MaRS for providing such a useful routing test-bed, and Deborah Estrin, Lee Breslau for their helpful suggestions and comments.

6.0 References

1. D. Estrin, S. Deering, Dino Farinacci, Van Jacobson, Ching-gung Liu and Liming Wei "Protocol Independent Multicast (PIM): Motivation and Architecture" , working draft, Oct 1994.

silently dropped. However if any interface is dense-mode configured, it call the dm_pim_data_action() function to flood the packet according to the dense-mode specifications.

The SPT-flag handling (when doing RPT to SPT switch) is also done here.

3.5 Link

The link component is based on that supplied with the MaRS package. The link component in pimsim differs in the following:

- It has a new type parameter: link_subtype, which can be point-to-point or multi-access.
- link_start() would not reject the setup of more than 2 attachment points for a multi-access link.
- It has a new mode parameter: smdm_mode, which can take the value of either Sparse-mode or Dense-mode.
- In link_send(), if it is a multiaccess link, it will only check one packet queue for packets. For a point-to-point link, still use standard MaRS stuff.
- link_receive() needs to deal with multi-access LANs and the chained (PIM) packets for different receivers.

3.6 Multicast Source

A source component has to be attached to a node component. The Mcast_source_action() routine contains all standard MaRS functions, plus the following PIM functions:

- Mcast_set_rp(), sets the RP address for this source.
- Mcast_produce_register(), generates PIM register messages toward the RP.
- Mcast_src_receive(), receives any PIM messages (e.g. PIM-registor-stop).
- Mcast_refresh(), handles PIM-register refreshes
- Mcast_trfc_timer_expire(), processes all timer expirations of the multicast source component.

3.7 Multicast Receiver

A multicast receiver component also has to be attached to a node component. It is implementated and treated as if it is just another interface of the node --- it will appear as an oif in the proper multicast routing table entry and packets will be forwarded to it. The difference from a real interface (or link) is that it terminates all packets received. The Mcast_sink_action() routine contains the following PIM functions:

- Mcast_set_rp()
- Mcast_produce_join(), produces PIM (*,G) join messages. If this receiver has it SPT-bit flag set, it will switch to SPT when it receives a data packet.
- Mcast_receive(), receives RP-reachable message, discards it.
- Mcast_refresh(), refreshes the joins.
- Mcast_trfc_timer_expire(), Processes expired timers for this receiver.

3.3.2 pim_processing()

Pim_processing() processes all incoming PIM routing messages. It consists of a big switch based on the packet type and calls one of the following packet processing functions: pim_join_prune_processing(), pim_rcv_register(), pim_rp_reachable_proessing() and pim_rcv_query().

3.3.3 pim_refresh()

pim_refresh() periodically calculates the join/prune messages and send them toward the right neighbor nodes. It first calculates the number of entries relevant to each interface, then reserve the memory space for each interface. It constructs the join/prune blocks for each multicast group. Then all packets are sent out to the respective interface. Note that for a multi-access LAN with more than 2 nodes attached, the pim-refresh() may need to construct a distinct PIM join/prune packet for each of the other attached node. As an optimization, all PIM join/prune packets destined to different next hop nodes on a LAN are chained together and sent to the multi-access LAN with one event --- after all, they were generated at the same time. Another PIM_REFRESH event is scheduled in the end.

3.3.4 pim_periodic_query()

pim_periodic_query() periodically sends PIM query message out all interfaces. When a PIM module receives a PIM query message, it will mark the interface which it received the message as a NON_LEAF network. If a router hasn't heard a PIM query message from an interface for 3 query intervals, it sets that interface to LEAF status.

If a pim module is configured as 'dead' or changed to the status of 'dead' during the simulation, it will stop generating the periodic query messages.

3.3.5 pim_timer_expire()

It processes all timer expiration event related to a particular PIM routing module instance. The timer types include oif timer, RP_REACHABLE timer and SOURCE timer.

All of the above functions (subsection 3.3.1 --- 3.3.5) are called from the pim_action() routine.

3.4 Node

The node component is based on the node component supplied with the MaRS package. The node module in pimsim has the following changes:

- The nd_reset() routine for pimsim also puts the attached multicast receivers and multicast senders in its local topology table.
- The node component has counters that count the numbers of multicast data packets sent and number of data packets dropped.
- It marks the prev_comp field with its own address in a packet, so that the link knows who sent this packet.
- If the packet is a multicast data packet, the node would lookup the multicast routing table. If an entry is found, it would forward according to its oif list. If there is no entry found, and all interfaces are sparse-mode configured, the incoming packet is

The maximum number of nodes supported by PIMSIM can be increased or decreased by changing the value of MAX_NO_OF_NODES include/route.h file (current default to 25). The default maximum number of links attached to a node is 10. This can be changed by editting the value of MAX_LINKS_PER_NODE in include/component.h.

3.1 The initialization of the global topology table

After finishing loading the configuration file, the smulator would initialize the adjancency matrix adjancency-matrix_structure_name, and calculate the all-pair shortest pathes for the network. The structure of the adjancency-matrix. Installation of the local topology table.

3.2 Packets

The simulator recognizes two packet types: multicast data packet and PIM routing packets. These packet types are defined in a MaRS compatible way.

For PIM join/prune messages, the variable length address blocks which contain the join/prune lists are stored in the packet payload area, i.e. in the area pointed to by field ‘tail’. A new Packet field ‘tailsize’ is added to indicate the size of the variable join/prune address list.

For all packets, the pk_prev_comp field points to the last component that sent this packet. This field is used by the multi-access link component. A multi-access link would produce a duplicate copy of the packet for all attached nodes except the previous one which generated this packet.

Other packet types, such as MTRACE_REQUEST and MTRACE_RESPONSE are defined in the packet.h file for reference and to facilitate extensions. They are not recognized by any modules in pimsim now.

3.3 PIM routing module (Pimt)

The pim routing module consists of the common MaRS functions and following mainly PIM related C funtions: pim_start(), pim_processing(), pim_refresh(), pim_periodic_-query() and pim_timer_expire().

3.3.1 pim_start()

Pim_start() initializes the particular PIM instance and its multicast routing table plus various counters. It initializes its cache about all interfaces to LEAF status, and each such interface would remain in the leaf network status until a PIM query is heard from it.

Pim_start() also initiates the periodic PIM_REFRESH [1], and PIM_SEND_QUERY events.

1. This makes the code simple to understand and with very little more overhead. To avoid starting the PIM_REFRESH event here we can do it when the first new multicast routing entry is established.

in the receiver configuration, the receiver will send (S,G) joins in response to new data packets.

2.6 Rendezvous Point (RP)

As stated in the PIM protocol specification, any PIM router in the simulated network can act as an RP. It does not need any prior configuration.

2.7 Multicast Data Packets

Multicast data packets are sent from the multicast source components and are only processed by the node components. It is used to simulate the data-driven events, such as RP tree to SPT switch, dense-mode flood and prune, resolution of duplicate packets and routing loops over multiaccess LANs.

2.8 PIM protocol (routing) packets

Look at the packet.h file for the list of all relevant message types. Currently, it recognizes the following PIM message types (corresponding to ‘code’ values in the spec):

- PIM_QUERY
- PIM_REGISTER
- PIM_REGISTER_STOP
- PIM_JOIN_PRUN E
- PIM_RP_REACHABLE
- PIM_ASSERT
- PIM_GRAFT
- PIM_GRAFT_ACK

Each time a PIM join/prune message is generated, it is assumed that the message size can be arbitrarily large --- only one packet is generated for arbitrarily many groups and source/RP addresses.

3.0 Implementation Details

This section describes in more detail the operations of each component and how the simulator initializes the important data structures and how components interact with each other.

Pimsim keeps one copy of the global topology table --- the adjancency and shortest path matrix. All local topology tables are filled in when the simulator is started. Two standard routines rpf_inf() and rpf_nnd() are provided to calculate the reverse-shortest-path next hop interface and next-hop node.

PIM routing modules, Multicast sources and sinks all have the standard portions of MaRS parameters.

2.1 Unicast routing

Since PIM is independent of unicast routing protocols and only uses the shortest path information from the unicast routing table, no unicast routing protocol is simulated. The unicast routing table is precomputed at simulation startup time, and ‘statically’ installed. When simulating topology changes, the simulator directly recomputes the global routing table and recalculates unicast routing tables on all nodes. Subsequent PIM protocol actions will get the new unicast routes.

2.2 Node

A node stores and forwards packets according to its unicast routing table. When a protocol (control) message is received, it is immediately passed to its attached PIM routing module. When data packet is received on a node with DM links (interfaces), the packet is fowarded according to the multicast routing entry if there exists one, or it will be flooded by the node component and correct protocol state will be installed.

2.3 Link

A link can be point-to-point type, or multi-access type. For example, ‘lk1-3’ in figure 1 is a point-2-point type link , but ‘lk7-10’ is a multi-access type link. When a packet (both control and data) is sent to a multi-access LAN, it is duplicated in the link component and delivered to all nodes attached to the link component. A link can also be configured as sparse-mode (SM), or dense-mode (DM).

2.4 PIM routing module

Each node inside the network must have a PIM routing module attached. All arriving PIM mesages are processed by this module. It updates the corresponding multicast routing entries, and computes/generates PIM messages to its neighbors periodically or on event-triggered basis.

There is a data structure (Pimt type) associated with each PIM router. It contains the unicast routing table, multicast routing table and local topology table in addition to other configuration information. Please refer to the comments in the header files under the include directory for further details of the PIM router and multicast routing table data structure.

2.5 Multicast source and multicast receiver

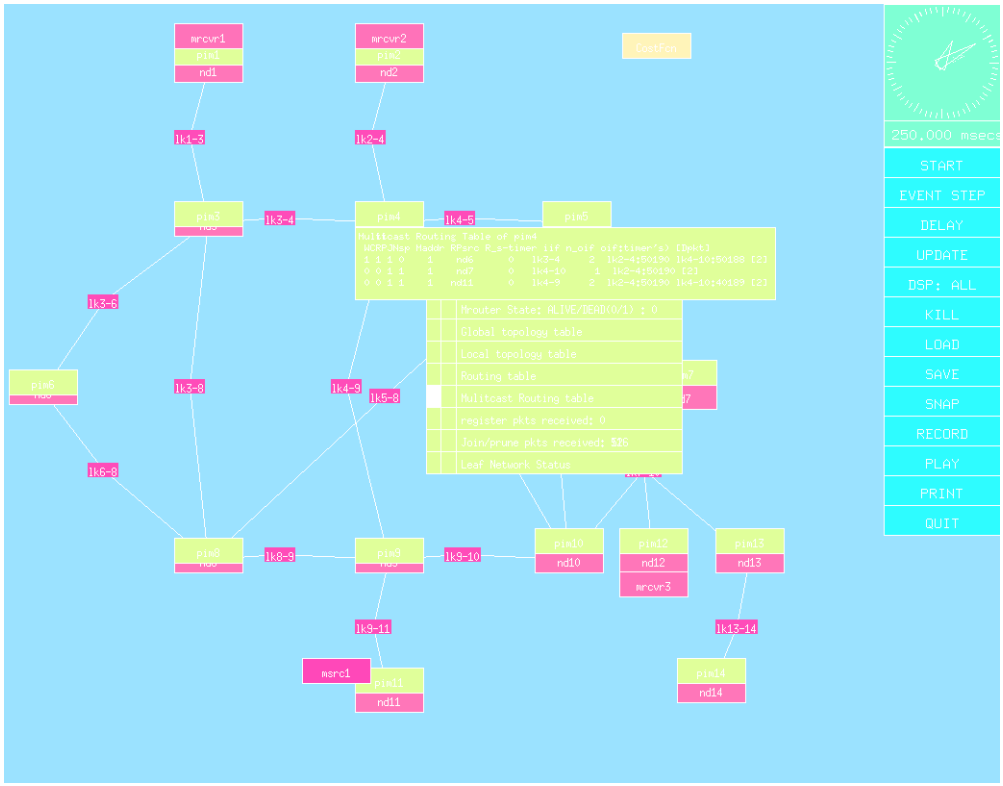
The multicast source module and multicast receiver module must be attached to a node component (instead of a link component). Each multicast source generates PIM register messages and directly sends them to the RP[1]. Each multicast sink generates periodically JOIN messages and sends them to its attached node component.

Each multicast receiver can be configured to either stay on the RP tree or switch to the SPT when data packets from new sources have been received. When the SPT-flag is set

1. RP see section 2.6 for RP actions.

Please refer to the ‘demo’ file for the syntax of configuration statements. To begin simulation, middle-click the ‘START’ button on the righthand side of the window. The simulation can be paused by middle-clicking the ‘CONTINUES’ button, which changes the simulator into ‘EVENT STEP’ mode. To inspect the status/parameters of a component, middle-click that component. Figure 2 shows a simulator window displaying status for multicast router ‘pim4’. Middle-clicking the left-most box in the row labelled “Multicast Routing Table” will bring up another window showing the multicast routing table entries in pim4. See figure 3.

FIGURE 3. Middle-clicking the left most box on “Multicast Routing Table” shows the multicast routing table



2.0 Simulating Components of a Network

This section describes all components/entities that are simulated by pimsim and their functional behaviors in the simulator. The simulated components include node, link, PIM routing module, Multicast source and multicast receiver.

FIGURE 1. Sample Network topologies (file ‘demo’) and components.

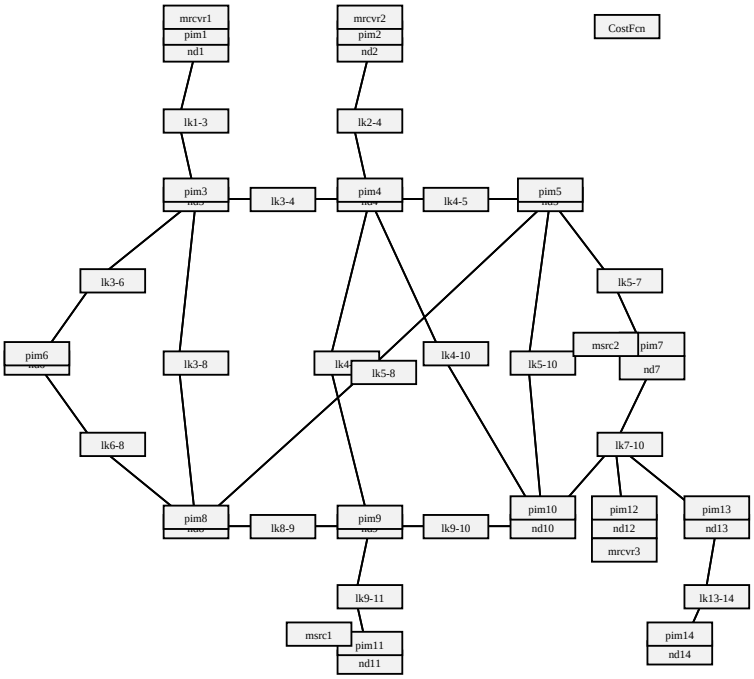
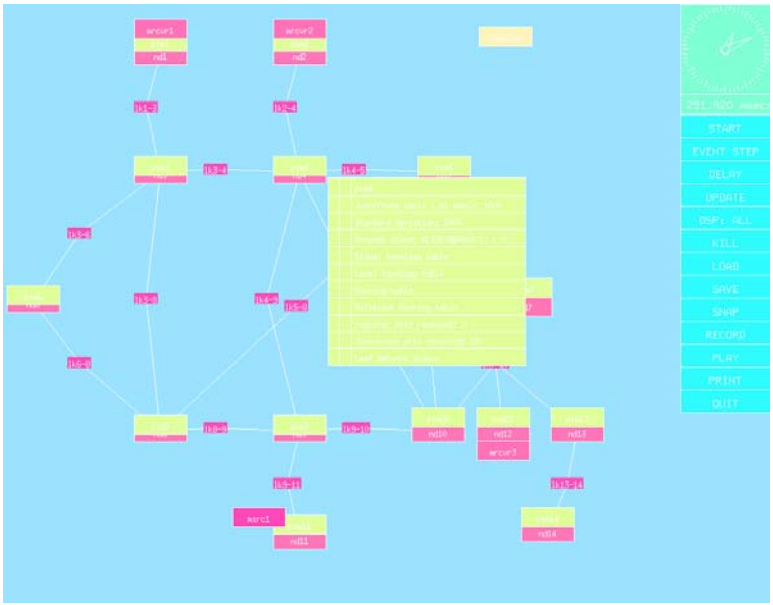


FIGURE 2. Status display of PIM router ‘pim4’



The Design of the USC PIM Simulator (pimsim)

Liming Wei

A user's guide and a document about the Implementation of the simulator.

The USC PIM simulator is an integrated Sparse Mode and Dense Mode PIM[1][2] simulator. It simulates the detailed protocol actions over both point-to-point and multi-access networks. All links can be configured as dense mode or sparse mode. It is based on the MaRS (Maryland Routing Simulator) package which provides a nice X-based user interface and an efficient event queue handler. The topology and components of the network can be displayed graphically on screen. The status of various parts of the network can be examined by middle-clicking the components or other appropriate places.

This document only deals with the PIM simulation related parts. For more details about the user interface, and the event-driven simulator engine, please see the MaRS architecture[3] and user documents[4].

The PIM simulator package can be obtained via anonymous ftp on: catarina.usc.edu:/pub/lwei/pimsim.tar.gz. MaRS is available from ftp.cs.umd.edu:/MaRS. It is recommanded that the machine running this simulator have at least 24Mbytes of memory.

1.0 Starting the Simulator

The tar file has an executable file 'pim' for the sun4 architecture and a sample configuration file 'demo' under subdirectory 'etc' for a 14-node network. To sun the simulator with the sample configuration, type the following

```
% pim etc/demo
```

To see all available command line options, type 'pim -h' on the command line. Once the simulator is started, it should pop up a window showing the sample 14-node network and a column of control buttons with a clock on the right handside. The sample network topology is shown in figure 1.