

DESIGN OF A HIGH SPEED PACKET ROUTING  
SWITCH UTILIZING TRANSPUTERS

Technical Report CSE-92-06

Christopher Ward  
and  
Saqib A. Khan

Department of Computer Science and Engineering  
Auburn University, AL 36849-5347

# Design of a High Speed Packet Routing Switch Utilizing Transputers

Christopher Ward and Saqib A. Khan  
*Department of Computer Science and Engineering*  
*Auburn University, Al 36849-5347, USA*

**Abstract.** A high speed Transputer based packet routing switch to model a satellite node is described with an emphasis on the mechanics used. The node will form part of a test-bed used to study characteristics of a low altitude multiple satellite (LAMS) network. Each node is modelled using a Host PC and five Transputers which communicate with other nodes through laser-crosslinks and are designed in accordance with International Standards Organization's (ISO) Open Systems Interconnection (OSI) 7 layered network model. Communications are full-duplex and expected to be between 1-5 Mbits/sec. Each node requires four external channels through which it communicates with other nodes. These channels receive NPDUs (Network layer Protocol Data Units) and either accept these packets or route them to the next node. The Transputer network has been designed so that dynamic routing table loading can be performed.

The simulation is expected to behave as a M/M/1 queueing model. Internal threads will be used to report queue lengths and throughput. Four extra TRAMs will be connected to the external links to model external nodes and will contain a "monitor" process which gather statistics on mean delay, number of packets accepted, etc.

## 1. Introduction

This paper describes the design of a Transputer based packet routing switch which simulates a satellite node. The node will form part of a testbed used to study different characteristics of a low altitude multiple satellite (LAMS) networks. The testbed is composed of five nodes. All nodes are able to directly communicate with one another, and three of the five nodes will be equipped with laser transceivers. The testbed forms part of a three year SDI research project<sup>1</sup> and is currently under construction.

Section 2 describes the requirements for this testbed. In Section 3 the design of the testbed is given. An outline of the process organization is included. Section 4 discusses the testing configuration to be used.

## 2. Background

A ground based testbed is required to perform studies on a retargetable communications laser for use in a LAMS network (details of LAMS networks may be found in [1, 2]). The requirements for the testbed are to model a five node LAMS network configuration. Three of these nodes will actually use laser trackers to communicate with each other. The others will be "directly connected" to one another through link simulators.

<sup>1</sup> "A simulation testbed for a satellite computer network with laser cross-links," Strategic Defense Command, Prime Contract DASG60-89-C-0119.

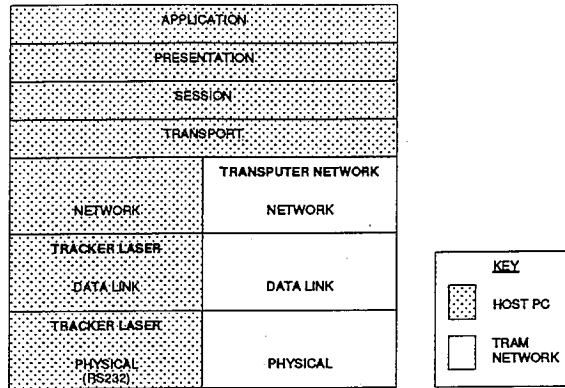


Figure 1 Testbed node ISO/OSI layer divisions.

These simulators permit a variety of link conditions to be investigated including burst errors, random errors, and link failure.

Since a node is modelled using a Host PC which cannot alone satisfy the high data rate requirements of the "directly connected" testbed node links, a high speed peripheral device with a PC interface is required. The high data rates and availability of high level language compilers (such as for the C language) makes the Transputer system an ideal choice for the Host PC's high speed data communications needs.

Testbed nodes are modeled using a Host PC connected to four RS232 serial boards and a Transputer network. The code for Layers 7-4 resides in the Host while the code for layers 3-1 resides in both the PC (for the order wire channels used to control the Tracker hardware) and the Transputer network (for the data communications). Figure 1 illustrates the node layer distribution. A comprehensive suite of functions was developed to explicitly model the significant ISO/OSI layers. Table 1 lists the C language functions that will be used to establish communication between the Layer 4 and Layer 3 processes.

Table 1 C language function suite for Transport and Network layers.

TRANSPORT LAYER ROUTINES	NETWORK CONTROL ROUTINES
GEN_PDU *gen_transport_from_net_control() int gen_transport_to_net_control(GEN_PDU *pdu) GEN_PDU *gen_transport_from_network() int gen_transport_to_network(GEN_PDU *pdu)	GEN_PDU *gen_net_control_from_transport() int gen_net_control_to_transport(GEN_PDU *pdu)
NETWORK LAYER ROUTINES	ROUTING ROUTINES
GEN_PDU *gen_network_from_transport() int gen_network_to_transport(GEN_PDU *pdu)	void *gen_calculate_routing_information(int algorithm_type) int gen_download_routing_information(int type, void *table) int gen_routing_information_lookup(int type, int destination)

The highly mobile nature of the nodes in a LAMS network requires periodic updating of routing tables. This "updating" is achieved by dynamically loading routing tables from the Host PC to the Transputer network. Layer 4 processes can provide this service by calling

### 3.2. TRAM Process Models

All process models make extensive use of threads to perform tasks. These threads communicate with each other through the use of queues or links (which behave in the same manner as queues). All threads monitor either their receiving link or input queue. If a signal is received indicating message arrival the thread is activated and removes the message from its link/queue, performs the necessary tasks on the message, and once again resumes its wait operation. Queues are identified in process models as rectangles, whereas processes are identified as circles.

Two separate process models are used to implement each node's Transputer network: the "Bridge" model above (Figure 3) and the "Router" model (Figure 4). The Bridge process model is resident upon the Bridge Transputer (see fig. 2) and the Router processes are resident upon the four external link TRAMs.

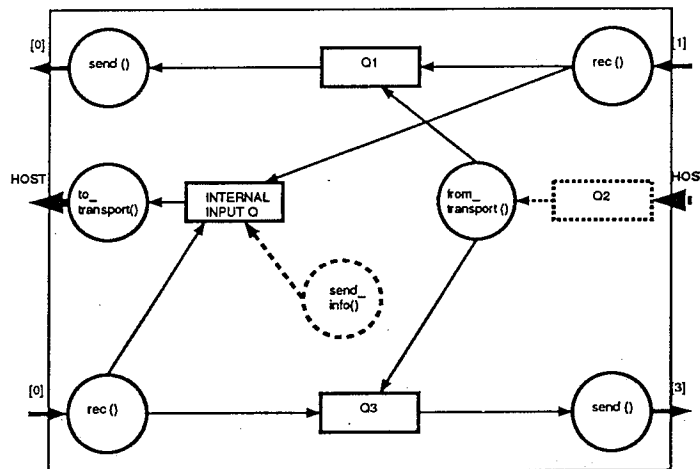


Figure 3 Bridge Process Model

specific library functions (see Table 1), however, the underlying Transputer network must be designed to accommodate these updates.

### 3. Transputer Network Prototype

#### 3.1. Node Overview

Figure 2 illustrates the Transputer network selected. This configuration was chosen in view of the connectivity constraints placed upon each Transputer due to the limited number of links available. One link from an individual TRAM (Transputer Module) is used to model each external *node's* link. The other three remaining links are used to connect all four TRAMs together, thereby providing a fully connected network. The connection between the Host PC and the underlying Transputer network is established by inserting a "bridge" TRAM between one of the packet streams connecting adjacent Transputers. This bridge TRAM acts as a transparent link between the adjacent TRAMs as long as no layer 3-4 communication is needed. If such a need arises, the bridge TRAM either inserts or accepts and delivers packets from the passing packet stream.

The "fringe" TRAMs (not connected directly to the "bridge") must route packets through the TRAMs directly connected to the "bridge" TRAM to reach the Host, and vice versa. The lack of direct connectivity between fringe TRAMs and the bridge is solved by introducing *internal* routing of packets within the Transputer network, as explained below.

The Transputer network's external channels receive NPDUs (Network layer Protocol Data Units) from other nodes and either accept these packets (i.e. deliver them to the layer 4) or route them to the next node (as computed by routing algorithms) via another external channel. Transparent dynamic routing table updates and I/O between layer 3 and upper layer processes is provided through the Network layer Service Access Point (N-SAP) using an internal packet format. This format allows NPDUs to be classified either as "data" (i.e. packets containing standard data) or "control information" (i.e. packets containing I/O information). Processes discern between control and data NPDUs and route the packets accordingly to their destination within or outside the Transputer network.

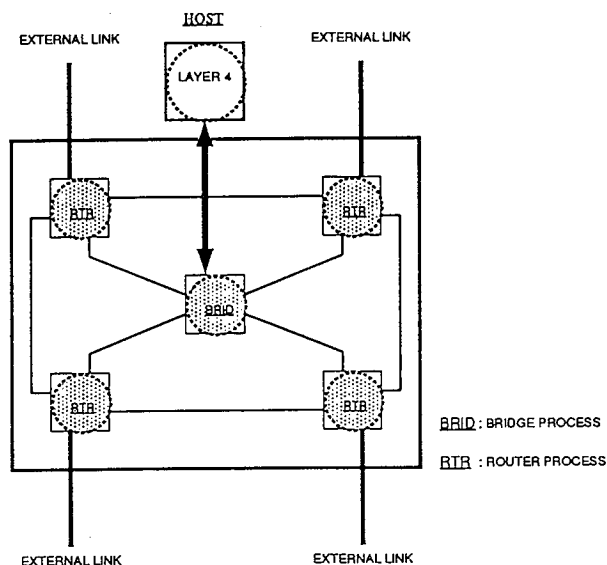


Figure 2 Transputer network configuration for a single testbed node.

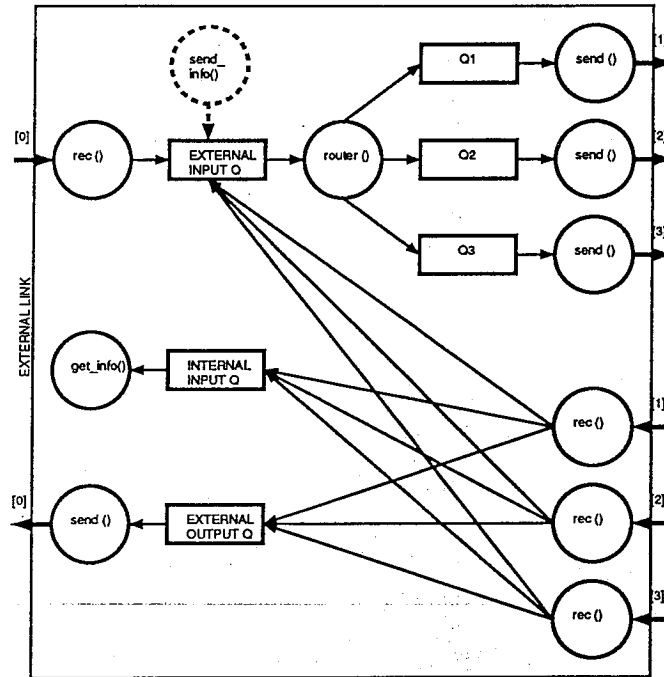


Figure 4 Router Process Model.

Both of these process models make extensive use of the “send()” (sender) and “rec()” threads. The external link connected “rec()” and “send()” threads bear the responsibility for the Datalink layer services. These threads will include code to deal with external link failures using a variety of FEC (Forward Error Correction) and ARQ (Automatic Repeat reQuest) algorithms (e.g. LAMS-DLC [2]). The basic psuedo-code for these processes is provided below:

```

process rec()
var
  header: HEADER;
  data: DATA_PTR;
  q: queue;
  port: PORT;

  {Compute dedicated queue & port}
  compute(q,port);

repeat
  {Allocate header and data}
  header:=allocate(sizeof(header));
  header:=receive_msg_header(port);
  data:=allocate(GET_SIZE(header));

  {Receive msg data}
  data:=receive_msg
    (GET_SIZE(header),port);

  {Add msg to Q # according to CTRL}
  if(GET_CTRL(header) = 'I')then
    add_to_q(q, header, data);
  else if(GET_CTRL(header) = 'E')then
    add_to_q(alt_q1, header, data);
  else if(GET_CTRL(header) = 'R')then
    add_to_q(alt_q2, header, data);
  else report_error(header, data);

until true
end

process send()
var
  header: HEADER;
  data: DATA_PTR;
  q: queue;
  port: PORT;

  {Compute dedicated queue & port}
  compute(q,port);

repeat

```

```

{Wait for packet in dedicated Q}
wait_for_packet(q);

{Remove header and msg from queue}
header:=remove_header_from_q(q);
data:=remove_msg_from_q(q);

{Xmit msg header and msg}
send_msg_header(header);
send_msg(data);

{Free header & data memory}
free(header);
free(data);

until true
end

```

The Bridge processes provide two services: accepting information from layer 4 and sending information to layer 4 (i.e. N-SAP). The principal processes are: "rec()" (receiver) processes which monitor all incoming packets and only if the packets are identified as intended for *this* node are they routed to "to\_transport()", otherwise packets are sent out through the corresponding "send()" (sender) process; the "to\_transport()" process which transfers data from layer 3 to layer 4; conversely, the "from\_transport()" process transfers data from layer 4 to layer 3. The "from\_transport()" accepts data directly from layer 4, decides upon the shortest route for the data to its destination (via the routing table) and injects the packet into the corresponding output queue on the Bridge.

The "send\_info()" process exists only for performance analysis purposes. It will be designed to periodically generate packets containing the TRAM's instantaneous queue length, throughput, etc.

The Router processes (shown in Figure 4) are similar to the Bridge processes. Packet routing is performed by dedicating "rec()" and "send()" threads to each physical link of the TRAM and routing received packets using the "router()" thread. The "get\_info()" thread is used to receive external routing table updates for modifying the TRAM's routing table. In future versions "get\_info()" might perform a variety of local tasks.

#### 4. Conclusions

This paper introduces the design of a Transputer based packet routing switch which when incorporated into a Host PC simulates a satellite node. The node will form part of a testbed used to study characteristics of a low altitude multiple satellite (LAMS) networks.

Prior to integration into the testbed, the Transputer network prototype (currently under construction) is being tested by connecting four separate TRAMs to the four external links of the switch. These TRAMs simulate other test-bed nodes by acting as packet sources and sinks. Each external TRAM gathers statistics on mean delay, number of packets accepted, etc. and reports these results to the Host at the end of a simulation run, thereby providing a capability to monitor the switch's behavior externally.

#### 5. References

- [1] Loren P. Clare, C. Y. Wang, and M. W. Atkinson. Multiple satellite networks: Performance Evaluation via simulation. Research project, Rockwell International Science Center, Thousand Oaks, California 91360, 1988.
- [2] Christopher Ward and Cheong H. Choi. The LAMS-DLC ARQ Protocol. In *Sigcomm'91*, Zurich, Switzerland, September 4-6 1991.