Fall'06-2
TelCom 2310
Computer Networks
Monday 3:00 pm - 5:50 pm
6110 Sennott Square
http://www.cs.pitt.edu/∼znati/tel2310.html

Project
Due Date : 12/6/2007

1 Purpose

The purpose of the project is to design and implement the exchange information component of a *Simple Routing Information Protocol* (SRIP) between *neighbor* routers. SRIP is used to advertize reachability information between two different types of routers in a network environment, namely **interior** and **exterior** routers. The routing and control information between the routers is achieved through Sockets in an Internet and Unix domain.

The project implementation includes the following components:

- The **configuration** component to set up the interior and exterior routers and their respective parameters,
- SRIP features and functionalities,
- The **failure emulation** component to "manually" inject packet errors and simulate link failures in a random fashion, and
- The **status report** component to display the status of different components of the systems, namely routers and links, and the current content of the routing table in each router.

The SRIP protocol main features include:

- Support for neighbor acquisition,
- Support for "alive" messages,
- Support for periodic exchange of routing information, and
- Support for mechanisms to deal with packet errors and link failures.

2 SRIP Messages

In order to accommodate the basic SRIP functionalities, the protocol defines three types of messages, namely neighbor acquisition messages, "alive" messages and routing update messages.

3 Neighbor Acquisition Messages

The neighbor acquisition messages are used by two adjacent routers to establish "neighborhood" for routing information exchange. This class of messages include *be_neighbors_request*, *be_neighbors_confirm*, *be_neighbors_refuse*, *cease_neighbors_request*, and *cease_neighbors_confirm*. The be_neighbors_request message is used by one router to invite another router to become neighbors.

In addition to the standard fields, the message contains initial values for the *hello interval*, the *update interval*, *router ID*, and any other parameter you see fit in the initialization process. The hello interval determines the amount of time a router waits before it tests if the neighbor is alive. The update interval determines the maximum frequency of routing updates. Additional parameters may include maximum packet length, if required, cost metrics, encryption keys, etc.

Upon receiving a be_neighbors_request message, the invited router can either accept the invitation, in which case it replies by sending a be_neighbors_confirm message, or reject the invitation by issuing a be_neighbors_refuse message. The choice of accepting or rejecting a neighborhood request is made by the system administrator. The neighborhood decision is initially made during the router configuration phase, but can be modified any time thereafter. A router may also refuse an invitation to become a neighbor if it does not run the same version of the SRIP protocol as the requesting router. In case of acceptance, the values of the initial parameters are negotiated during this phase to accommodate the profile of each router.

3.1 Alive Messages

The "alive" messages are sent periodically by one router to test whether the neighbor is still alive. The period is determined by the length of the *hello interval*.

3.2 Routing Update Messages

The routing update messages are sent periodically by a router to convey information about reachable networks to its neighbors. The routing information is used by the router to update its routing table. We will assume that the router uses a distance vector based implementation of the shortest path routing algorithm. Loops are dealt with using either the simple form of "Split Horizon", or the more aggressive "Split Horizon with Poison Reverse".

Each entry of the routing table contains the following information:

- The network identity,
- The number of hops to reach the network
- The address of the first gateway along the path to the specified network.

The basic update messages include :

- Tbl_Hdr Packet : This Packet carries the table header which contains the table sequence number. This number is updated by a router every time a new information is stored in the table.
- Tbl_Entry Packet = This Packet contains information stored in a table entry.
- Tbl_Ovr Packet = This Packet signals the end of transmission of a routing table.
- Tbl_Ack = This packet carries a positive acknowledgment of a previously transmitted packet.
- Tbl_Nck = This packet carries an negative acknowledgment of a previously transmitted packet.
- Tbl_Ftl = This packet signals a fatal error.

The protocol used by a router to transfer the routing table information is driven by the sender; the receiver simply acknowledges each packet it gets. If an error is detected, the receiver sends back a negative acknowledgment which causes the retransmission of the last packet.

The transaction begins when the sender transmits a Tbl_Hdr packet to indicate that a new routing table is to be transmitted. The sender indicates to the receiver the sequence number of the table to be transmitted. Upon receiving an acknowledgment of the Tbl_Entry packet, the sender proceeds to transmit as many Tbl_Entry packets as required, and waits either for a Tbl_Ack or a Tbl_Nck packet. Tbl_Ack and Tbl_Ncks are sent according to the window strategy used (i.e., stop-and-wait, sliding window, etc). A packet, Tbl_Ovr, signals the end of the transmission of the routing table. Finally, a Tbl_Ftl packet reports a fatal error and signals the end of the transaction.

3.3 SRIP Packet Format

- Ver: SRIP Version Number.
- Len: Total length of the packet.
- Seq: The sequence number of the packet (0¡=Seq¡64).
- Type: The type of the packet, as defined above.
- Data: The type of the packet, as defined above.
- FCS: Frame Check Sequence

4 Requirements

As a **minimum** requirement, SRIP must:

- Provide a user interface that can be used to properly configure the protocol version number, update and alive intervals, the neighbor acquisition authorization, timeout intervals, link failures and packet errors distribution parameters, and any other parameters as required by your design. The configuration phase is undertaken only once, at the beginning of the session.
  - The user interface should allow viewing of the information contained in the routing table both before and after routing updates.
  - The user interface should also allow the display of current status of a router's links.
- Handle exchange of routing and control information in typical configurations such as the one depicted in Figure 1. In this figure, the interior routers reside within the same host as their associated exterior router. The role of the interior router is to randomly generates new routes to a **predefined** set of IP address destinations and advertize these routes to the exterior router. The exterior router in turn updates its Routing Table and advertizes new routes to adjacent exterior routers. Notice that exterior routers do not advertize routes to interior routers. Furthermore, the same destination can be advertized by different interior routers with potentially different path lengths. Interior and exterior routes within the same host can communicate using either Unix or Internet type sockets.
- Implement at least a stop-and-wait based scheme to handle errors and flow control (a window-based protocol is a bonus).
- The implementation must be able to simulate link failures between routers and handle potential looping due to these failures, using a simple **split horizon** based approach. Split horizon is a method of preventing a routing loop in a network. The basic principle stems from the observation that the likelihood of loop formation can be reduced if information about routing carried by a particular packet is never sent back in the direction from which it was received.

5 Extra Credit

Split horizon can be achieved by means of a technique called **poison reverse**. This is the equivalent of route poisoning all possible reverse paths - that is, informing all routers that the path back to the originating node for a particular packet has an infinite metric. Split horizon with poison reverse is more effective than simple split horizon in networks with multiple routing paths, although it affords no improvement over simple split horizon in networks with only one routing path.

Route poisoning is a method of preventing a network from sending packets through a route that has become invalid. When the routing protocol detects an invalid route (such as can be caused by a severed cable or the failure of a network node), all of the routers in the network are informed that the bad route has a hop count of infinity. This makes all nodes on the invalid route appear infinitely distant, thereby preventing any of the routers from sending packets over the invalid route.

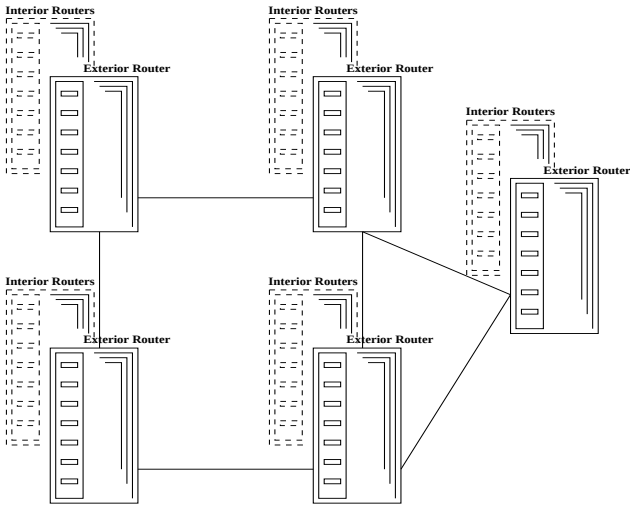

Figure 1: Basic Router Configuration

When the path between two routers in a network goes bad, all the routers in the network are informed immediately. However, it is possible for this information to be lost, causing some routers to once again attempt to send packets over the bad route. This requires that they be informed again that the route is invalid, and again, this information can be lost. The resulting problem is known as a routing loop. Route poisoning can be used in conjunction with **holddowns**. A holddown keeps update messages from falsely reinstating the validity of a bad route. This prevents routing loops, improving the overall efficiency of the network.

To secure **extra credits**, the project must include:

- The implementation of additional flavors of split horizons, namely poison reverse and holddowns.
- Comparative analysis of the different split horizons techniques.

6 Design Phases

The first phase of the project deals with the design of the protocol architecture. Prior to implementing the system, you must submit an initial design report. The report should include the following:

- A detailed description of the structure of the user interface. The design of the interface should allow the user to ask and obtain at least one level of help for each command provided at the interface.
- A modular decomposition of each component of the system, a description of the interface between each pair of components, and a justification of your design choices. Careful thought

must be given to the design to allow for new functionalities to be added to the system. The proposed architecture will be evaluated and either **approved** or **recommended** for modifications.

- The second phase of the project is concerned with the implementation of the approved design. As part of the final report, you are expected to discuss the program design and implementation in detail, and justify whatever design decisions you have made. Enclose a well documented *user manual* and a listing of the source code.

7 Due Dates

- The project Design is Due November 6, 2006.
- The project code and final report is due no **later** than December 6, 2006.

Every group will be asked to perform a demonstration of the program.