Performance Enhancing Proxy (PEP)

Models User's Guide

7.0

Contract DASW01 03 D 0008

Disclaimer: As of October 2007, NETWARS was redesignated by the Program Manager Office as the Joint Communications Simulation System (JCSS). JCSS was selected as the new industry name to better reflect the inherent joint communication capabilities of the software. Users should be aware that no software updates were conducted as part of the software name change.

January 7, 2008

Prepared for:
Defense Contracting Command - Washington
Washington, DC 20310-5200

Prepared by:
OPNET Technologies, Inc.
Bethesda, MD 20814-7904

OPNET®
Optimum Network Performance

Contents

List of Figures

List of Tables

1 Overview

This document describes key features, attributes, usage, and statistics of the Performance Enhancing Proxy (PEP) device models shipped as part of the NETWARS model. PEPs are used to improve the performance of the Internet protocols on network paths where native performance suffers due to characteristics of a link or sub-network on the path. The performance of the TCP suffers greatly when the physical medium is wireless, particularly when the product of bandwidth and propagation delay is high or the error rate is high. Some of the factors that affect the TCP performance over satellite are:

- Window Size – TCP window size (max 64 kilobytes) limits the traffic throughput over 250ms (one direction) satellite link to 128kbps.

- Acknowledgements (ACKs) – The hosts with smaller round-trip times have ACKs come back faster, thereby end up capturing most of the network bandwidth, at the expense of long-delay connections.

- TCP Slow Start – Due to the small initial window in slow start, a significant number of round trips may be required for the congestion window to grow large enough to effectively utilize the link bandwidth. Most data transfers over a satellite link can complete without ever having attained a window large enough for optimal link utilization.

- TCP Congestion Avoidance – In the congestion avoidance phase, window growth is much slower than in slow start. However, even a single loss results in halving the window. If a loss occurs after the window has grown quite large, the window is halved and the satellite link is under-utilized for a prolonged period while TCP recovers and grows its window back to the former size.

Both transport layer and application layer PEPs are in use. Examples of application layer PEPs include Web caches and relay Mail Transfer Agents, but only transport layer PEPs are considered here. A variety of PEP designs are possible in order to enhance performance of TCP over links. Techniques include TCP message interception (commonly known as *TCP-spoofing*) and Split Connections.

1.1 Intended Use

This document is intended for NETWARS users who have a reasonable level of knowledge about the PEP family of devices and their use in a Tactical network. This document is written to help the engineer learn about configuration of PEP devices in NETWARS and the types of interfaces, attributes, and statistics available in both the Discrete Event Simulation and Capacity Planning use cases.

1.2 Feature Summary

This section provides a list of the main features available in the PEP model.

Table 1: PEP Features

Feature	Description	Support
TCP Connection Splitting	Breaking a single TCP connection into two separate connections at the PEP	Supported
Spoofing	PEP using the IP addresses of each client as the source IP address when sending packets to the other client	Supported
PEP Enable/Disable	Capability of disabling the PEP functionality on a router so that it performs as a regular router	Supported
Any-to-Any Operation	Each PEP node can pair with multiple other PEP nodes (for separate TCP connections)	Supported
IP Layer Functionality		COTS IP Layer Features

1.3 Feature Description

The TCP PEP model simulates a basic Transport Layer PEP for Transport Control Protocol (TCP). This model will run in NETWARS, but can be adapted for standard OPNET Discrete Event Simulation (DES).

The main objective of this model effort is to provide a means to estimate the performance improvement enabled by such a PEP in an environment where the latency * bandwidth product is high. Each PEP device is a router node with the capability of enabling the PEP to become a PEP node. A PEP terminates all TCP connections coming from a client (call client A) to the node and opens new TCP connections to the destination client (call client B). Since the PEP uses the IP address of either of the clients as the source address when communicating with the other client, its existence and operation is transparent to both clients. Using two PEPs, every end-to-end TCP connection will be broken into three new connections and if the PEPs are at the two ends of a bottleneck link, the TCP parameters of the middle connection (PEP to PEP) can be optimized (large windows, SACK, etc.), independently of the TCP settings of the actual end points, to increase the performance.

These are the salient characteristics of the model:

- The model simulates a connection splitting PEP.
- The connections to be split are those for TCP running over both IPv4 and IPv6 (COTS IP).
- The model simulates packet compression (COTS IP).
- The model will simulate QoS per hop forwarding behavior based on the traffic class (or Diff Serv Code Point, RFC 2474) (COTS IP).

- The model will be capable of supporting an any-to-any configuration of PEPs.
- The model has the ability to enable or disable PEP function using a node attribute.
- The model has the ability to enable debug output using *ltrace*.

1.4 Limitations/Assumptions

This section lists known limitations of both the PEP technology and the proposed OPNET model.

1.4.1 Use with Encryption Devices

Section 4.1.1 of Reference RFC 3135 describes the security implications of using PEP devices. RFC 3135 notes that, in general, security applied above the transport layer can be used with transport layer PEPs. However, network (IP) layer security, though transparent to the application layer, will cause problems for PEPs, since the TCP headers must be unencrypted within each PEP. Since the envisioned implementation is a standard OPNET router with enhancements, this can be modeled, even though it may not be what one would want deployed in the real world (because of security considerations).

Other configurations can be modeled as well, such as placing encryptors between the PEPs and their associated satellite terminal equipment. This, of course, has the disadvantage of requiring the land-based portions of the network to be red.

The real world should be the guide concerning what is feasible and what is not. If it must be red in the real world, then it must be red in an OPNET network model.

1.4.2 Use with ACE

OPNET Modeler supports the use of packet analyzer nodes to capture packet trace output. This data can be used as input to the OPNET ACE tool, just as the output of ACE agents can be used. Although the use of PEP devices in the network will not interfere with the capture of traces, the fact that PEPs will spoof network addresses will confuse ACE.

Again, the real world should be the guide. In both the real world as well as in OPNET Modeler, the port and network addresses of packets captured at the ingress port will be identical to those captured at the egress port, and this will confuse ACE.

1.4.3 Topology

The “one-arm topology”, in which a PEP can be connected to a LAN by one Ethernet interface, is not supported. Only the pass-through topology will be supported. Also, a PEP must be the only gateway that connects the client to the Internet. In other words, all packets originated from or destined to the client must pass through the PEP.

1.4.4 Intermediate Protocol

The current version of the PEP gateway device uses TCP as the intermediate protocol. Since the actual real-world gateway may use SCPS-TP or some other, possibly proprietary, protocol, the model may not be high fidelity. It is not known how well TCP can model the characteristics of different intermediate protocols for a particular intended use of a modeling effort. Certainly, for example, it can't be used to compare the merits of different devices using different custom protocols.

1.4.5 Port-Specific Handling

At least one current real-world PEP implementation has port-specific behavior which is not modeled in the current PEP model. In particular, connection-splitting is not done for applications known to be chatty. These include CITRIX and HTTP.

1.4.6 Flow Control

In this version, there is no flow control in the PEP application layer. This means, for example, that data received from the client will buffer in the application layer until it can be forwarded to the server.

1.4.7 Out of Sequence Packets

Some real-world devices (e.g. iDirect) will forward out of sequence packets as they were received, but since we are using a complete TCP stack, we can't forward a packet until all prior packets have been received.

1.5 Model List

The standard NETWARS mode library contains two node models representing a generic TCP PEP device. One is an advanced model and the other is a standard model (derived).

Table 2: PEP Models

Model Name	LAN Ports	
	Serial	Ethernet
tcp_pep	2	2
tcp_pep_adv	2	2

1.6 Reference Documents

- RFC 793, "Transmission Control Protocol", 1981.
- RFC 1323, "TCP Extensions for High Performance", 1992.

- RFC 2474, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”, 1998.

- RFC 2488, “Enhancing TCP Over Satellite Channels using Standard Mechanisms”, 1999.

- RFC 3135, “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations”, 2001.

- RFC 3155, “End-to-end Performance Implications of Links with Errors”, 2001.

- “An Architecture for Internet Service via Broadband Satellite Networks” by Vijay G. Bharadwaj, John S. Baras, Norman P. Butt. International Journal of Satellite Communications, Vol. 19, Issue 1.

- “Performance Enhancing Proxies (PEPs) for DoD Satellite IP Modem Systems”, DISA-GE52/MITRE.

- Modeler Standard Models User Guide, Chapter 11, “IP QoS Model User Guide”, Product Release 11.5, OPNET Technologies, Inc.

- Modeler Standard Models User Guide, Chapter 9, “IP Model User Guide”, Product Release 11.5, OPNET Technologies, Inc.

- Modeler Standard Models User Guide, Chapter 25, “TCP Model User Guide”, Product Release 11.5, OPNET Technologies, Inc.

1.7 Additional Information

For additional information not provided in this document, consult the NETWARS Model Development Guide (MDG) or other NETWARS documentation. Information provided in the Model Development Guide includes interfacing with NETWARS models, using NETWARS traffic types, expected packet formats and interrupts, and suggested model development processes. NETWARS documentation is provided inside a standard NETWARS installation, and can be found in the **<NETWARS Installation Directory>\Documents** directory (where **<NETWARS Installation Directory>** is the directory where NETWARS is installed).

Also, for any questions regarding OPNET specific functionality, consult the standard OPNET documentation. This documentation can be accessed inside NETWARS from the **Help > Documentation > ITGuru Documentation** menu.

Finally, Promina files can be located in the **<NETWARS Installation Directory>\Sim_Domain\op_models\netwars_std_models\pep** directory. These models and files can be accessed and modified using the OPNET Modeler software.

2 Configuration

2.1 PEP and TCP Parameters

The TCP PEP is configured exactly like a generic router except for a single attribute "PEP.PEP Enable". When the attribute is disabled, the node acts exactly like a router. When enabled, all TCP connections routed through the device are split and the addresses are spoofed. The two sub connections are negotiated according to the "TCP.TCP Parameters" attribute for the device.[1] In addition compression is set as an attribute of the interface, under "IP.IP Routing Parameters.Interface Information[].Compression Information"

Table 3: Attributes of TCP.TCP Parameters

Attribute	Comments
Maximum Segment Size (bytes)	
Receive Buffer (bytes)	The data rate will be limited to size / round trip time.
Slow Start Initial Count (MSS)	
Window Scaling	Must be enabled, otherwise, throughput will be limited.
Selective ACK (SACK)	Recommended to be enabled
Timestamp	Supports SCPS-TP 6.2.2.3.3 requirement.
Acceleration	Supports SCPS-TP TAO (TCP Accelerated Open)

[1] The ideal arrangement would be to allow configuration of parameters on a per-interface basis, but up until the present, there is only one TCP process per node, and the actual attributes for each connection result from negotiation when the connection is set up.

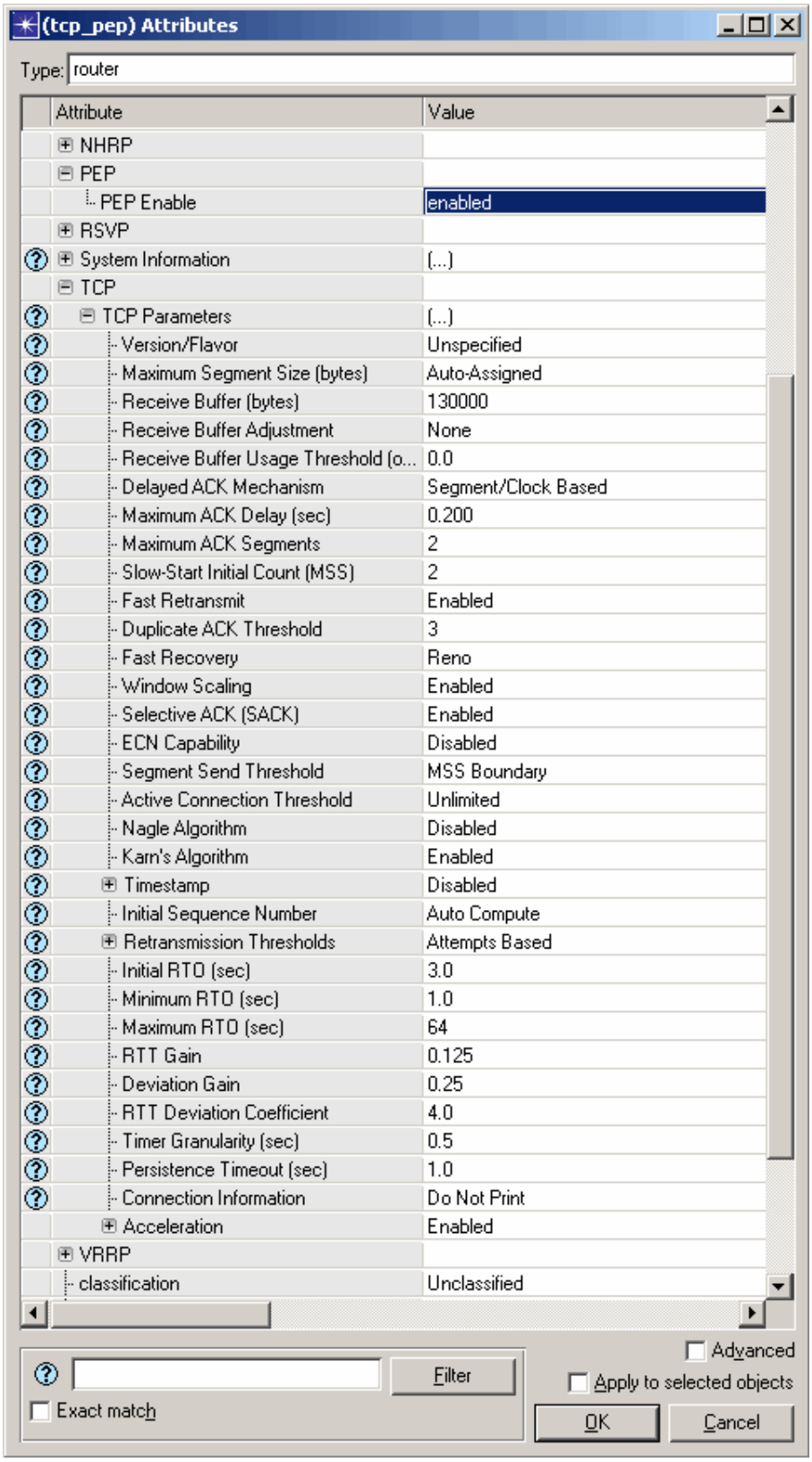

Figure 1: TCP.TCP Parameters

2.2 TCP Accelerated Open

Under certain circumstances, it may be advantageous to enable TCP Accelerated Open (TAO). TAO is a primary feature of the SCPS-TP extensions to TCP and is described in RFC-1644. If the traffic consists of a lot of short transactions, consisting of opening a TCP connection to exchange a single command and response, then TAO may reduce the total transaction time by one round-trip time, e.g., 250 ms for a satellite connection. Otherwise, the use of TAO is not indicated.

The "Acceleration" attribute under "TCP.TCP Parameters" is a compound attribute, with two attributes, both of which should generally be enabled at once.

"Acceleration.Accelerated Open" enables the TCP Accelerated Open. A host with TAO enabled is completely interoperable with hosts without this feature.

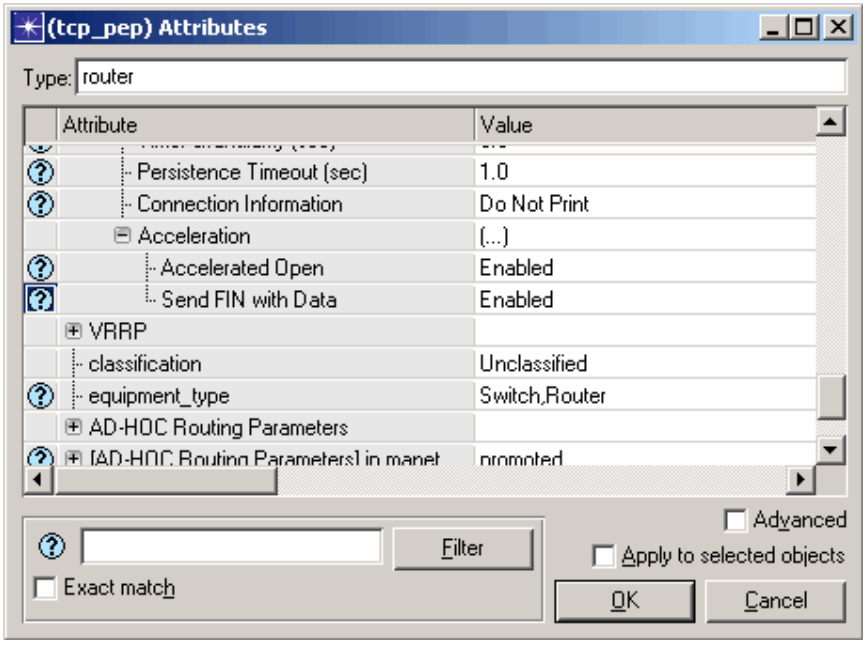

Figure 2: TCP.TCP Parameters.Acceleration

"Acceleration.Send FIN with Data" enables the FIN flag to be sent with a data segment. Most TCP implementations always send a FIN in a segment with no other data.

This feature can be used to reduce the number of packets sent.

3 Usage

The PEP model allows you to model networks which utilize TCP-level Performance Enhancmement Proxy devices. This includes typical configurations where two PEP devices are placed on the two ends of a satellite, and each performs PEP functionality for all TCP connections that come from their corresponding networks and pass through the satellite link. In this configuration, even though the TCP parameters of endpoint machines are not optimized for the satellite link, the PEP TCP parameters will be tuned to maximize the throughput on the satellite link. Figure 3 shows an example NETWARS scenario with this configuration. The middle link, between the PEP nodes, doesn't have to be a satellite link. Moreover, the middle link may include encryptor devices or Promina circuits as explained in section 3.4.

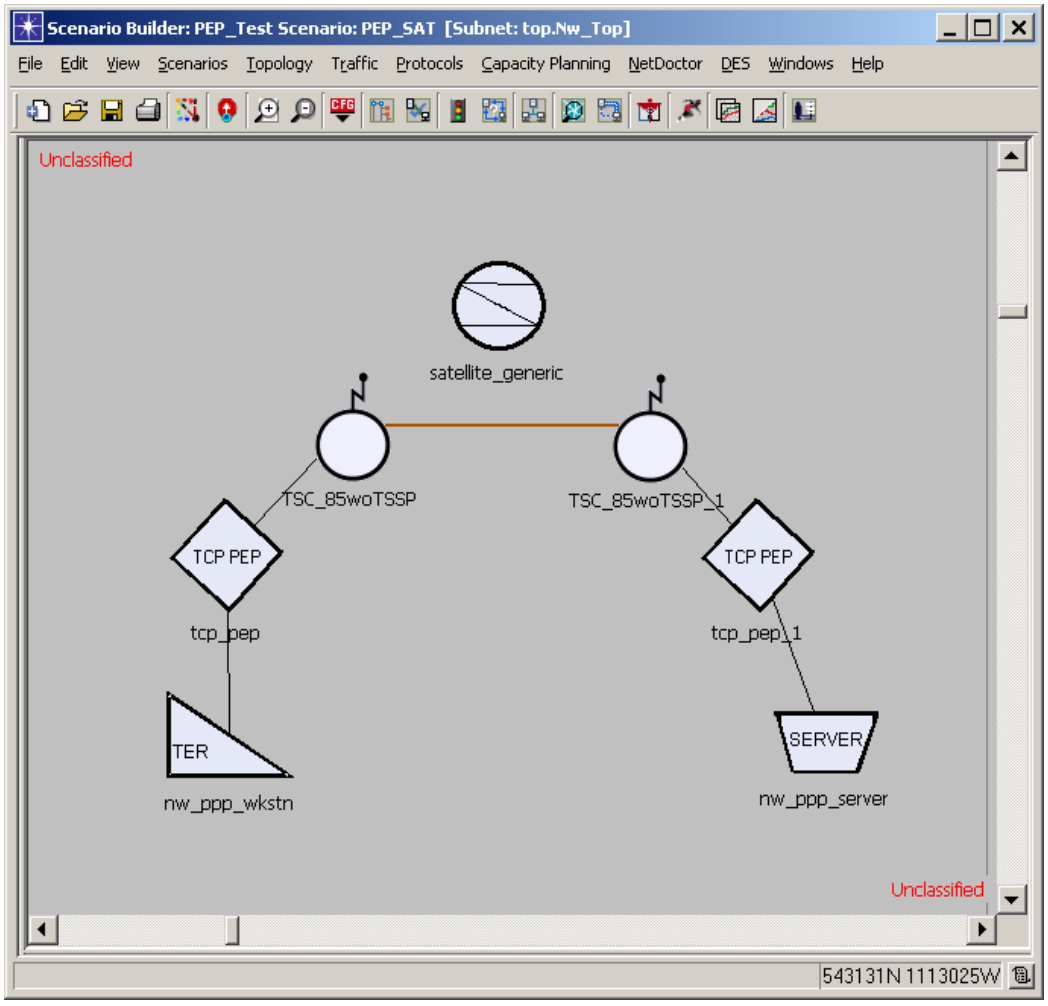

Figure 3: Typical PEP Usage on Both Sides of a Satellite Link

3.1 Capacity Planning Reports

The Capacity Planner is a tool which allows you to quickly study a network by analytically calculating the performance. This provides you with a large amount of studies such as failure, configuration, and capacity studies.

To run Capacity Planner on a network, click the **Capacity Planning > Evaluate…** menu after opening a particular NETWARS scenario. Refer to the NETWARS documentation for more information regarding how to use the Capacity Planner tool.

Once the Capacity Planner tool is run, a web report will be shown displaying the results from the analysis. It is important to note that Capacity Planner will work with the TCP PEP devices. However, as Capacity Planner does not use the actual Layer-3 or Layer-4 protocols during analysis, the reports will not use the different TCP setups which are common to PEP devices when computing statistics. This point illustrates the differences between Capacity Planner and Discrete Event Simulation. Capacity Planner shows the bottlenecks and utilization studies associated with the network. For the most part, utilization is not effected by the different setup times displayed by TCP (as the same amount of data is sent throughout the network). Therefore, the TCP PEP device will act similar to an IP router inside the Capacity Planner. Any lower-level, protocol layer studies are reserved for Discrete Event Simulation where setup times, end-to-end delay, etc. are able to be modeled more appropriately.

3.2 Discrete Event Simulation Statistics

The following table lists PEP-specific statistics. They can be gathered as both global and local statistics.

Table 4: PEP Statistics

Statistic	Capture Mode	Description
Aborted Connection Count	bucket/default total/sum	Number of aborted connections.
Active Connection Count	bucket/default total/sum/no reset	Number of active PEP connections.
Attempted Connection Count	bucket/default total/sum	Number of attempted connections.
Failed Connection Count	bucket/default total/sum	Number of failed connections.
Load (bytes/sec)	bucket/default total/sum_time	Throughput in bytes per second.
Load (packets/sec)	bucket/default total/sum_time	Throughput in packets per second.
Successful Connection Count	bucket/default total/sum	Number of successful connections.

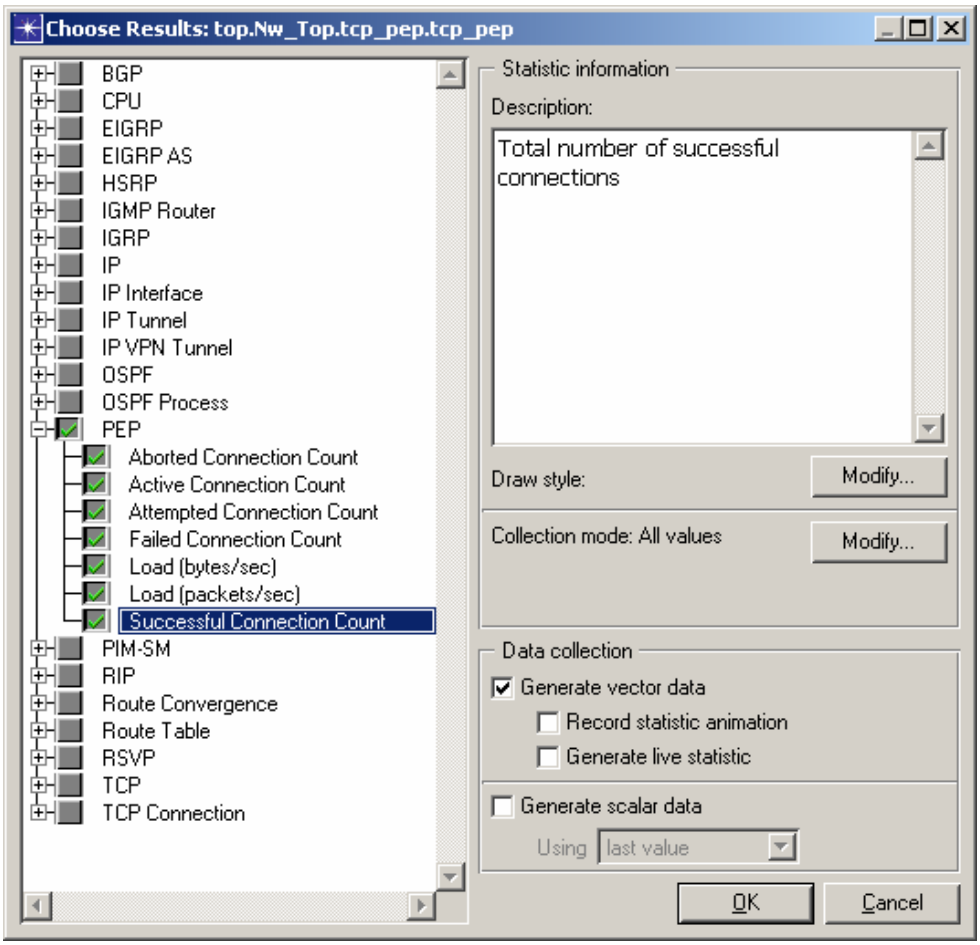

Figure 4: PEP Statistics

TCP statistics will be exposed as well. They have already been implemented in the TCP processor. They are listed in the following table.

Table 5: TCP Statistics

Group Name	Statistic Name
TCP	Active Connection Count
TCP	Blocked Connection Count
TCP	Connection Aborts
TCP	Connection Aborts (RST Rcvd)
TCP	Connection Aborts (RST Sent)
TCP	Connection Port
TCP	Delay (sec)
TCP	Load (bytes)
TCP	Load (bytes/sec)
TCP	Load (packets)
TCP	Load (packets/sec)
TCP	Retransmission Count
TCP	Segment Delay (sec)
TCP	Traffic Received (bytes)
TCP	Traffic Received (bytes/sec)
TCP	Traffic Received (packets)
TCP	Traffic Received (packets/sec)
TCP Connection	Congestion Window Size (bytes)
TCP Connection	Delay (sec)
TCP Connection	Flight Size (bytes)
TCP Connection	Load (bytes)
TCP Connection	Load (bytes/sec)
TCP Connection	Load (packets)
TCP Connection	Load (packets/sec)
TCP Connection	Received Segment Ack Number
TCP Connection	Received Segment Sequence Number
TCP Connection	Remote Receive Window Size (bytes)
TCP Connection	Retransmission Count
TCP Connection	Retransmission Timeout (seconds)
TCP Connection	Segment Delay (sec)
TCP Connection	Segment Round Trip Time (sec)
TCP Connection	Segment Round Trip Time Deviation
TCP Connection	Selectively ACKed Data (bytes)
TCP Connection	Send Delay (CWND) (sec)
TCP Connection	Send Delay (Nagle's) (sec)
TCP Connection	Send Delay (RCV-WND) (sec)
TCP Connection	Sent Segment Ack Number
TCP Connection	Sent Segment Sequence Number
TCP Connection	Traffic Received (bytes)
TCP Connection	Traffic Received (bytes/sec)
TCP Connection	Traffic Received (packets)
TCP Connection	Traffic Received (packets/sec)

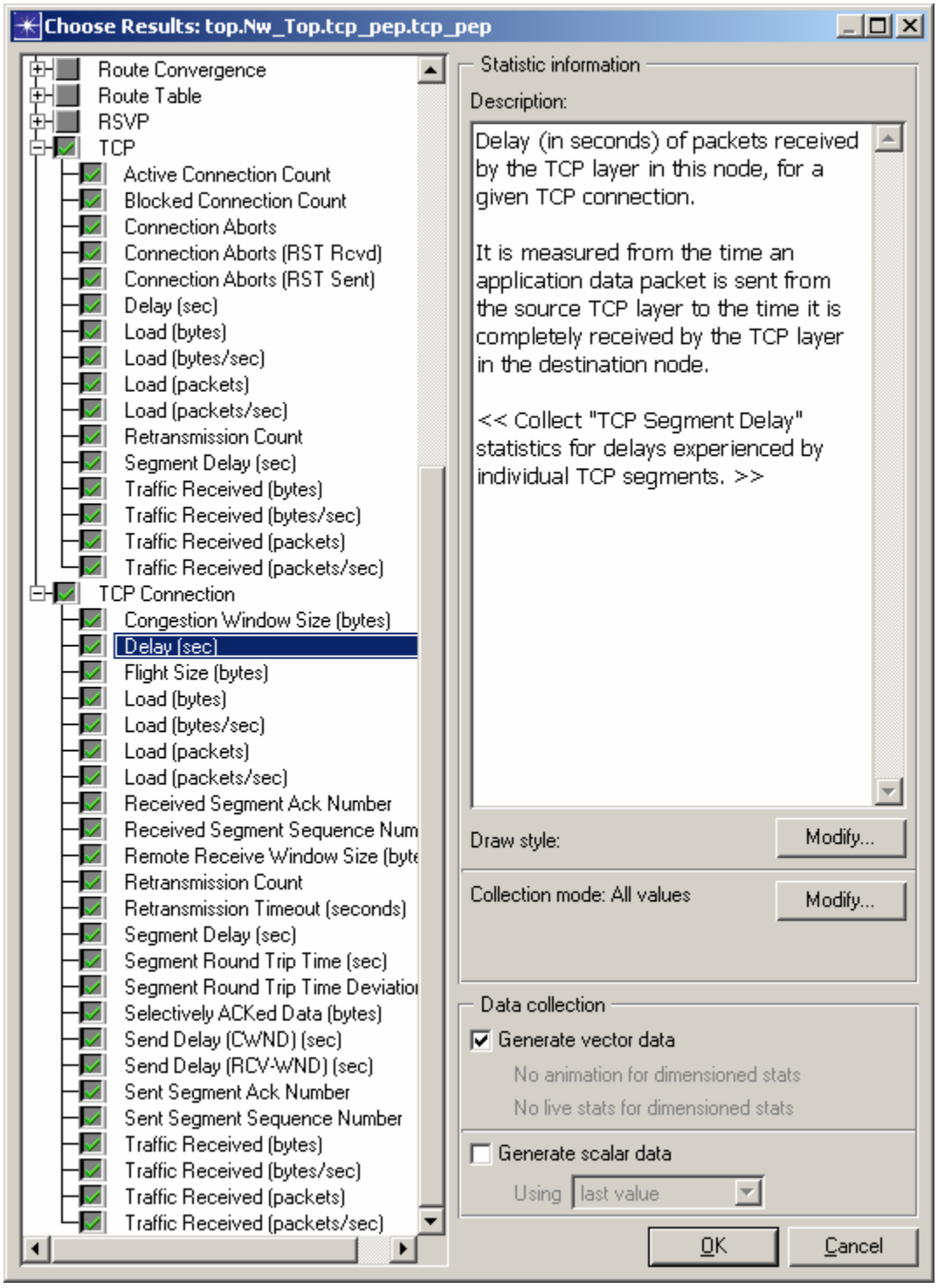

Figure 5: TCP Statistics

3.3 Logical Views

Logical Views provide the ability to filter the network so that the user can look at particular layers or technologies more closely. In the case of TCP PEP devices, the user can look strictly at these devices in the network by using the "IP" layer. This layer can be located under the "Network Layer Filters" section of the logical links dialog box. Note that the logical links functionality can be found under the **View > Show Logical Views…** menu (dialog box shown below).

Once a logical view of the TCP PEP devices is created, the user will be able to see a graphical representation of the IP connectivity of the devices inside the scenario. This is helpful in making sure that the TCP PEP devices are configured properly and hold connectivity to the end devices which use the PEP functionality. This is especially important as TCP typically runs over the IP layer.

Note that it is recommended that this functionality be used before investing time in a Discrete Event Simulation. Logical views can quickly and graphically show the user network problems while the Discrete Event Simulation may take a much longer time to finish.

3.4 Supported Equipment Strings

In general, PEP nodes are special routers that are placed as gateways that connect a client to the rest of the network. Figure 6 and Figure 7 are examples of typical configurations. In general, PEP can connect to both Ethernet and Serial devices on both sides, and the choice of the link is a function of the interface type of the connecting devices.

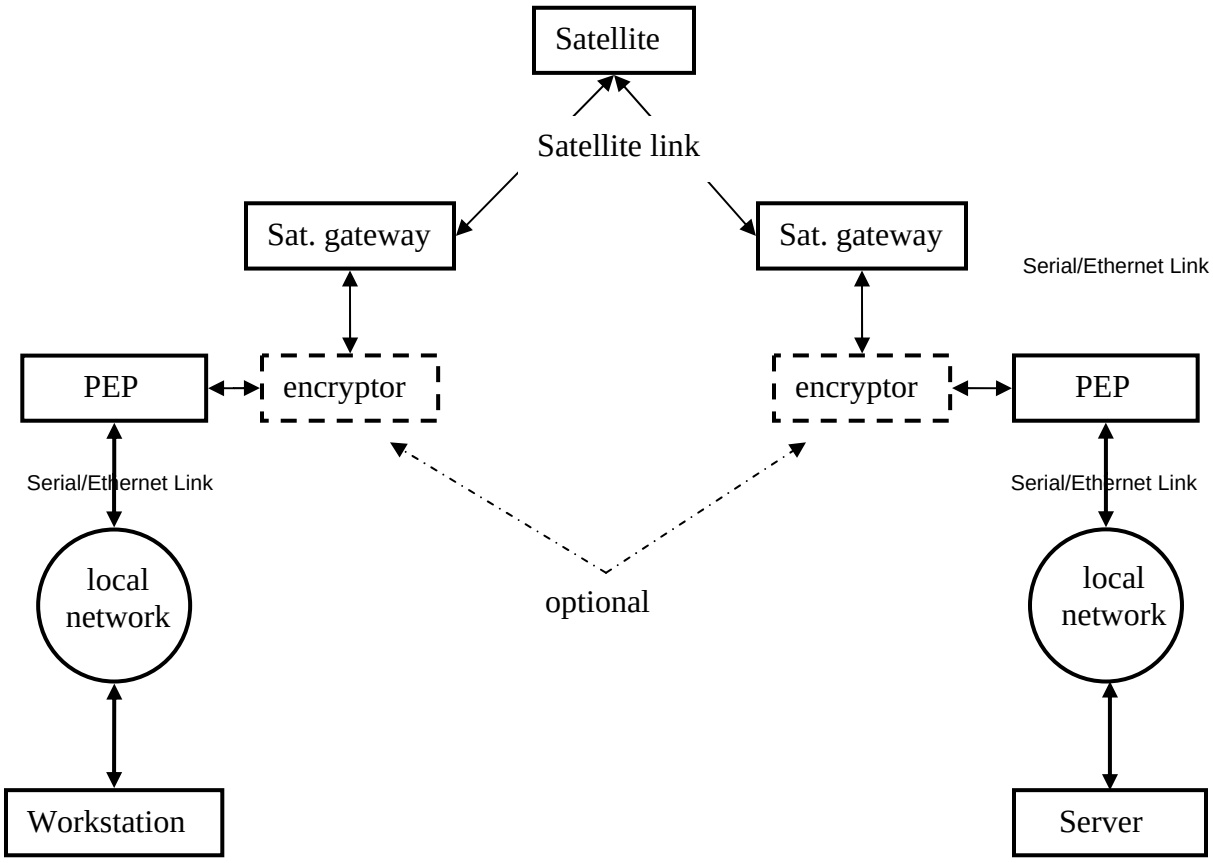

Figure 6: Using Two PEP Devices to Maximize the Utilization of Satellite Link

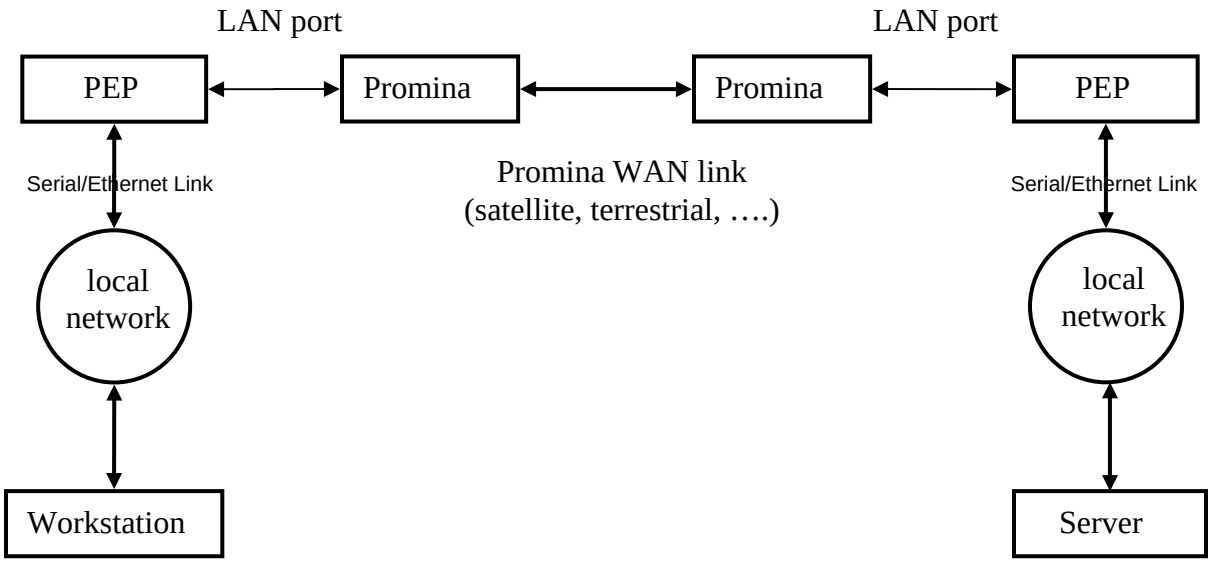

Figure 7: PEP Nodes on Two Sides of a Promina WAN Link

3.5 Traffic Usage

All data passing through the TCP PEP are treated as generic. Therefore, all traffic types which work on the end devices (such as IERs, Applications, Flows, etc.) will successfully route through the TCP PEP device.

3.6 Usage in Modeler

The following modified standard OPNET files must be included in the model directories to make PEP functionality available in Modeler.

Table 6: Modified Standard OPNET Files

File Name	Contents
ip_dispatch.pr.m	Modified standard Modeler process.
ip_rte_support.ex.c	Modified standard Modeler external C file.
ip_rte_support.h	Modified standard Modeler external C header file.
pep_app.h	Defines interface between TCP PEP manager and connection process.
pep_app.pr.m	PEP process manager.
pep_app_support.ex.c	PEP support functions.
pep_app_support.h	Header file for PEP support functions.
pep_app_tcp.pr.m	PEP connection process.
tcp_pep.nd.d	TCP PEP derived model.
tcp_pep_adv.nd.m	TCP PEP node model.

4 Node Models

4.1 Device Creator

The PEP devices are not currently supported by the Device Creator utility.

4.2 Link Models

The following Ethernet links are supported:

- 10BaseT
- 100BaseT

The following Serial links are supported:

- PPP_DS0
- PPP_DS1
- PPP_DS3
- PPP_SONET_OC1
- PPP_SONET_OC3
- PPP_SONET_OC12
- PPP_E1
- PPP_E3
- T1
- T3
- wire_ptp

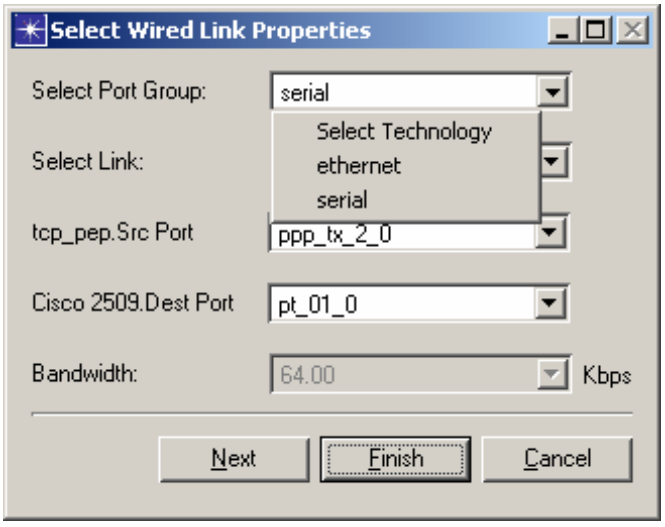

Figure 8: Select Wired Link Properties Dialog Box

4.3 Node Model Architecture

4.3.1 Node Architecture

The PEP node model is depicted in Figure 9. This node model is typical of IP routers, but has a modified IP process model and one additional process model, PEP.

The IP process model is modified to pass incoming IP packets containing TCP packets to an upper layer instead of simply routing them to the other port. Certain packets will be sent directly to the PEP processor, while most will be sent to ip_encap. It will also perform network address spoofing for packets going to and from the ip_encap processor.

The PEP model interfaces directly with the IP module, to receive certain TCP packets intercepted by IP, and with the TCP module (via the Connection API) to open new TCP connections and to send and receive TCP segments.

The PEP node self-description characteristic called "machine type" is set to the value of the router.

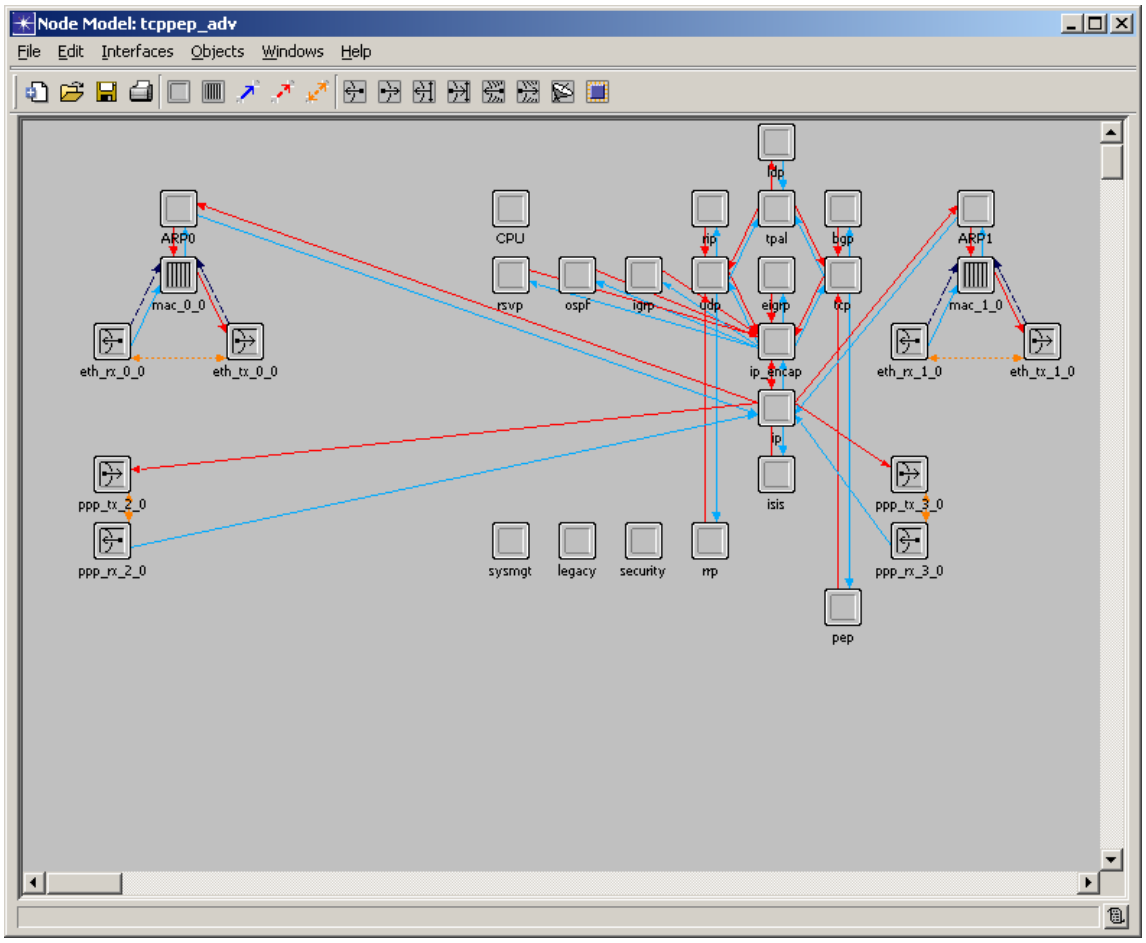

Figure 9: PEP Node Model

4.3.2 Process Models

The following process models provide the PEP functionality:

- pep_app.pr.m
- pep_app_tcp.pr.m

The main functions performed by these models are discussed in the following sections.

4.3.2.1 Initialization

As part of the initialization process, the PEP device:

- Reads in the attributes indicating the mode of operation and the supported transport protocol that must be used to set up the connection across the physical link whose performance needs to be enhanced.

- Initializes a connection table to map an incoming TCP connection to a connection with peer PEP.

- Initializes the statistic handles.

4.3.2.2 Packet Processing

In the IP layer when a packet is received, it is either received from the higher layer or the lower layer. For all the packets received from the higher layer no additional processing is required, but for the packets received from the lower layer if it is TCP-based traffic, then it will either be Outbound Processed or Inbound Processed depending on whether it is received on the network_intf or peer_pep_intf respectively. All the intercepted inbound packets must be either sent directly to the PEP module or indirectly to the TCP module via ip_encap.

The PEP module is primarily responsible for creating new connections using the Connection API, while the IP module will implement most of the *spoofing* mechanisms.[2] For example, except for the TCP SYN packet, all IP packets containing TCP from the network_intf would have the destination host address rewritten to be the IP address of the PEP node, that is, the loopback address. Similarly, IP packets in the other direction would have the source address rewritten to be the IP address being spoofed. This requires a connection table, local to the PEP node, which is updated and maintained by the PEP module and is also accessible by the IP module.

The following discussion details TCP connection establishment from Client to Server (in the figure below) and data transfer from Client to Server. The same logic can be expanded for support in other direction as well.

[2] Actually, in the external C file, ip_rte_support.ex.c.

Below, we focus our discussion on a case where the transport protocol between the two PEPs is also TCP (rather than SCPS-TP, for example). The following discussion refers to node models in the following diagram, (Figure 10) Client, PEP A, PEP B and Server.

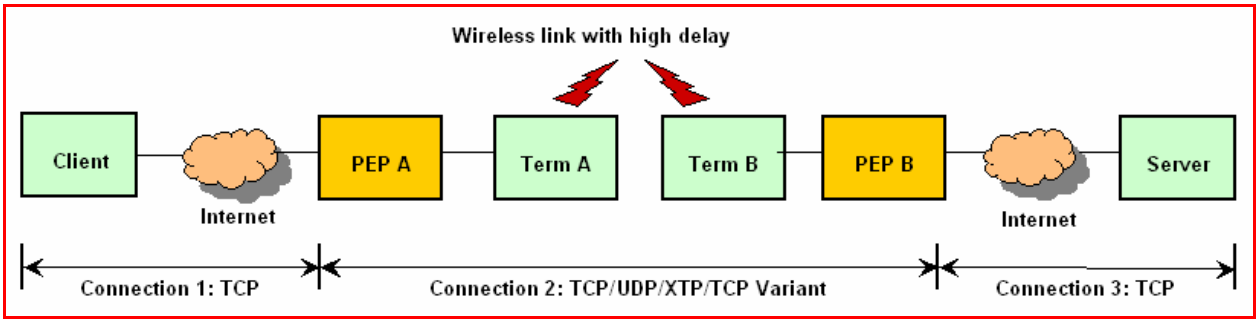

Figure 10: Connection Establishment

4.3.2.3 Connection Establishment

- All incoming IP packets, in particular those containing TCP messages, will be sent to the IP module, regardless of the port at which a packet is received. The IP module will determine the source of the packet (from a host or another PEP) and spoof the destination address, if necessary.[3]

- TCP SYN
  - TCP SYN is used to initiate TCP call establishment and will be handled separately from other packets. Upon receiving a TCP SYN message (from Client) the IP module will send the packet directly to the PEP module.
  - The PEP module will initiate a transport connection with peer PEP, PEP B, using the Server IP address (which is the destination IP address from the TCP SYN message) as the destination address and its own IP address as the source address.
    - The Connection API will be used to open a transport connection.
    - The Connection API will supply a source port number to be used for this connection.
    - A new connection map entry to the connection table for the incoming call will be added. The information about the call to PEP B will be written to this entry. The state of this connection must be set to *init*.
    - The original SYN packet (still within its IP envelope) will be stored with the connection map.
  - The IP module will spoof the source address (including the port number) of all TCP packets being sent to PEP B to be that of the Client.
  - A similar sequence will take place in PEP B. The IP module of PEP B will inform PEP module about the reception of a TCP SYN message, which just happens to be from another PEP.

[3] There is a pathological exception. It is possible to misconfigure a device in the network so that TCP traffic received in one port would be routed back through the same port. Such traffic should simply be handled in the same way as a router would handle it.

        - The PEP module will initiate another connection to the final destination, with the IP module spoofing the address of the originator.
        - Connection map will be updated here as well. A copy of the SYN message from the PEP A will be stored with the connection and the state will be set to *init*.
    - Eventually the last connection is established between PEP B and the Server.

- TCP SYN/ACK
    - When a SYN/ACK from the server is received by a PEP, the IP module looks up the connection map entry and substitutes the node's loopback IP address and the C2 port number as the destination address, and the message is sent all the way up to the TCP module:
        - This causes a TCP ESTAB indication to be generated and sent to the PEP module.
        - The TCP module also generates an ACK message to be sent in order to complete the 3-way handshake.
    - When the PEP module finds that the connection has been established (by receipt of the ESTAB), the following will be done:
        - In the PEP connection-mapping table, the state of this connection will be set to *half_open*.
        - The PEP module will cause connection C1 to be opened by performing the next step.
    - To complete the opening of connection C1, the PEP will do the following:
        - The PEP module will use the Connection API to initiate a passive open.
        - The SYN message that was stored when the connection was initialized will be sent to the ip_encap module (with the destination address spoofed) so that appropriate SYN/ACK is automatically generated by the TCP and the connection C1 established.
        - When the ACK for connection C1 is received, an ESTAB indication will be sent to the PEP, which will set the state will be set to *estab*.
        - Keep in mind that the IP module will handle all the spoofing of IP addresses.
    - This same process happens first in PEP B and then in PEP A.

- Duplicate TCP SYN packets
    - If there is already an entry corresponding to the SYN packet in the connection table, and the state is *init*, then just drop the packet. If the state is something else, then just spoof the destination address and send it up to TCP.

4.3.2.4 Data Transfer

- If the TCP segment contains data, then the following sequence will take place. This is the same for both connections.
    - The IP module will send this message to the TCP module (or other transport protocol module), spoofing the destination address as that of the PEP node and TCP will then acknowledge it in a standard fashion. As discussed above, the IP

module will intercept the acknowledgement and spoof the source address of this ACK as that of the ultimate source or destination address, either that of the Client or the Server.

- The TCP module will send a SEG_FWD indication to the PEP module, so that the data packet will be received by the PEP module.
- The PEP module will look up the paired connection in the connection table and use the Connection API to send the data packet. The state of the connection should be *estab*, the data is sent on that connection, or else the data segment will be dropped, logging details on the packet being dropped and the location.

- Note that data can still be transferred when a TCP connection is in the half closed state. This is reflected in Figure 12.

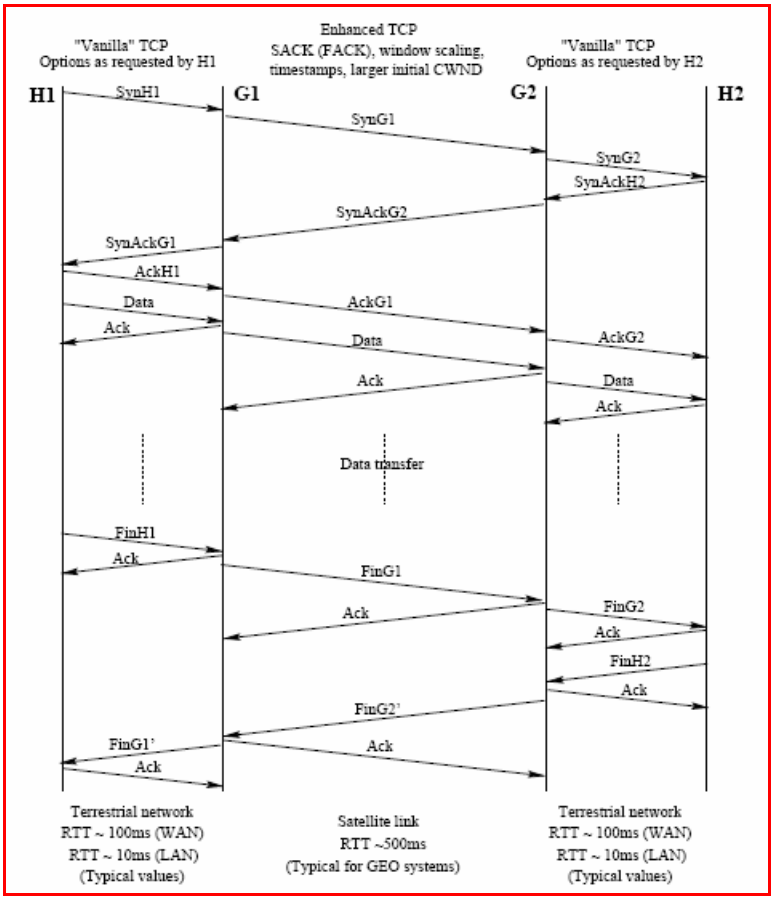

Figure 11: Timing Diagram for Simple Unidirectional Transfer from H1 to H2, from Bharadwaj.

4.3.2.5 Connection Close

- Typically, the Client initiates the closure of a connection by sending a TCP FIN message. (The analysis for a server-initiated closure is very similar.) From the point of view of the Client, data may continue to be received from the Server until a corresponding FIN is received.

- The following procedure is intended to preserve the end-to-end control as shown in Figure 11

  - A FIN from the Client received by PEP A will cause the IP module to spoof the destination and send it upward to the TCP module, via *ip_encap*.
  - The TCP module will acknowledge the data, if any, and send a SEG_FWD indication to the PEP module, which will forward the data as usual.
  - Then the TCP module will send a FIN_RECEIVED indication to the PEP module, which will initiate the closing of connection C2 and enter the *connection_C2_closing* state.
  - The process is repeated in PEP B.
  - When the Server receives the FIN, the connection between PEP B and the Server will be closed in the normal way.
  - Eventually, the connection from PEP A to PEP B will be closed, and the TCP module will send a CLOSED indication to the PEP module.
  - At this point, connection C2 is completely closed, so the PEP module will initiate the closing of connection C1 and set the state to *C2_closed_C1_closing*.
  - The TCP module will close the connection, sending a FIN packet to the Client and a CLOSED indication to the PEP module.
  - When the PEP module receives the CLOSED indication for C1, both connections are completely close, so it will delete the entry in the connection table.

- Nearly identical logic will be used to handle a connection close from the other direction.

- Similar logic will be used to handle an ABORTED indication from TCP.

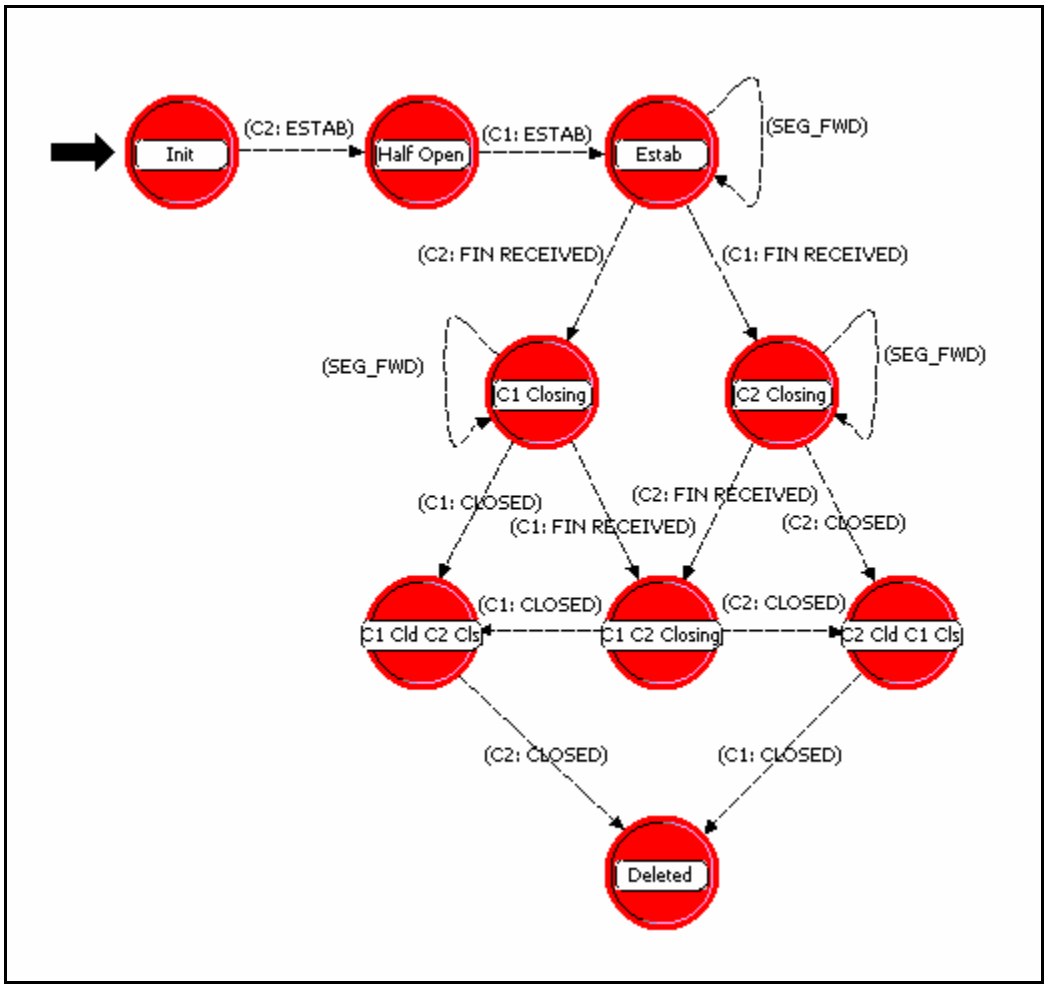

Figure 12: States for a Split Connection (Simplified, Normal Operation)

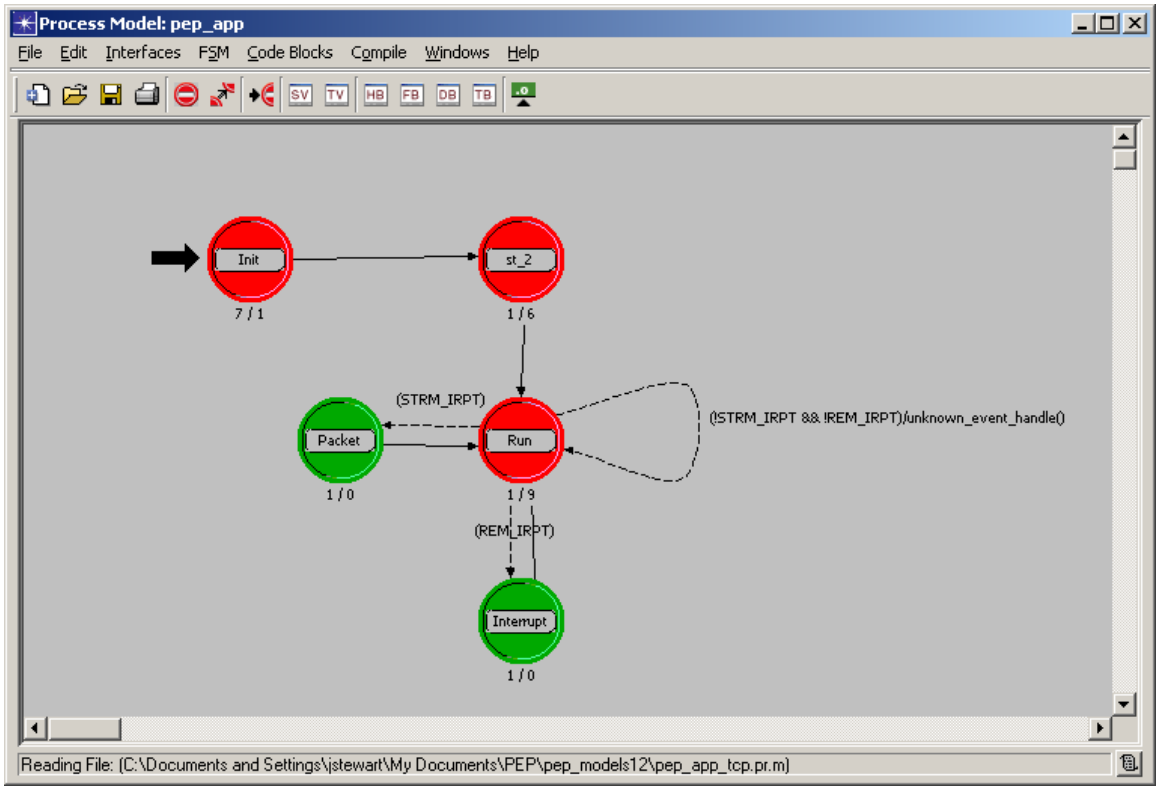

Figure 13: States for the PEP Application Process Manager

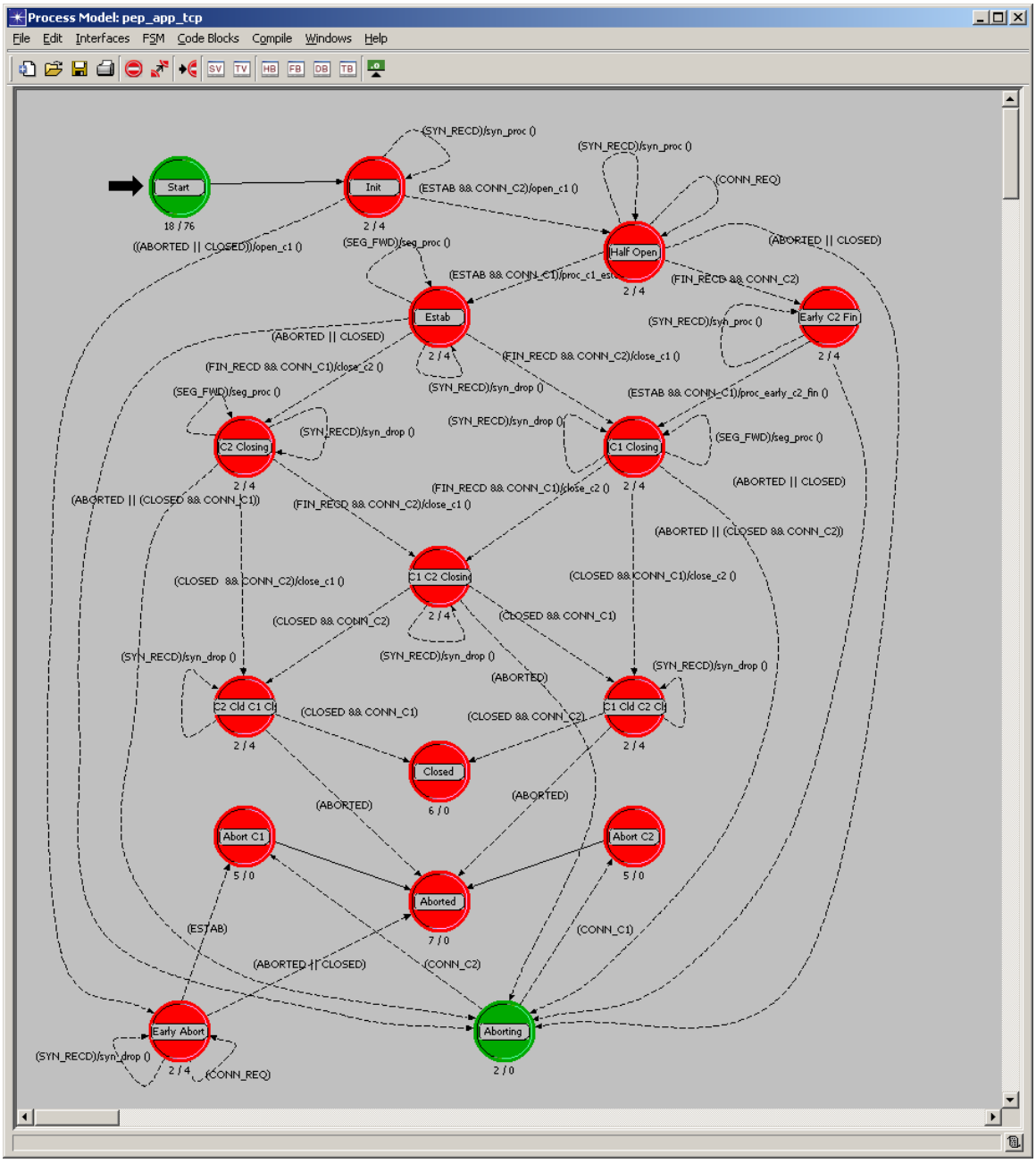

Figure 14: Full State Machine for TCP PEP Connection, as Implemented

4.3.3 Packet Formats

There are no additional packet formats required, beyond the standard formats.

4.3.4 Interfaces

The Self Description information for the PEP node model is shown in Figure 15 to Figure 18. As shown in the figures, the PEP Self Description is essentially the same as the Self Description of a generic IP router with two Ethernet and two Serial interfaces.

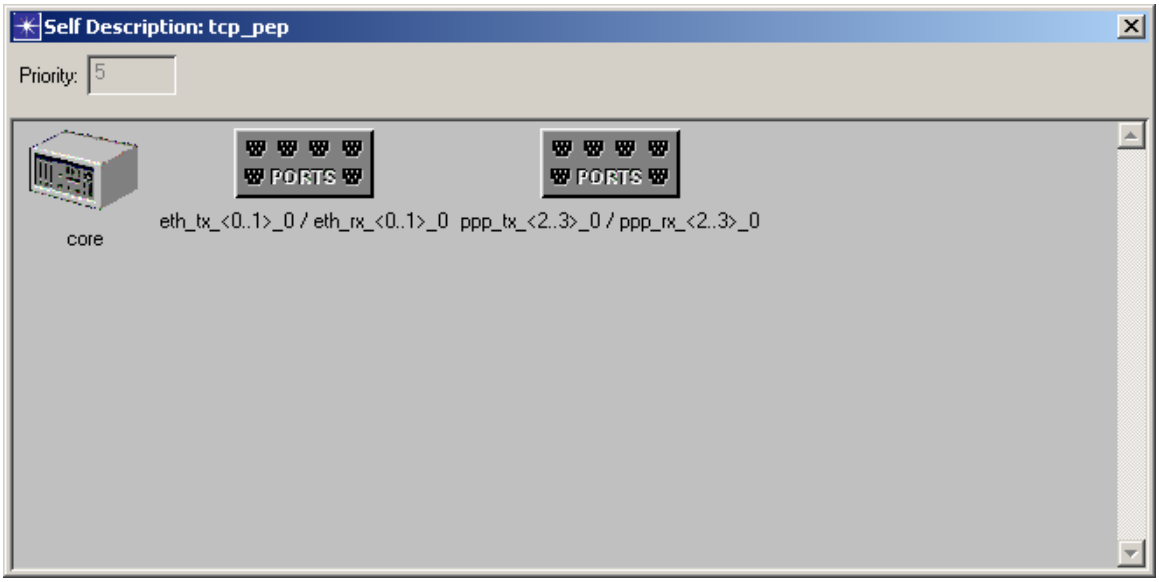

Figure 15: Self Description: tcp_pep

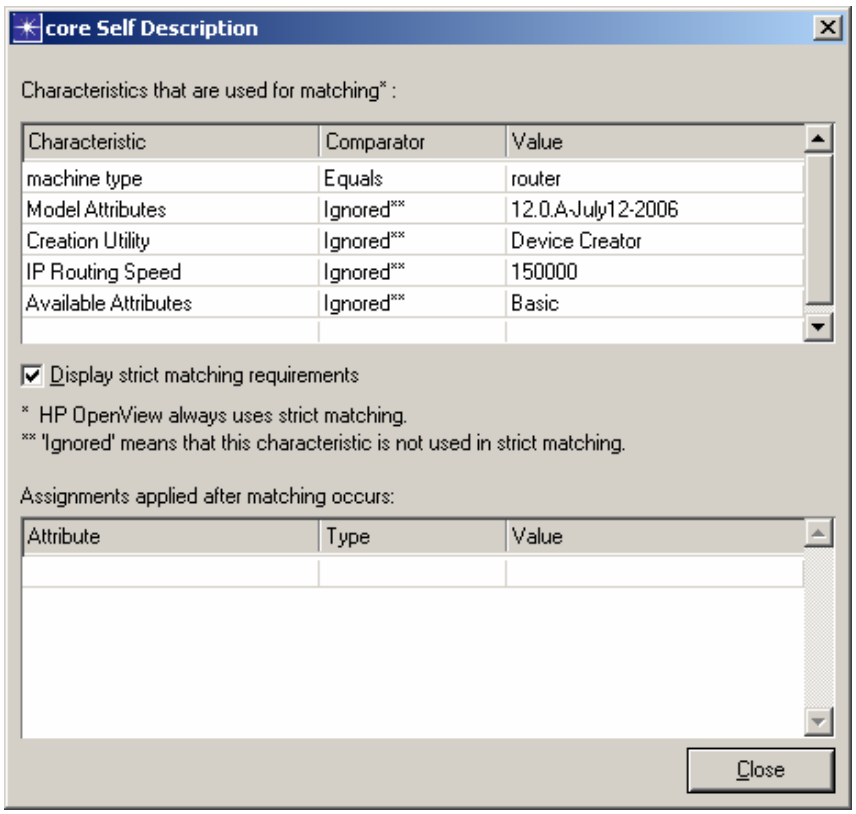

Figure 16: Core Self Description Dialog Box

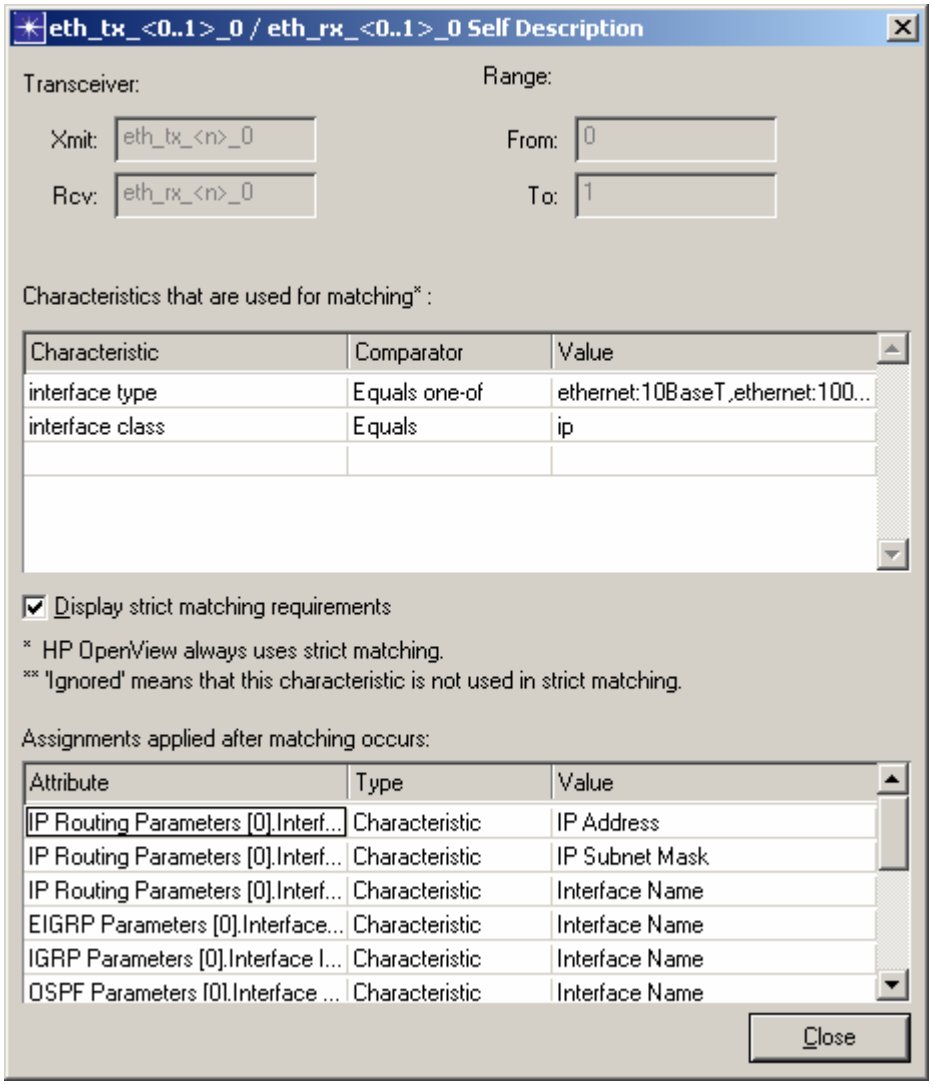

Figure 17: eth_tx… / eth_rx… Self Description Dialog Box

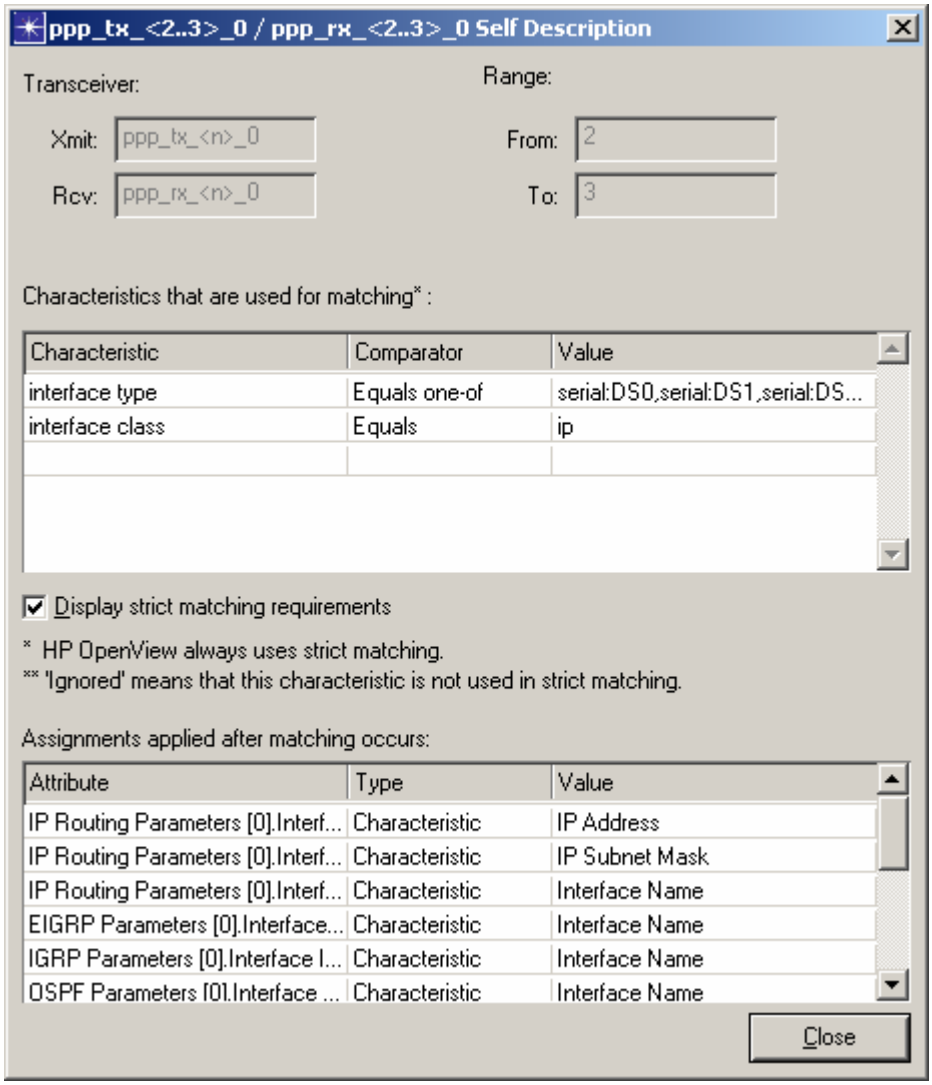

Figure 18: ppp_tx… / ppp_rx… Self Description Dialog Box