1-1-2002

Mechanisms and protocols for intermediate processing at routers in the Internet

Sonal Pandey
Iowa State University

Mechanisms and protocols for intermediate processing at routers in the Internet

by

Sonal Pandey

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Dr. Arun Somani, Co-major Professor
Dr. Akhilesh Tyagi, Co-major Professor
Dr. David Fernandez-Baca

Iowa State University

Ames, Iowa

2002

Graduate College
Iowa State University

This is to certify that the master's thesis of

Sonal Pandey

has met the thesis requirements of Iowa State University

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. OVERVIEW

The Internet consists of a number of interconnected networks supporting communication among host computers using the Internet protocols. These protocols include the Internet Protocol (IP) [2], the Internet Control Message Protocol (ICMP), the Transmission Control Protocol (TCP) [3], and applications using them. IP is a datagram, or connectionless, internetwork service and includes provision for addressing, type-of-service specification, fragmentation and reassembly of packets, and security information. In the Internet, reliable data delivery is provided in the Internet protocol suite by transport-level protocols such as the Transmission Control Protocol (TCP), which provides end-to-end retransmission, re-sequencing and connection control. Transport-level connectionless service is provided by the User Datagram Protocol (UDP). ICMP provides error reporting, flow control and first-hop gateway redirection.

1.1 Introduction

One of the most important constituents of the Internet is an IP router or a gateway [4]. A router scans the destination address of an IP datagram and then forwards the datagram to another router or to a directly connected host.

In current practice, routers are normally realized by packet-switching software executing on them. Special-purpose hardware may also be used. A router is connected to two or more networks, appearing to each of these networks as a connected host. Thus, it has a physical interface and an IP address on each of the connected networks. Forwarding an IP datagram generally requires the router to choose the address of the next-hop router or (for the final hop) the destination host. This choice, called "routing", depends upon a routing data-base within the router. Thus the computation done on a datagram at a router is limited to scanning the IP header and updating header information (e.g. Time To Live).

Lightly to moderately loaded routers use only a part of the CPU processing power. [8] shows the yearly CPU usage average for routers monitored for their CPU utilization with

Multi Router Traffic Grapher [8] which is a tool to monitor traffic load on network links. The graphs indicate that CPU utilization may not be very high in many routers. Such routers could be used for performing more rigorous computation on the datagrams. It is also likely that certain routers remain heavily used for a certain period of time, but only incur low traffic at other points in time. It is also this kind of routers that could support computation on datagrams.

1.2 Intermediate processing

If the processing capability of a router is further enhanced to support computation on parts of data contained in a datagram, some of the end host computation can be delegated to the routers that the data passes through. The instructions about how to do the processing itself could be provided by the end hosts.

This would be helpful for client-server kind of applications where the server delivers to the client some data based on the client's request. The server may send out an approximate set of data that could be pruned accordingly using the code supplied by the server. This pruning could be done at the routers. This helps alleviate computation load on the server. On the other hand, code could also be placed on the routers so that they assist the client receiving the data in performing some part of the computation on the data while it is en route to the client. This would help alleviate computation load on the client. Another application of intermediate processing could be in providing Quality of Service (QoS) information about connection data to the routers so that the data is treated in the way described by the accompanying code with the data. This approach can also be used in making available services to intermediate nodes that would otherwise be available only at the end hosts. More applications of intermediate processing are described in [13].

The ideas discussed in the previous sections fall into the paradigm of active networks [1]. Active Networks represent a new approach to network architecture that incorporates intermediate computation. These networks are active in a way that routers and switches within the network can perform computations on the application data flowing through them.

Also, users can program the network, by supplying their own programs to perform these computations.

1.3 Active networks research

Active networking, where network nodes perform customized processing of packets, is a rapidly expanding field of research. The research work in laboratories around the world is being pursued with a positive outlook that active networking technology will mature to a point where it can be commercially deployed on a larger scale [10]. Active networks aim to constructively mutate the strong end-to-end semantics of the current transport mechanisms to realize the benefits of customized intermediate processing in a network. [11].

One of the earliest projects in this area of research started at Massachusetts Institute of Technology in 1996 [6]. The authors proposed an active network architecture in which passive packets are replaced with active capsules - encapsulated program fragments that are executed at each switch they traverse. This approach allows application-specific processing to be injected into the network. The feasibility of computation and storage within the network provides a base that can be enhanced to build global applications, including those that invoke customized multicast and merge processing. The work described an extension to the IP options mechanism that supports the embedding of program fragments in datagrams and the evaluation of these fragments as they traverse the Internet.

A sequel to this architecture was the ANTS project [9], a toolkit for building and dynamically deploying network protocols. The approach is based on mobile code [14], demand loading, and caching techniques. The architecture of the system allows new protocols to be dynamically deployed at both routers and end systems, without the need for coordination and without unwanted interaction between co-existing protocols.

NetScript, [7] developed at Columbia University is a programming language and environment for building networked systems. Its programs are organized as mobile agents that are dispatched to remote systems and executed under local or remote control. The goal of NetScript is to simplify the development of networked systems and to enable their remote

programming. The NetScript project aims at networks that allow for flexible programmability and dynamic deployment of software at all nodes. Application developers could develop code for both end-nodes and intermediate nodes and incorporate innovative features without having to wait for slow standardization processes. The NetScript language includes constructs and abstractions to simplify the design of traffic-handling software.

The Composable Active Network Elements (CANEs)[12] project at Georgia Institute of Technology is a DARPA/ITO-funded effort investigating architectures and applications for active networking. The CANEs project seeks an approach to active networks that supports high performance while also permitting dynamic modification of network behavior to support specific applications or provide new services. The project comprises of an overall architecture for the active network; a specific user-network interface (the CANEs environment); and experimental studies of the effectiveness of various active networks in support of specific applications and services.

As part of the DARPA Active Nets program, BBN Technologies is developing a capability for packets to carry programs which are executed at each node the packet visits in the network. The programs implement additional diagnostic functionality in the network. To do this, BBN is developing a special language for programming Smart Packets [15] called Sprocket. A complete Sprocket program is to be encoded in either a single IPv4 or IPv6 datagram. The goal is to fit a program in an Ethernet-sized packet, which requires a very compact representation. Furthermore, the program would be authenticated before interpretation and run-time limited during execution.

In the series of projects on developing programming languages for active networks is the PLAN project at University of Pennsylvania [16]. PLAN is a resource-bounded functional programming language that uses a form of remote procedure call to realize active network programming.

This thesis defines a reliable transport layer protocol, Intermediate Processing Protocol (IPP), that comprises a set of rules and guidelines for processing in the Internet. The guiding factor in the design of IPP has been to adhere to the current well established IP routed network

architecture while enhancing its ability to support active networking without replacing or making drastic changes to the Internet protocols currently used. This could lead to faster and easier deployment of intermediate processing since we are not replacing the current IP routed set up but only enhancing it. The design is based upon the realization that it may not be practical to expect all the nodes of the Internet to become "active", hence the protocol definition should include provision for seamless integration with the "non-active" IP nodes. At the same time, there should be no restrictions on the "activeness" of the networks due to Ethernet frame size and/or IP fragmentation.

1.4 Thesis outline

The outline of this thesis is as follows. Following the introductory material and literature survey in this chapter, active networks terminology and preliminary design concepts are presented in Chapter 2. Chapter 3 introduces the new TCP and IP options that have to be an integral part of IPP. The chapter describes the design of the options in detail. The design of IPP layer modules is discussed in detail in Chapter 4 which also introduces further concepts and data structures suggested for implementation of IPP. Chapters 5 and 6 describe the working of IPP in terms of connection set up, code transfer and processing on data. Chapter 7 deals with the control mechanisms and security issues that arise in case of deployment of IPP in active networking. Chapter 8 describes the IPP header format and summarizes the types of messages required for control and data transfer in IPP. Chapter 9 is a discussion about an experimental implementation of IPP on Redhat Linux kernel 2.2.17. The chapter also makes recommendations on the future work that could be pursued further in the field of active networking protocols and specifically the enhancement of IPP.

CHAPTER 2. NETWORK MODEL AND DESIGN PRELIMINARIES

2.1 Terminology

- *Active router:* An IP router that supports IPP. In this document, the word router stands for an active router unless stated otherwise.
- *Active host:* A host that supports IPP. In this document, the word host stands for an active host unless stated otherwise.
- *Sender based system:* A pair of end hosts where the sender of the data is required to do certain computation on the data before sending it to the receiver end host. An example of this is searching a set of records on the basis of certain search parameters before sending the relevant records to a receiver.
- *Receiver based system:* A pair of end hosts where the receiver of the data is required to do certain computation on the data received from the other end host. An example of this is the decoding of MPEG files that a receiver end host receives from a sender end host.
- *Sender-receiver based system:* A system that is both sender based and receiver based.
- *IPP segment:* A TCP segment provided with additional information about intermediate processing.
- *Logical segments:* The set of IPP segments containing the result obtained by performing one round of computation on the data at a router. These segments together depict the result of the computation and delivery of all of these is required for the receiver to comprehend the result. IPP segments that are logical segments carry the same sequence number in their headers.

2.2 Network Model

The model of network assumed for the work stated here is the IP routed packet switched model. The network consists of nodes some of which are configured as routers. Some or all of the nodes support IPP. Figure 2.1 depicts an example network. It is required that both the end hosts that wish to communicate using IPP support IPP, though it is not necessary that all routers support IPP. The end hosts could have knowledge of the location of active routers in the network(s) between them. This is not a necessity, since the protocol allows for active routers to identity themselves on the fly to the end hosts during a connection request.

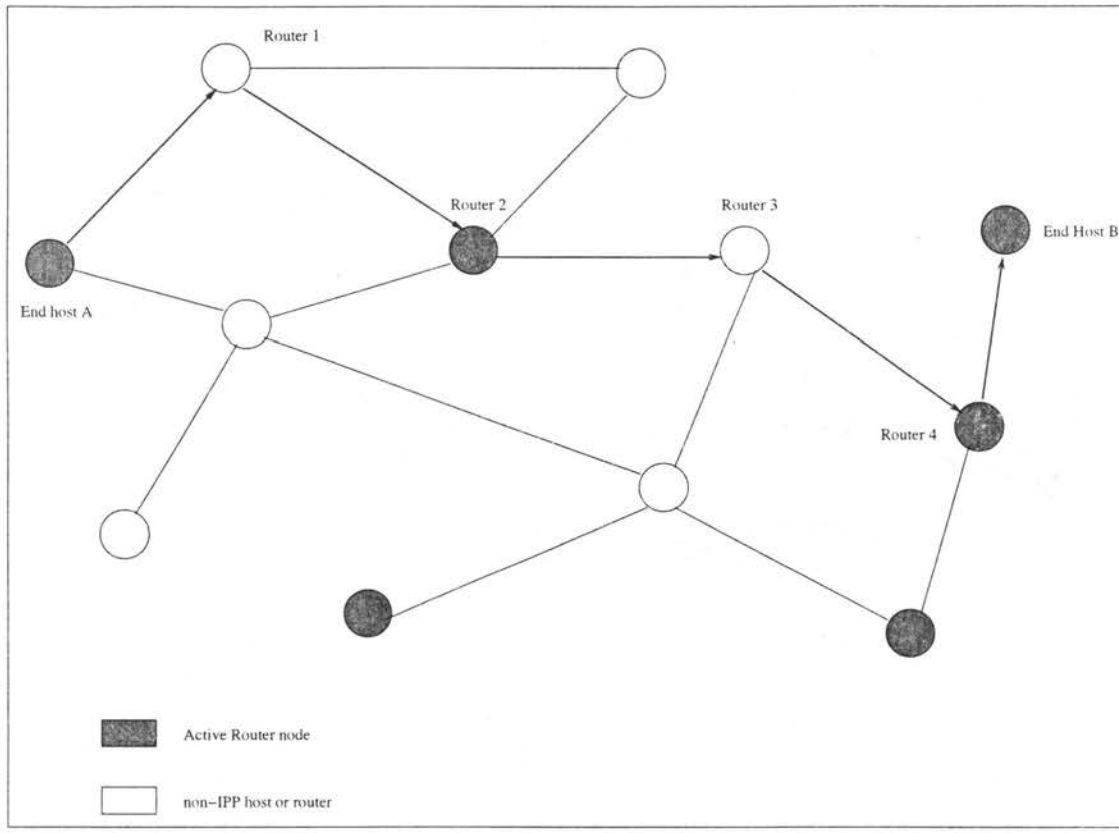

Figure 2.1 Nodes configured as hosts and routers in a network

It is helpful to have the computation to be possible on parts of data so that the computation does not require all the data to be present at a router at the same time. The active routers must be able to defragment the IP fragments that they receive for an application requiring

intermediate processing. The sequence of events describes the mechanism in terms of one end host sending and the other receiving. But the protocol is not limited to simplex data exchange nor does it require an end host to be exclusively a sender or a receiver. That an application is an active application is indicated by the well-known port numbers the host attaches itself to. If both the end hosts participating in a session wish to make use of intermediate processing, they should both attach themselves to ports that indicate so.

The data is stored in a format such that the end hosts can recognize how much of data is required for one round of computation. By one round of computation it is meant that there is a periodicity in the way computation is done: the same computation is performed on parts of data. If there are multiple levels of computation required, the result of the previous level forms the input for the next level.

2.3 Intermediate Processing Protocol: an overview

This work presents an "active" reliable transport layer protocol, Intermediate Processing Protocol (IPP), that would facilitate processing of data on routers. This protocol is "active" in that routers within the network can perform computations on user data flowing through them and also users or network administrators can program the network, by supplying their own programs to perform these computations on the data they are exchanging with other end hosts. The trade-off between routing and processing at a router can be achieved based on the traffic statistics at the router. Routers that are not utilized to their full capacity for routing would be able to provide more processing time than ones that are highly utilized for routing purposes. An important goal of the protocol design is to place minimal additional complexity at a router so that normal routing performance does not suffer (or at least not significantly) because of any processing done at the router. The IPP is based on Transmission Control Protocol (TCP) and relies on the same underlying concepts as used in TCP. Additional information is carried in TCP segments to indicate that the data for the connection requires intermediate processing. Modifications to retransmission policy, window updates policy, congestion and flow control are needed to make TCP better fit the active networking paradigm.

The IPP is intended to provide host-to-host connectivity to machines wishing to communicate. At the same time, the IPP is intended to provide point-to-point connectivity between routers. The IPP is a reliable transport layer protocol allowing the end hosts to view a connection as an end-to-end connection and also letting intermediate nodes (routers) to view the same connection as a point to point (router-to-router) connection. The protocol definition includes provisions for connection set up handshaking, router reservation, intermediate processing of data, acknowledgment of data received, retransmission of lost data, flow control, congestion control, ordered delivery of data and security issues. Since it is not practical to require that all the nodes of the Internet are able to support active networking, the protocol described in this work ensures seamless integration of active nodes with the non-active ones in the Internet.

The location of IPP in the 7-layer model is as shown in figure 2.2. The IPP forms the layer above the IP layer.

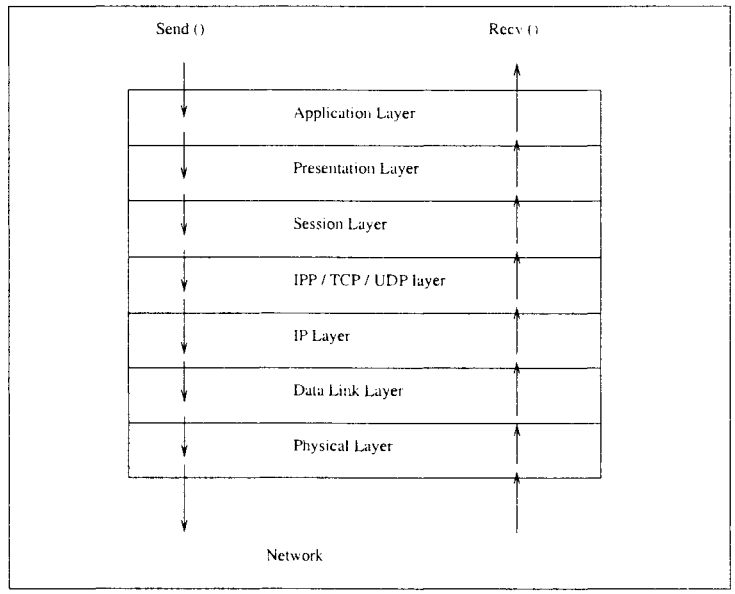

Figure 2.2 Placement of IPP in the protocol stack

The IPP layer does not function like a full-fledged transport layer and that is important to minimize the processing overhead incurred at an active router. The IPP layer at a router or an end host does not maintain state information about a connection. In that way, IPP serves more like an addition to the IP layer than an independent transport layer on its own. The

design goal is to support processing at an intermediate node while limiting the memory and computation overhead to an acceptable maximum.

CHAPTER 3. ADDITIONAL TCP AND IP OPTIONS

3.1 IP options

The IP layer is associated with routing of datagrams flowing through the router. There are modifications needed at the IP layer to support intermediate processing. Instead of directly forwarding the data that needs intermediate processing, the IP layer passes it to the IPP layer with additional information about the type of handling data requires in that case. The following IP options must be implemented at the IP layer of active routers and active hosts:

1. IP Active Record Route (IPARR)

2. IP Active Process Route (IPAPR)

3.1.1 IPARR option

The IPARR option is used to reserve active routers in a session between two end hosts at the time of connection set up. Any active routers along the route that the connection request traverses can indicate their capability and willingness to support intermediate processing for the session. They can do so by placing their IP addresses in the list of router addresses contained in the option data part of the option. The IPARR option is an IP level option and would be carried in the IP header part of the connection request. The IPARR option would be set at the end host requiring intermediate processing from active routers. The format of the IP option is as follows:

1. Type field contains IPARR

2. Length field contains the length of the IP option

3. Option data field contains the sub-fields:

    - Levels of processing required

- List of active routers
- Nature of Request

The type field of the IP option field contains IPARR identifying that it is the IP Active Record Route option. The length field of the IP option specifies the length of the options including the type and length fields.

Levels of processing implies that the data needs to be operated upon by so many different scripts in a certain order. This also implies that the end host requires so many active routers to commit processing. One router may commit to more than one level of processing. It may be acceptable to have fewer routers commit processing than required. This case may be taken care of by the end hosts depending on whether the system is a sender based system or a receiver based system i.e. an end host may decide to do some part of the computation itself.

The list of active routers contains the 32 bit IP addresses of the routers the connection set up request must pass through. Depending on whether a certain router is willing to do processing or not, the routers may place their IP addresses in the list. It will also be depend on how many more levels of processing are required to complete the computation. If there are no more routers needed to do the computation, the subsequent routers must not place their IP addresses into the list irrespective of whether they can commit processing or not. In case the sender host does have the knowledge of the location of active routers in the network, it could send out a list of router IP addresses. The active routers that the connection request passes through may decide to commit processing. If a router cannot commit processing, but its IP address is listed in this list, it must remove its IP address from the list. If a router commits to more than 1 levels of processing, it should place its IP address in the IPARR address list that many times. So, the length of the IP address list is the same as the number of levels of processing required.

Nature of Request indicates whether the option is for requesting reservation or replying to a reservation request.

The list of routers contains only the active routers on the route. The actual route may include non-active routers, but they are not reserved. All that is required is that the same set

of active routers be traversed by the data in the order of router reservations.

3.1.2 IPAPR option

The IPAPR option is used to indicate to the routers and the end hosts that data carried in the datagram needs intermediate processing. When a router receives a datagram with this option set, it defragments the datagram if needed, and then passes it up to the IPP layer. The IPAPR option is an IP level option and is carried in the IP header part of the datagram. The IPAPR option would be set at the end host requiring intermediate processing from active routers. The format of the IP option is as follows:

1. Type field contains IPAPR
2. Length field contains the length of the IP option
3. Option data field contains the list of active routers that need to do the processing.

The IPAPR option if set, would specify to the active router that the data contained in the corresponding datagram needs to be processed. For this purpose, the router should defragment the datagram and pass the segment to the IPP layer. The type field of the IP option field contains IPAPR identifying that it is the IP Active Process Route option. The length field of the IP option specifies the length of the options including the type and length fields. From the order in which its IP address appears in the list of routers, a router can determine what level of processing it has to do on the data.

The options IPARR and IPAPR should be copied during fragmentation. This is because the active router would withhold the fragments and defragment them, if required, for passing the segment up to the IPP layer.

3.2 TCP option

The IPP protocol requires that a new option, TCP Intermediate Processing (TIP) be implemented at the active routers and the end hosts. This option is used to convey information about code and data corresponding to the connection. The format of the option is as follows:

1. Type field contains TIP

2. Length field contains the length of the options including the type and length fields

3. Options data field contains the following sub-fields depending on the scenario it is being used in:

   - *C, Script ID:* Here C stands for code. This sub-type indicates that the information contained in this options data part has to be used in the context of code associated with the connection. The Script ID is a unique identifier associated with the script for the connection. The script ID itself should be an indicator of the level of processing the script can be used to do. Suppose a script is to be used for 3^{rd} level of processing in a connection. Then its script ID should be 3. After connection set up, code should be transferred to the routers and the receiver host. The segments that carry code must have the TIP option set with C flag set and Script ID information placed.

   - *D, Script ID, Sub-sequence number:* Here D stands for code. This sub-type indicates that the information contained in this options data part has to be used in the context of data associated with the connection. The Script ID is a unique identifier associated with the script for the connection. The script ID itself should be an indicator of the level of processing the script can be used to do. Suppose a script is to be used for the 3^{rd} level of processing in a connection. Then its script ID should be 3. It is possible that the result of a computation at the router spans across multiple segments. In this case, these segments are logically related in that all of them put together will form the complete result. So, it is necessary to be able to order these segments at the receiver. Sub-sequence number is the offset of the data contained in the segments that are logically related.

   - *A, Sub-sequence number:* Here A stands for ACK. this option is referred to as TACK option in further discussion in this document. This option is set when the receiver receives an out of order segment. The receiver acknowledges the last successfully

received byte in the sequence number field of the header and sub-sequence number of the segment in this option data field. The sub-sequence number information contained in the option data is relevant only to the router used just before the receiver node. This point is discussed in greater detail in the chapter on retransmissions.

- *F, Sub-sequence number:* Here F stands for finish. This sub-type indicates that all levels of processing have been completed on the segment identified by Sequence number, Sub-sequence number tuple and the data just needs to be delivered to the application without further processing.

CHAPTER 4. IPP LAYER MODULES AND DATA STRUCTURES

The IPP layer structure can be broken up into per-connection and global entities (shown in figure 4.1). Per-connection entities are used to keep information related to an open IPP connection whereas global entities are needed for the overall working of the IPP.

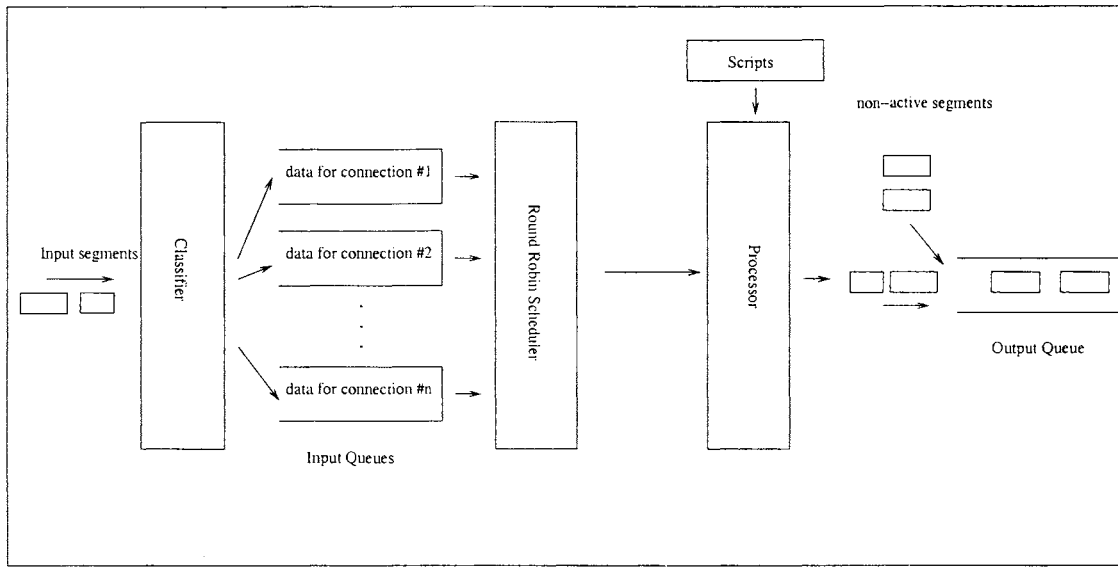

Figure 4.1 Modules and data structures in an active router

4.1 Per-connection data structures

A router uses a unique connection identifier to differentiate among various open active connections. In addition, an active router maintains the following data structures per open active connection:

- Input queue: The First In First Out input queue stores all the data for a connection yet to be processed. When the connection queue is scheduled for processing, data is removed from the queue. The amount of data removed from the queue is based on the amount of data required to complete one round of processing. This follows from the assumption

that the data is sent in such a form that there are demarcations between parts of data that are not used in the same round of computation.

- Reservation timer: The reservation timer is used to keep the processing reservation at the router. When the reservation timer for a connection goes off, the data structures for the connection are freed by the router.

- Processing timer: The processing timer is used to provide an upper limit to the time that the router can allot per round of computation for a connection. When this timer goes off, the router halts processing activity for the connection and sends a Processing Timeout Error message to the sender of the data.

4.2 Global modules at a router

- Packet classifier: This modules classifies the incoming IPP segments according to their corresponding connection and places them into their corresponding input buffers. Connection identifiers are used to classify the segments.

- Round robin processing scheduler: The round robin processing scheduler schedules connections for processing of their segments in a round robin manner. There is an upper bound to the time it takes before a connection is scheduled and that is in the form of the timers provided for a connection.

- Script store: The script store stores the scripts for the connections differentiated based on their connection identifiers.

- Output Queue: The processed IPP segments for all the connections are sent out to an output queue where they are dispatched to the IP layer.

- Interpreter: An interpreter is required at the IPP layer that can decipher the scripts and execute them.

4.3 Connection identifiers

A connection identifier is a combination of the following parameters:

1. The IP addresses of the end hosts participating in the communication.
2. The port numbers of the end hosts participating in the communication.
3. The direction of flow of data that needs to be processed at a router.

Connection identifier should also depict the direction in which the data is flowing. This is required to be able to distinguish between scripts that are to be used for upstream and downstream communication of data. It is possible that both the end hosts require intermediate processing. In this case, script identifiers will become ambiguous if the direction of flow of data is not maintained. The reservations on the routers are linked with the direction of the data flow too. If two hosts A and B are communicating, and both need intermediate processing for the data they are exchanging, they may have two different sets of routers for each direction. In case reservation cannot be made in one of the directions, the associated host may decide to do the computation itself. By associated host it is meant the host that is responsible for the computation.

4.4 Sub-sequence numbers

The data that comes to a router for processing is carried in segments that are numbered by the sequence number of the first byte of data they contain. When the data undergoes processing at a router, it is possible that the size of the result is different from the size of the input data. So, the sequence numbers that originally counted the data bytes are not useful anymore. But the original sequence numbers are required so that the receiver can acknowledge the sender of the data bytes it has received. Thus we need a means of assigning sequence numbers to the resulting data at a router while retaining the original sequence numbers. Consider the following example:

Example:

Suppose there is one router reserved between the sender and receiver end hosts. This router receives data bytes with sequence numbers 35 to 452 from the sender at some point of time. This data may have reached the router encapsulated in one segment or in multiple segments. The router may require byte numbered 35 to byte number 257 for completing one round of computation on the data. Suppose the result of this computation is of size 100 bytes. Suppose the router allocates 2 new segments for this resulting data and fills 35 in the sequence number field of the TCP (IPP) headers of both of the segments. These two segments, which cannot be differentiated by means of their sequence numbers are called logical segments.

Suppose the first 59 bytes of the result were filled in one segment and the remaining 41 in the other. Now, the router assigns a sub-sequence number of 1 to the first segment and a sub-sequence number of END to the other. Generalising, the last segment in a set of logical segments will have the sequence number END. The sub-sequence numbers are really the offsets of the resulting bytes except that assigned to the last segment. The receiver will now wait to acknowledge original sequence number up to 257 to the sender until it receives all the logical segments corresponding to that sequence number correctly. The receiver will receive two segments with the same sequence number, but different sub-sequence numbers. The length fields of the segments when added, give the total size, i.e. 100, of the data received. The length of original data field gives the actual data bytes to be acknowledged.

Summarizing, the sub-sequence numbers are used:

- by any up-stream routers to order the logical segments which may then be processed.
- by the receiver so that it receives the complete results corresponding to a set of original bytes sent by the sender.

4.5 Scripts

The motivation is to use router processing capacity rather than router intelligence to do intermediate processing customized to the application requirements. The end hosts provide instructions to the router in the form of a program that is downloaded at a router after connection set up. The scripts could be generated at the time when the application at the end

host is being generated too. In that way, the application will have a set of scripts associated with it and when the application is run, scripts would be transferred to routers as per the protocol guidelines.

4.5.1 Execution of scripts at a router

The routers themselves would have a run-time environment at the kernel level that would support running of the scripts supplied by the end host. The script executables should be in a format supported by the active routers. The protocol does not place as limit on what programming languages should be used to supply scripts to the router, but it is necessary for the routers and the end hosts to agree upon an execution environment that the end hosts can supply scripts for.

4.5.2 Support for multiple levels of computation

An end host may require multiple levels of processing on certain data and multiple routers may commit processing at different levels. For example, if an end host needed to prune a set of records on the basis of two search parameters, one way is to look for records which satisfy both the search parameters at the same time. Another way is to break up the computation into two levels, by searching the records for one parameter first and then searching the results of the first search to look for records satisfying the second parameter also. These two levels of computation can be delegated to the routers and scripts provided to do the respective levels of computation. A router has the knowledge of whether there is a next level of computation that has to be performed on the results it is forwarding to the next hop node. The router knows this from the list of routers it receives in the form of IPAPR option. So, once it is done with the processing of the data, the new segments that are generated should have the new next level of script identifier, or F flag set in the TIP option if all processing is over on the data.

CHAPTER 5. HANDSHAKE AND CODE TRANSFER

5.1 Connection set up

When two IPP enabled hosts wish to communicate they need to establish a connection between them. Either of the hosts may initiate the connection or both the hosts may initiate the connection at the same time (shown in figure 5.1). During the connection set up, they need to reserve a set of IPP enabled routers if they require intermediate processing to be done on the data. For this purpose, the protocol requires that routers willing and capable of supporting intermediate processing inform the end hosts. One of the end hosts will actively pursue the

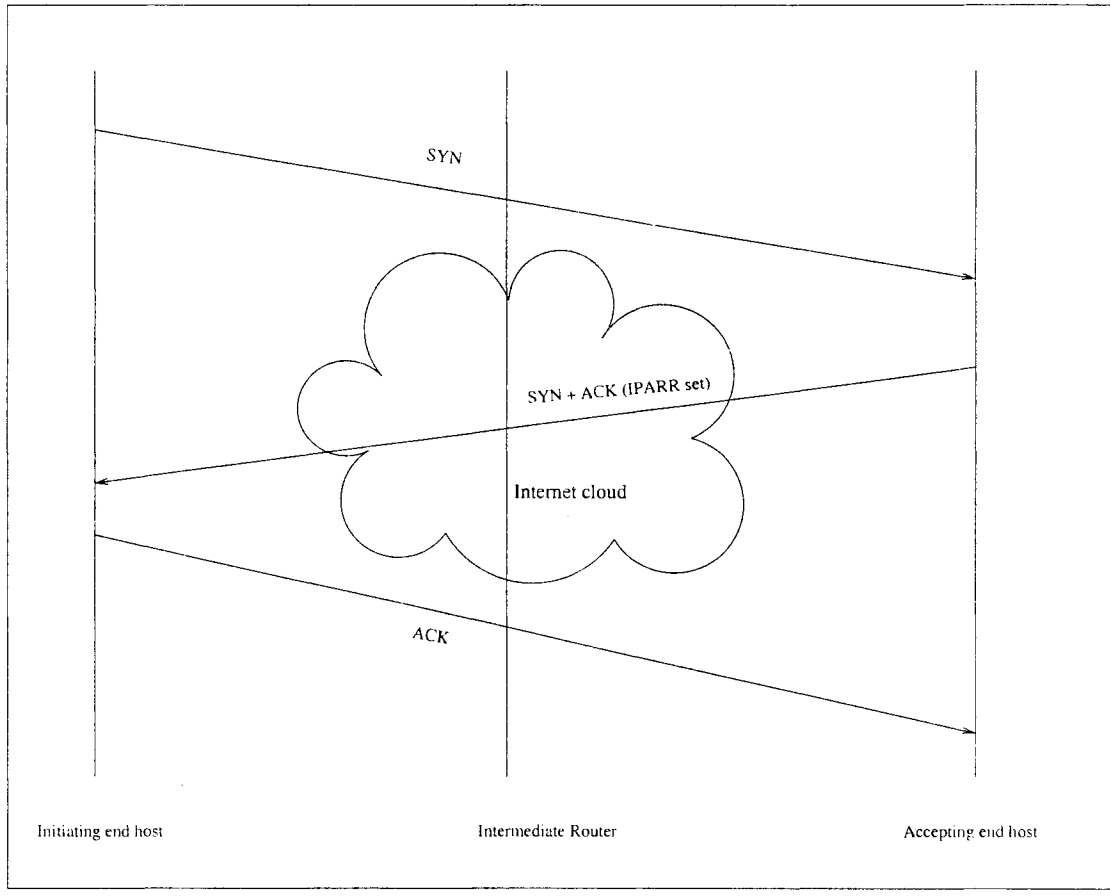

Figure 5.1 Connection Set up using the three-way handshake

connection set up and the other will wait passively for connection requests. The connection set up is depicted in the following sequence of steps:

1. An end host application waits at a well-known port for connection requests from other hosts wishing to communicate. The port number to which the end hosts attach themselves to should be indicative of whether the system is sender based or receiver based or both.

2. The end host receives a connection request (SYN) from a client.

3. If the connection is acceptable to the passively waiting end host, it replies with a corresponding SYN and ACK. This SYN+ACK segment also has the IPARR IP option set if the acknowledging end host needs intermediate processing to be done on the data. This tells the intermediate active routers that this particular connection needs to make use of processing services.

4. The initiating end host receives the acknowledgment and sends an ACK in return and enters the ESTABLISHED state. When the accepting end host receives the ACK, it enters the ESTABLISHED state.

5. When a router receives an IP datagram containing an IPARR option, it assumes that a connection would be set up. Based on the source and destination IP addresses and port numbers, it generates a connection identifier and initializes some data structures for the connection. This connection identifier must also take into account the direction in which the data would be flowing. If both the end hosts would require intermediate processing, two separate connection identifiers must be generated to prevent ambiguity. At the same time, the routers that commited processing start off their reservation timers.

6. A participating IPP router collects information about sequence numbers to be used in the connection by copying the initial sequence number from the sequence number field of the SYN segment. This SYN segment is passed to the IPP layer since IPARR option is set in the IP options field.

7. At the time SYN is received, there is no existing connection information, hence the IPP layer checks if it is possible to accept a new connection. If it is not possible for the router to accept the router reservation request, the request is ignored. If the router is ready to support intermediate processing for this connection, the initial sequence number from the sequence number field is noted. This would help the router order segments for the connection. When the connection accepting end host receives this SYN segment, it sends a SYN and ACK and also the acceptable Maximum Segment Size (MSS).

8. An end host receiving an IPARR (request) message copies the list of IP addresses in IPARR (reply) message in the next outgoing segment sent to the requesting end host.

9. Both the accepting host and the initiating host are in ESTABLISHED state.

Any routers not supporting IPP are not reserved and any router hop may be taken to go from one reserved router to the next reserved router. The protocol only requires that the reserved routers be always traversed and in the same order. Each ACK sent out by a receiver passes through the intermediate routers in the opposite order that they were traversed up to the receiver. This ACK is really the ACK segment with the TIP TACK option set. It carries information about the original sequence number acknowledged, sub-sequence numbers acknowledged and window updates. The acknowledgments carry IPARR or IPAPR option based on the connection state and these options identify the active segments to the active routers. The ACK traverses the route to the sender end host passing through the same set of routers as listed in the IPARR option field of the ACK. When the ACK for the SYN is returned along this route, the router would also note the MSS value for the next router. The router that is immediately preceding the receiver end host receives this ACK and notes the MSS value for the receiver. It then overwrites the MSS value in the ACK segment with its own MSS value.

5.2 Router Reservation

A router may monitor the CPU usage over a period of time and decide whether it is capable of supporting any intermediate processing requested by active applications. It also needs to decide how many of such connections it can support at one time so that the performance in terms of time taken for the computation is reasonably good. A router commits itself to a connection by placing its IP address in the list of routers that the IPAPRR option carries with it. The reservation at a router is really the reservation of its CPU cycles and memory that can be used for active processing.

The reservation is a forward reservation. The reservation is a soft state reservation in that the requesting end host must keep refreshing the reservation at the router by sending out Reservation Refresh messages to the router. The reservation at a router remains till it keeps receiving Reservation Refresh messages from the corresponding end host.The sender may send reservation refresh messages to refresh the soft state reservation at the routers. These reservation refresh messages travel all the way to the receiver that acknowledges the message. Reservation timeout may happen at a router before it receives such a refresh message. In that case, the router deletes the reservation state without informing the end hosts. The next time the sender tries to send data along the route that it assumes is still reserved, the first active router on the way sends it an error message indicating that there is no connection. Based on this, the sender may decide to re-establish the connection.

Because non-existent reservations are detected at the first active router along the way, the protocol avoids a possible flurry of error messages from all the routers along the way. This was a possibility if the data were allowed to travel further down the network. Periodic transmission of refresh messages by hosts and routers is expected to handle the occasional loss of a Reservation Refresh message. If the effective cleanup timeout is set to K times the refresh timeout period, then a router can tolerate K-1 successive Reservation Refresh message losses without falsely deleting state. The network traffic control mechanism could be statically configured to grant some minimal bandwidth for Reservation Refresh messages to protect them from congestion losses.

When data arrives at an intermediate router, it checks the sequence number and then the corresponding sub-sequence numbers. The sequence number of the incoming segment is checked against the current largest sequence number available at the router for that connection. Duplicated segments or missing segments can be detected by sequence number check. If an out of order segment is detected, a NACK is sent to the previous router. This NACK is just a duplicate ACK addressed to the previous router and with the option TIP TACK set along with the sequence number and sub-sequence number set to that of the last received segment.

5.3 Code transfer

If it is a sender based system, the sender needs to transfer the required scripts to the routers. Then the sender can send the unprocessed data which is intercepted by the routers and sent to the receiver. If it is a receiver based system, the receiver needs to transfer the required scripts to the routers. The sender will send the data and the routers will perform the computation that the receiver should have performed on the data upon receiving it. Figure 5.2 depicts the steps involved in the transfer of code.

In the following points in this section, an end host refers to the sender or the receiver based on the type of the system.

1. The end host IPP forms code segments with IPAPR option set and also TIP option set to Code.

2. The router IP layer passes the defragmented segment to the IPP layer. The IPP layer processes the TIP Code option by downloading the script if the level of the script matches the level of processing the router committed to.

3. The code segments reach the other end host which then acknowledges the receipt of segments. The acknowledgement of the receipt of code informs the sender of the code that the code has been correctly received at the routers and the receiver.

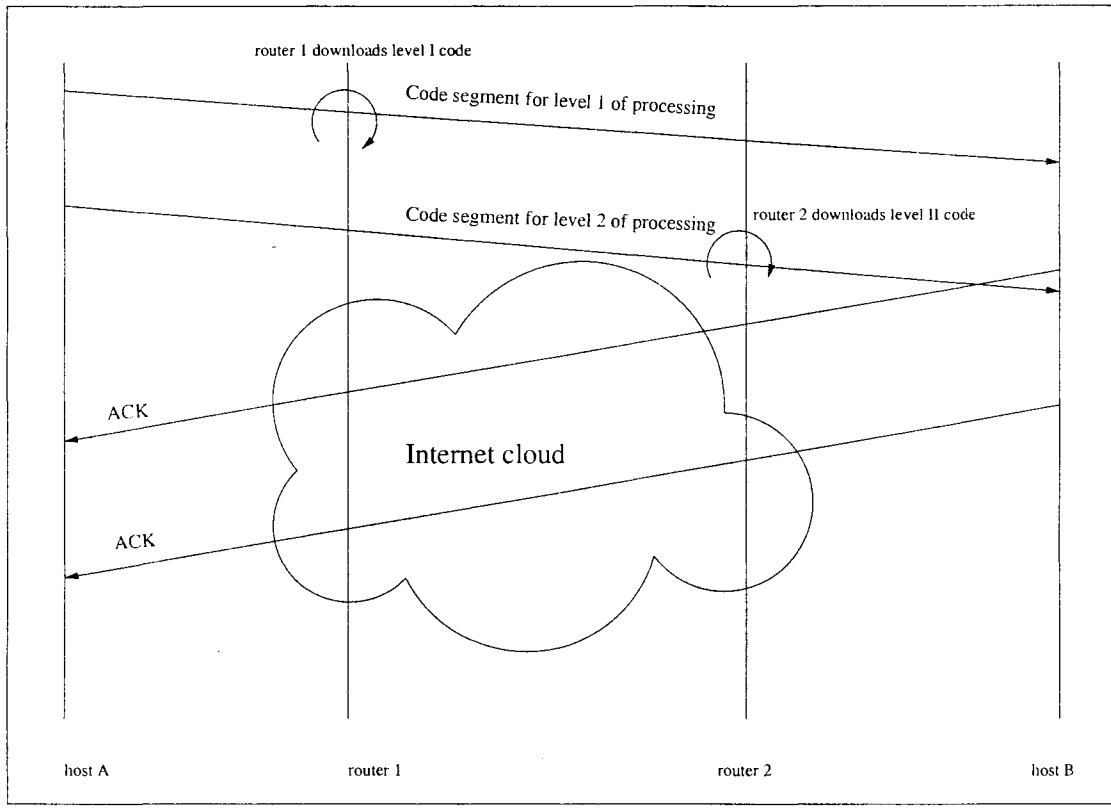

Figure 5.2 Code transfer to the intermediate routers

CHAPTER 6. PROCESSING AT THE IPP LAYER

6.1 Data transfer and processing

The reserved routers perform computation on the connection data based on the scripts supplied by the applications. Figure 6.1 depicts the steps involved in data transfer and processing.

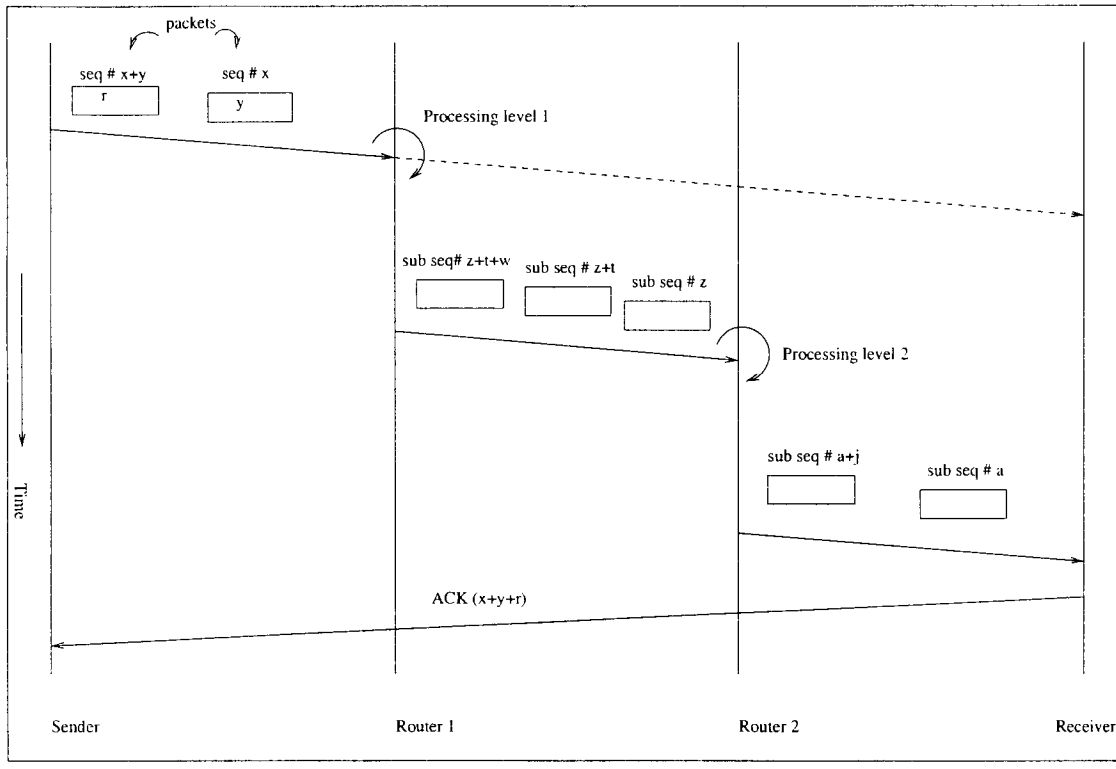

Figure 6.1 Processing at the routers

1. The IP layer of a node supporting IPP receives a datagram with IPAPR option set. The IPP layers checks this and passes the IPP segment to the IPP layer, after defragmentation, if needed.

2. The segment classifier at the IPP layer places the incoming IPP segment in the input

queue corresponding to the connection.

3. The round robin scheduler schedules the connections in a round robin manner. A router may allocate a certain amount of CPU time to be distributed among all the open active connections passing through it. The round robin scheduler may schedule the processing according to the time allocated to each connection.

4. At this time, the processing timer for the connection is started.

5. The router CPU performs the computation on the data and packs the resulting data into segments. It is possible that the size of the resulting segment is very small. In that case, the router may decide to club results from multiple rounds of computation for one connection into one segment.

6. If the processing is complete before the processing timer goes off, the processing timer is stopped. Else, if the processing timer goes off, the router halts processing for the connection and sends an error message to the sender of data.

7. The resulting segments are placed on the transmission queue where non-active packets are also placed in the order they arrive.

8. These packets are scheduled to be dispatched to the IP layer as it would be done in the normal case.

9. During any of the above steps if the reservation timer goes off, the processing for that connection is halted, the information about the connection is deleted and the data for that connection is discarded.

6.2 Segmentation rules

An IPP segment is used to carry both code and data for a connection. Option data carried in the TCP (IPP) and IP headers are used by a router to differentiate between code and data. The router also comes to know from the options as to what script has to be run on a particular piece of data. There are certain constraints to how the data and code have to be

arranged in a segment and limits to the information that header options are allowed to carry in a segment. These rules have been framed in order to let the routers unambiguously decide what computation has to be done on data and whether data or code is carried in a segment.

- Scripts for different levels of processing must be sent in separate segments so that Code, Script ID part of TIP option unambiguously indicates the level of script carried in it.
- Unprocessed data and processed data must be sent in separate segments. This is to avoid ambiguity that will arise due to some part of the segment requiring a certain script to be run on it and the other part of the segment being already processed.
- Parts of data requiring different scripts to be run on them must be sent in separate segments.
- Code and data must be sent in separate segments.
- Data that has been processed fully may be clubbed with other fully processed data for the connection. This implies that a router in the chain of processing routers may place results from consecutive rounds of computation in the same segment.
- The data should be in such a form that the end hosts and the routers can identify how much data is required to complete one round of computation. For example, if the data is in the form of records, there could be demarcations between records so that the end hosts and the routers are able to identify the boundaries of the records.

6.3 Router buffering and retransmissions

The computation done on data may result in the size of data that is different from the input data. If there is any part of the result lost on the way to the receiver, it will trigger retransmission of the data from the sender. The retransmitted data would have to undergo the processing at various routers again and this may lead to delay. There is also no guarantee that this time the results will reach the receiver in their entirety. So, it would be helpful to

have the routers buffer the results for a pre-determined amount of time in case retransmissions are required.

In this system consisting of active hosts and routers, the acknowledgment process is broken down into various segments. For an active host or router, the immediate predecessor is responsible for data buffering and retransmission, if required. In this case, there is only local acknowledgement of data: between successive active hosts and routers. The acknowledgement consists of sequence number and the sub-sequence numbers of the bytes of data acknowledged.

M1 receives an ACK before its retransmission timer goes off. Retransmission timer at M1 runs out before an ACK is, if at all, received. In the first case, M1 removes the acknowledged sequence numbered data from its retransmission queue and updates its count of acknowledged sequence numbers for the connection. In the second case, M1 retransmits the data to M2 and starts its retransmission timer. Between M2 and M3 similar process of local acknowledgment and buffering takes place. M2 deletes the data from its retransmission queue once it receives a local acknowledgment from M3. M2 starts its retransmission timer when it sends out the data for M3. In case of data loss between M2 and M3, the data is furnished by M2 from its retransmission queue. By use of local acknowledgement, we avoid the overhead of buffering data at each of the routers until an acknowledgment from the receiver end host is received. This approach involving local acknowledgments also provides the latest form of processed data unlike a retransmission of unprocessed data by the sender end host.

Receipt of an acknowledgment would influence variables such as receivers window, congestion window, data buffered for retransmission and retransmission timer status.

If a router encounters lost data (out of order segment), it sends a NACK to the previous router in the chain requesting retransmission. This lost data segment is identified by means of (sequence number, sub-sequence number) tuple. Sub-sequence number can be used to point out a segment from a set of logical segments which in turn is identified by means of an original sequence number. When all the logical segments reach the end host successfully, it sends an ACK to the previous IPP router. Any data for retransmission is thus made available by the previous IPP node in the chain as opposed to by an end host.

6.4 Connection Tear Down

There are two cases:

1. Sender based system

   (a) Sender initiated CLOSE In this case, by initiating the FIN, the sender indicates that it has no more data to send. In the meanwhile, the sender needs to keep the reservation at the routers alive until it receives acknowledgment of all the data that it wanted the receiver to receive. For this, the sender continues to send Reservation Refresh messages periodically until it receives the ACKs for all the bytes transmitted. So, even though the application does not have data to send, the IPP must keep the reservation alive by sending Reservation Refresh messages.

   (b) Receiver initiated CLOSE In this case, the receiver does a CLOSE as it would happen normally. The receiver does not have any reservation at the routers.

2. Receiver based system

   (a) Receiver initiated CLOSE In this case, the receiver indicates that it does not have any data to send and closes the connection normally.

   (b) Sender initiated CLOSE In this case, by initiating the FIN, the sender indicates that it has no more data to send. When the receiver receives this FIN, it needs to tear down the reservation it has on the routers. The receiver keeps refreshing the reservation at the routers unless it receives the byte sequence numbers till the FIN segment sequence number. So, even though the sender application does not have data to send, the receiver IPP must keep the reservation alive by sending Refresh messages.

At the time of connection tear down, the end hosts need not inform the routers that the reservations are no longer needed as the routers can delete the reservation information when the Reservation Timer for the connection expires.

CHAPTER 7. IPP CONTROL MECHANISMS

7.1 Flow control

Flow control is done by taking into account the receiver window of the next active node in the connection. This is done by attaching a window update in the acknowledgment option field. The previous router stores this value of window update for the next router and it is used at the time of next transmission to the router for that connection. At the time of transmission, a router or an end host takes into account the receiver window size of the next active router (or host) and its own congestion window. This flow control is local between pairs of active nodes (hosts and routers). The issue of flow control is being dealt with at each IPP router rather than by the end hosts alone. An IPP router could decide its receiver window size to report based on the amount of incoming data that is being taken up for processing and dispatching to the next IPP node.

7.2 Congestion control

Each of the intermediate routers has information about the next router's receiver window and its own value of congestion window based on the network state. So it has to output the data to the next router (the data is possibly going via many normal routers to reach this next active router) with respect to these two parameters. The size of the data remains constant between any two successive active routers and router-host pair. The minimum of receiver window and congestion window governs the amount of data that can be sent. The congestion window size is decided based on the congestion control algorithm that is being deployed at the moment [5].

7.3 Security

There are two parts to the security mechanism that is needed when intermediate routing is employed:

1. Security mechanism at the routers participating in the processing.

2. Security mechanism at the end hosts participating in the processing.

7.3.1 Router security

Intermediate processing involves downloading a piece of code by the routers and executing it on the data supplied by the host. This can pose security threats for a router in may ways. So there has to be a mechanism at the routers so that any abnormal use of router resources is identified and curbed.

1. The code supplied by an end host may try to consume an unlimited amount of router resources, e.g. processing time, router memory etc. leading to a possible resource starvation to other applications.

2. The code supplied may try to change the routing information for other connections at the router.

3. An end host may reserve the router for processing and never send any data.

The first problem can be taken care of by placing an upper limit to the router resources that can be made available to active applications. Processing timer ensures that no application engages a router CPU in indefinitely long computation. An upper limit to the size of input queue for a connection ensures that no application can occupy indefinitely large memory space at a router. The second problem can be addressed by restricting the memory access by the code to only the pre-allocated area of router memory. The processing should be aborted if the code tries to access any part of user or kernel memory outside its allocated space. The router reservation is a soft state reservation which means that the reservation is torn down

after a finite amount of time in case it is not renewed by the host that originally requested the reservation.

7.3.2 Host security

To make use of intermediate processing on a byte stream of data, the end hosts make sure that all packets pass through the same set of routers and in the same order of traversal of routers. Making all data pass through the routers can pose a security threat because the routers can possibly put together all of data and actually make sense out of it. This is less of a possibility in the normal IP routing where the route between is decided packet per packet.

1. The router may try to use scripts supplied by an end host for a connection for another connection.

2. The router may try to modify the scripts to its advantage.

The data could be encoded in such a form that only the scripts for that connection would be able to decipher it. This may prevent usage of scripts for other than the connection owning them in a way that the scripts may not be able to decipher data encoded for other connections. Scripts would normally be executables that are ideally impossible to reverse-engineer. This is so that a malicious router does not modify the script to use it to its advantage.

7.4 Fault recovery

A network is susceptible to node and link failures that disrupt the connections passing through those nodes. To ensure reliable transfer of data between the end hosts, fault recovery mechanisms are required. There has to be a provision for retransmission of data that might have been lost due to network failures. Since the routers are also modifying the data, and the receiver receives data different from what the sender sent out, there is scope for loss of data without the knowledge of the end hosts if one of the nodes along the route crashes. The effect of the case when one of the reserved routers fails is different from the case when one of the end hosts fails and recovery mechanism has to be different accordingly.

1. *End host failure:* If one of the end host IPP crashes and loses all the memory of sequence numbers it has been using, the connection would have to be re-established. RFC 793 states that to be sure that a TCP (hence IPP here) does not create a segment that carries a sequence number which may be duplicated by an old segment number remaining in the network, the TCP (IPP) must keep quiet for a Maximum Segment Lifetime (MSL) before assigning any sequence numbers upon starting up or recovering from a crash in which memory of the sequence numbers in use was lost. If a TCP (IPP) is reinitialized in some sense, yet retains its memory of sequence numbers in use, then it need not wait at all; it must only be sure to use sequence numbers larger than those recently used.

2. *Router failure:* In this case, the end hosts have the knowledge of what sequence numbers have been acknowledged and where they should pick up from in the next incarnation of the connection. The connection needs to be re-established because there is a possibility that the router that crashed may not recover. So, the end hosts must reroute the connection through other routers. It is possible that the same router may be a part of the next connection also. In this type of failure, both the end hosts have the memory of the sequence numbers in use before a router crashed. To be sure that a TCP (IPP) does not create a segment that carries a sequence number which may be duplicated by an old segment remaining in the network, the end hosts must make sure to use sequence numbers larger than those recently used.

CHAPTER 8. IPP HEADER FORMAT AND MESSAGE TYPES

8.1 Header Format

The IPP protocol is based on TCP and uses the same header format. The inherent concepts as used in TCP remain intact as additional information for intermediate processing is forward to the routers and end hosts using TCP segments. Introduction of new TCP and IP options provides a vehicle to carry code and data information to the routers.

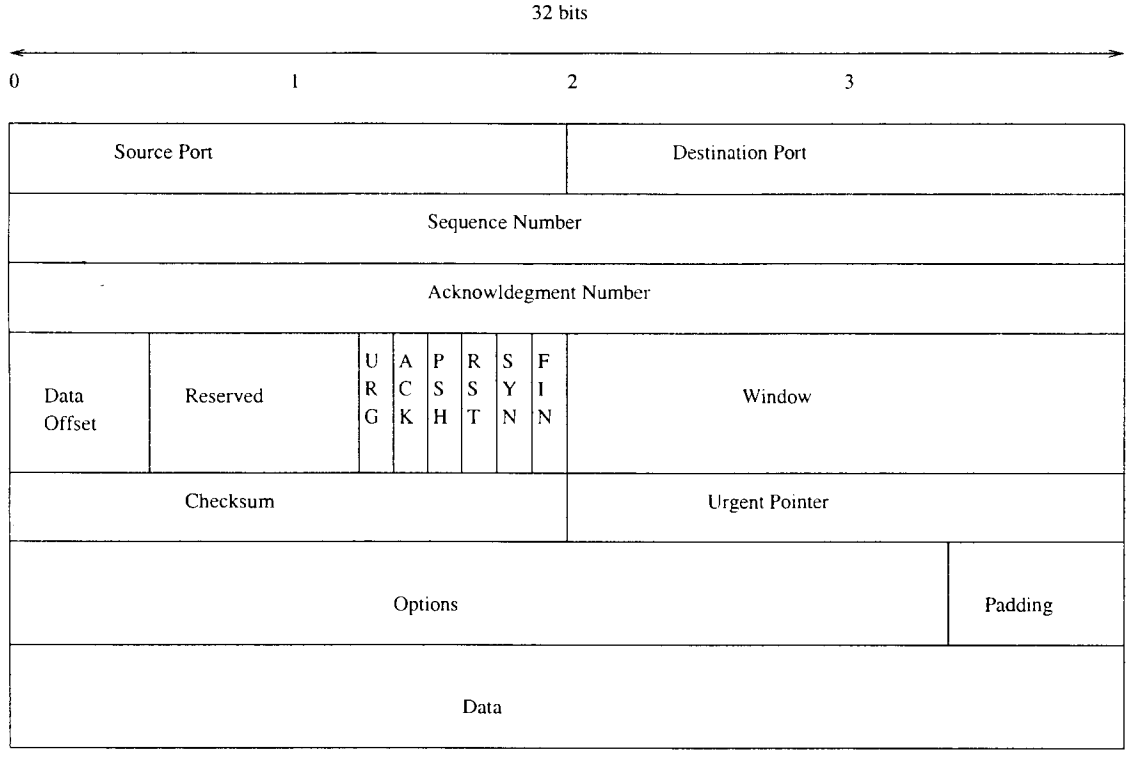

Figure 8.1 TCP header format

- Source Port: (16 bits) The source port number.
- Destination Port: (16 bits) The destination port number.

- Sequence Number: (32 bits) The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

- Acknowledgment Number: (32 bits) If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

- Data Offset: (4 bits) The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.

- Reserved: (6 bits) Reserved for future use. Must be zero.

- Control Bits: (6 bits, from left to right):

  URG: Urgent Pointer field significant ACK: Acknowledgment field significant PSH: Push Function RST: Reset the connection SYN: Synchronize sequence numbers FIN: No more data from sender

- Window: (16 bits) The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

- Checksum: (16 bits) The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros. The checksum also covers a 96 bit pseudo header conceptually prefixed to the TCP header. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in the arguments or results of calls by the TCP on the IP.

- Urgent Pointer: (16 bits) This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set.

- Options: (variable) Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum. An option may begin on any octet boundary. There are two cases for the format of an option:

  Case 1: A single octet of option-kind.

  Case 2: An octet of option-kind, an octet of option-length, and the actual option-data octets.

  The option-length counts the two octets of option-kind and option-length as well as the option-data octets.

8.2 Types of messages

Other than the normal TCP control messages, the protocol requires the following additional messages:

1. *Reservation Request messages:* in the form of IPARR messages. This option is set in the SYN+ACK or ACK segment at the time of connection set up and hence does not require separate message to carry this information. Out of the two end hosts, the end host requiring intermediate processing has to send out these messages with IPARR (request) option set.

2. *Reservation Reply Messages:* An end host receiving a segment containing the IPARR option (request) set would need to reply to the message by setting the IPARR (reply) option of the next outgoing segment. These messages confirm whether the reservation of router(s) requested by the other end host were successful or not. The host sending out IPARR (reply) message copies the list of IP addresses (possibly empty) from the IPARR (request) message into the list of IP addresses field of IPARR (reply) message.

3. *Reservation Refresh messages:* to maintain the soft state reservation at the routers. Out of the two end hosts, the end host requiring intermediate processing has to send out these messages. These messages are sent with zero data and IPAPR option set. That indicates to the routers and the receiver that it is a reservation refresh message and carries no data to process. The receiver acknowledges the receipt of the message.

4. *Non-existent Reservation Error Messages:* that will be generated by a router when it receives a segment with IPAPR option set but the connection information for that segment does not exist. This error message has to be sent to the end host that is responsible for the computation and not necessarily to the sender of the segment. The end host on receipt of this error message may decide to reserve routers for the computation. The end host, may take one of the following actions:

   - If this end host had done an active opening of the connection (like connect()), it may try to reconnect.
   - If this end host had passively accepted the connection (like accept()), it may reset the connection (send a RESET to the other end host) and wait for new connection requests. An end host should ignore any error messages for a non-existent connection.

5. *Processing Error messages:* will be generated at a router and would be sent to the sender of the data on which processing failed. These error messages carry the starting sequence number of the data on which processing failed. The sender of data checks the sequence number to avoid processing of any error messages generated in a previous incarnation of the connection.

6. *Duplicate acknowledgment option messages:* are sent by an IPP node when it receives an out of order segment. These messages are like the TCP duplicate ACK messages, except that here the TIP option TACK is also set which lets the previous node know that it a a retransmission request message.

CHAPTER 9. AN IMPLEMENTATION OF IPP ON LINUX

9.1 Implementation of IPP on Linux 2.2.17

An experimental implementation of IPP was done on Redhat Linux 2.2.17. The goal of the implementation was to demonstrate the working of IPP highlighting the concepts of connection set up, router reservation, the working of TIP, IPARR and IPAPR options, data transfer, intermediate processing and reliable communication. The following features of IPP were demonstrated by means of implementation:

1. IP options IPARR (IP Record Route) and IPAPR (IP Process Route)
2. TCP option TIP (TCP Intermediate Processing)
3. Mechanism for router reservation as a part of TCP connection set up handshake procedure.
4. Mechanism for code transfer as a part of TCP connection set up handshake procedure.
5. Mechanism to maintain active connection information at active routers and end hosts. This information includes routing information, scripts and buffer data structures.
6. Mechanism for generating new TCP (here IPP) segments at an active routers after computation has been done on original data. This accomodates expansion/ contraction in the size of original data as a result of processing. This required sending out additional information with the data to help the end host in ordering the result segments.
7. Mechanism at active hosts for acknowledging the original number of bytes sent out by the originating host irrespective of how many result bytes it received from an active router. This required changes to the way TCP handles sequence numbering of data.
8. Network packet analyzer tool "tcpdump" and "Ethereal Network Analyzer" were used to check the correctness of the implementation.

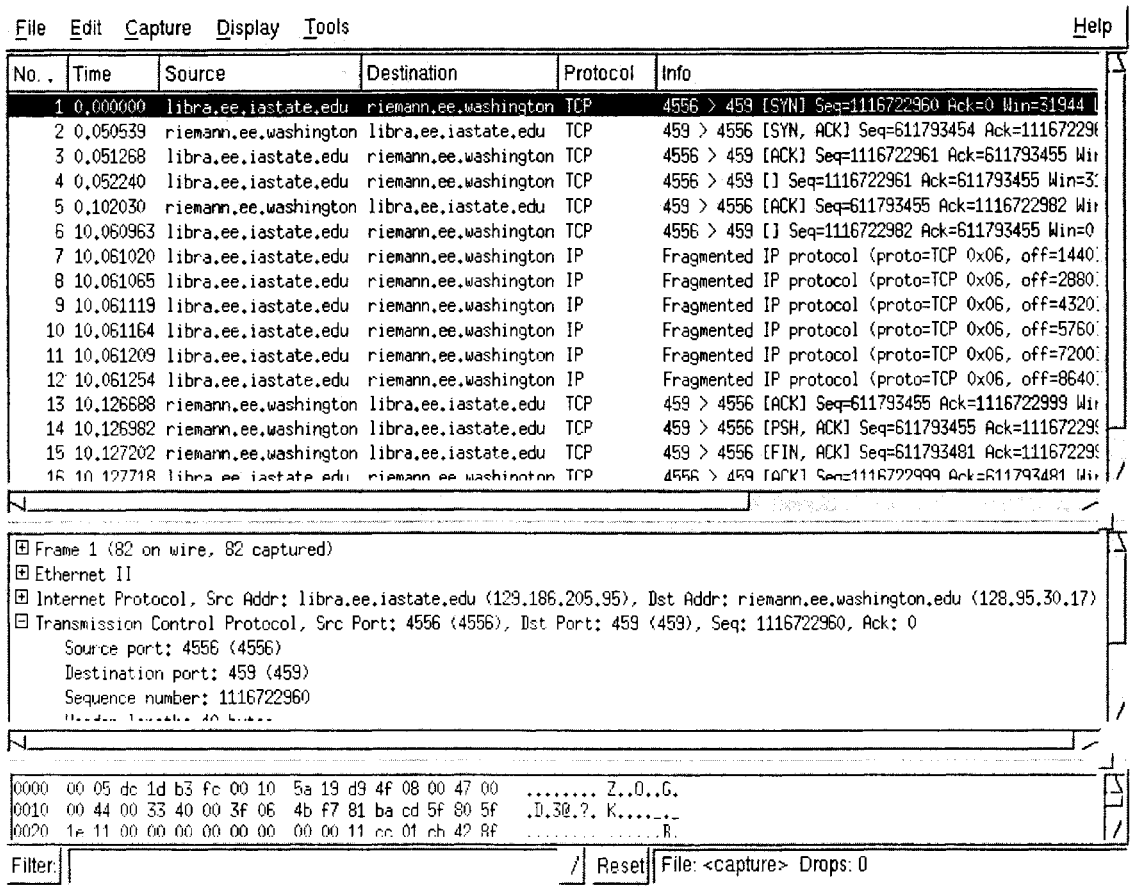

Figure 9.1 Ethereal Network Analyzer tool

9.2 Experimental set up

For a demonstration of correct and consistent functioning of the protocol, a set up of 4 Linux machines was used as shown in figure 9.2. The University of Washington machine, shown as Machine 1 acts as a remote server for the two client machines, Machine 2 and 4 respectively located at Iowa State University. Machine 4 acts as a router for the two machines. All these machines have IPP enabled on them. The set up was used to dempnstrate the flow of request and replies between the server and the clients based on IPP rules. The clients request connections from the server at which time the server acknowledges and sends out the code first to be used at the router. Once the connection set up is done, data coming from the server passes through the Internet and Machine 2 to the clients. Machine 2 performs certain

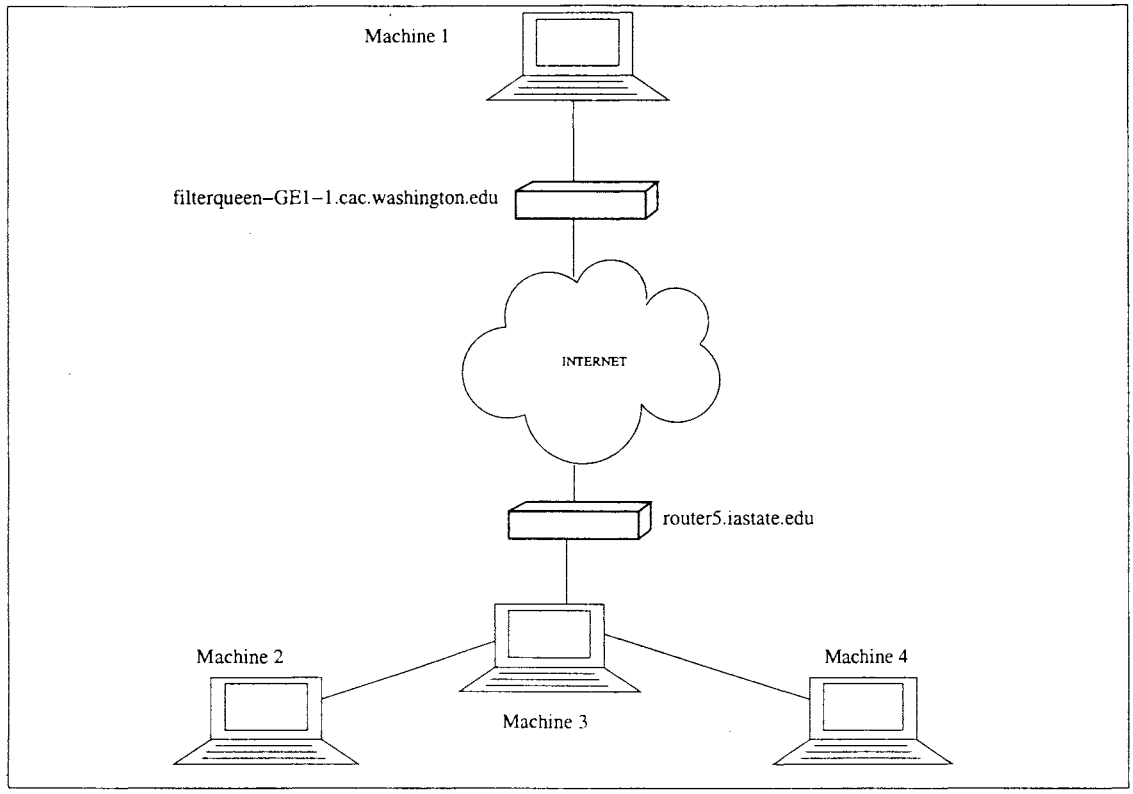

Figure 9.2 Experimental set up for demonstration of IPP

computation on the data before forwarding them to the clients.

There were two scenarios that were tested:

- Server availability: Due to an IPP server sending out unprocessed data to be processed at the routers, it is expected that this increases the server availability. The parameter for server availability that was used was the time it takes for a server to process a certain number of requests with and without the use of IPP.

- Client latency: Due to code transfer overhead and additional checks at the routers performing the computation, it is expected that a client would have to wait longer for getting the results. This latency incurred should be within acceptable bounds. The measure of this latency was the total time it took for a client to send a request to a server and get the results back.

9.3 Future work on IPP

Work in the following areas could be pursued:

1. This work presents and implements the reservation of routers statically, i,e, at the time of connection set up. Dynamic reservation in which a router could decide to process data passing through it on a per segment basis rather than a whole connection basis could be a more flexible approach.

2. As with other mobile code systems, a system deploying IPP is prone to security attacks. Recommendations have been made in the thesis about some security measures, and more rigorous security measures would make an IPP system more robust.

3. Parallelization of intermediate computation and routing decision at a router would not only speed up processing time, but also allow a router to continue its routing function in times of heavy traffic.

4. Reliable communication could be made a configurable option rather than a part of the protocol design so that an application could choose between fast but unreliable communication and slightly slow but reliable communication.

5. More concise scripting languages tailored to be used for IPP would help make the size of code segments smaller and would take up less memory at the intermediate nodes.

BIBLIOGRAPHY

[1] Active Networks. *http://www.sds.lcs.mit.edu/activeware*, date accessed: March 26, 2002.

[2] J. Postel. *Internet Protocol*, RFC 791, Information Sciences Institute, University of Southern California, Marina del Rey, California, September 1981.

[3] J. Postel. *Transmission Control Protocol*, RFC 793, Information Sciences Institute, University of Southern California, Marina del Rey, California, September 1981.

[4] J. Baker. *Requirements for IP version 4 Routers*, RFC 1812, Cisco Systems, Santa Barbara, California, June 1995.

[5] M. Allman, V. Paxon, W. Stevens. *TCP Congestion Control*, RFC 2581, Tucson, Arizona, April 1999.

[6] Dave Wetherall and David Tennenhouse. *Active IP option*, Proceedings of the 7th ACM SIGOPS European Workshop, Connemara, Ireland, September 1996

[7] S. da Silva, D. Florissi and Y. Yemini. *Composing Active Services in NetScript*, ARPA Active Networks Workshop, Tucson, Arizona, March 9-10, 1998.

[8] Multi Router Traffic Grapher, *http://mrtg.hdl.com/mrtg.html*, date accessed: March 26, 2002.

[9] David J. Wetherall, John Guttag, and David L. Tennenhouse. *ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols*, IEEE OPENARCH'98, San Francisco, California, April 1998.

[10] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden.*A Survey of Active Network Research*, IEEE Communications Magazine, Vol. 35, No. 1, pp 80-86. January 1997.

[11] S. Bhattacharjee, K. Calvert and E. Zegura. *Active Networking and End-to-End Arguments*, IEEE Network Magazine, 1998.

[12] K. Calvert, E. Zegura, J. Sterbenz. *CANEs: A Modest Approach to Active Networking*, Presented at IEEE Computer Communications Workshop, Phoenix, Arizona, September 1997.

[13] Ulana Legedza, David J. Wetherall, and John Guttag. *Improving The Performance of Distributed Applications Using Active Networks*, IEEE INFOCOM, IEEE, March 1998.

[14] Mobile Code, Agents and Java. *http://www.infosys.tuwien.ac.at/Research/Agents*, date accessed: March 26, 2002.

[15] Beverly Schwartz, Alden W. Jackson, W. Timothy Strayer, Wenyi Zhou, Dennis Rockwell and Craig Partridge. *Smart Packets for Active Networks*, Presented at Openarch, New York, NY, March 1999.

[16] Micheal Hicks, Pankaj Kakkar, Jonathan T. Moore, Carl A. Gunter and Scott Nettles. *Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming Languages*, pp 86-93, 1998.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, I thank Dr. Arun Somani for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have inspired me and renewed my hopes for completing my graduate work. I would also like to thank him for helping me pursue graduate work in the area of research that I was most interested in. In this regard, I highly appreciate his support.

I would also like to thank my committee members for their efforts and contributions to this work: Dr. Akhilesh Tyagi and Dr. David Fernandez-baca. I would additionally like to thank Dr. Akhilesh Tyagi for providing me an initial insight into my thesis project.

I also take the opportunity to thanks the members of DCNL whose helpful inputs gave me a deeper understanding of the various aspects of my research work.

Last but not the least, I sincerely acknowledge the motivation and help provided by my friend Sharad without whose constant support I might not have been able to complete my graduate education. I am also thankful to my family in India for their continuing moral support.