

A Reliable Protocol for Processing within IP-routed Networks

Sonal Pandey, Arun Somani and Akhilesh Tyagi

Iowa State University

Electrical and Computer Engineering Department

Ames, Iowa 50010, USA

Email: {sonal,arun,tyagi}@iastate.edu

Abstract—The task of an Internet router is to scan the IP headers for the destination address and make a routing decision based on the information. If the processing capability of a router is enhanced to support computation on a datagram, some of the host computation may be delegated to the intermediate routers. The instructions about how to do the processing may be provided by the end hosts. We propose a reliable transport layer protocol, Intermediate Processing Protocol (IPP) for processing within the Internet. The protocol design makes provisions for connection set up handshake, router reservation, intermediate processing, data acknowledgement, buffering and retransmission, flow and congestion control, ordered delivery and security issues.

I. INTRODUCTION

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) [1], the Transmission Control Protocol (TCP) [2], the Internet Control Message Protocol (ICMP) [3], and applications using them. One of the most important constituents of the Internet is an IP router or a gateway [4]. Computation on a datagram at a router is limited to scanning the IP header and updating header information. Lightly to moderately loaded routers use only a part of the CPU processing power [9]. 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.

The ideas discussed here fall into the paradigm of active networks [6] wherein intermediate nodes can be used to perform computations on application data providing more control over certain applications and easier access to certain network-related parameters. Active networks aim to constructively mutate the strong end-to-end semantics of the current transport mechanisms to realize the benefits of customized processing in a network [12] and are a rapidly expanding field of research [11], [13].

One of the earliest projects in this area of research started at Massachusetts Institute of Technology in 1996, [7], an active network architecture in which passive packets were replaced with active capsules. A sequel to this architecture was the ANTS project [10], a toolkit for building and dynamically deploying network protocols. Special languages for use in active

networks are being developed [16], [8], [15]. Further applications in active networking are discussed in [14].

Intermediate processing may be used in client-server applications where the server delivers to the client data based on the client's request. A server may send out an approximate set of results that could be pruned at the routers using the code supplied by the server. This approach may also be used in making available services to intermediate nodes that would otherwise be available only at the hosts. We propose an active reliable transport layer protocol, Intermediate Processing Protocol (IPP), that would facilitate processing of data in IP routed networks. An important goal of the protocol design is to place minimal additional complexity at a router so that normal routing performance does not suffer significantly. The IPP forms the layer above the IP layer. The IPP is based on Transmission Control Protocol and relies on the same underlying concepts as used in TCP. Additional information is carried in TCP segments to indicate the requirement for intermediate processing. At a router, the IPP layer does not function like a full-fledged transport layer and that is important to minimize the processing overhead incurred. The model of network assumed 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. It is required that both the end hosts that wish to communicate using IPP support IPP. The end hosts may have knowledge of the location (IP address) of active routers in the network.

II. TERMINOLOGY

- *Active router*: An IP router that supports IPP.
- *Active host*: A host that supports IPP.
- *Sender based system*: A pair of hosts where the sender host is required to do certain computation on the data before sending them to the receiver host. An example of this is searching a set of records based on search parameters before sending out the relevant records.
- *Receiver based system*: A pair of hosts where the receiver of the data is required to do certain computation on the data received. An example of this is the decoding of MPEG files that a receiver host receives from a sender.
- *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.
- *Connection identifier*: A combination of IP addresses and port numbers of the end hosts and the direction of flow of data at a router.

III. ADDITIONAL HEADER OPTIONS

A. IP layer options

By use of additional IP options, the IP layer at a router is instructed to pass the data up to the IPP layer for processing. For this purpose, two IP options have been introduced at the IP layer of active routers and active hosts:

- IP Active Record Route (IPARR)
- IP Active Process Route (IPAPR)

These options would be set by the host requiring intermediate processing in the datagrams. The IPARR option is used to reserve active routers in a session between two hosts at the time of connection set up. Active routers along the route that the connection request traverses can indicate their capability to support intermediate processing for the session. The IPAPR option is used to indicate to the routers and the end hosts that data carried in the datagram requires intermediate processing. When a router receives a datagram with this option set, it reassembles the datagram and passes it up to the IPP layer. The format of the IP options is as follows:

- 1) Type field contains IPARR or IPAPR identifier.
- 2) Length field contains the length of the IP option including the type and length fields.
- 3) Option data field contains the following sub-fields in IPARR:
 - Levels of processing implies that data needs to be operated by so many different scripts.
 - List of active routers contains the IP addresses of the routers from which intermediate processing is requested. The IP address is that of an interface of the router reachable from the host requesting reservation. The option also ensures that the request passes through these routers in the order they are listed.
 - Nature of Request indicates whether the option is being used for requesting reservation or replying to a reservation request.

In case of IPAPR, option data field contains the list of active routers that need to do the processing. The IP layer at an active router checks for the presence of the IP address of the router in the option data for IPARR and IPAPR before passing it up to the IPP layer.

B. Transport Layer option

The IPP protocol design incorporates a new option, TCP Intermediate Processing (TIP) to be implemented at the IPP layer of active nodes. This option is used to convey information about code and data for a connection. The format of the option is as follows:

- 1) Type field contains the identifier TIP.
- 2) Length field contains the length of the option including the type and length fields.
- 3) Options data field contains the following sub-fields based on the scenario it is used:
 - *C, Script ID*: Here *C* stands for code. This sub-type indicates that the information contained in the options data has to be used in the context of code associated with the connection. The Script ID is a unique identifier associated with the connection.
 - *D, Script ID, Sub-sequence number*: Here *D* stands for data. 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. Sub-sequence number is the offset of the data contained in the segments that are logically related. Sub-sequence numbers are discussed in a later section.
 - *A, Sub-sequence number*: Here *A* stands for acknowledgment. This option, referred to as TACK option, is set while acknowledging receipt of a segment.

IV. IPP CONNECTION SET UP

Either or both the hosts may initiate a connection request. During the connection set up, a host needs to reserve a set of IPP enabled routers if it requires intermediate processing. Fig. 1 depicts the three way handshake during connection set up. Following steps are involved in a connection set up using IPP:

- 1) An application waits at a well-known port for connection requests from other hosts wishing to communicate.

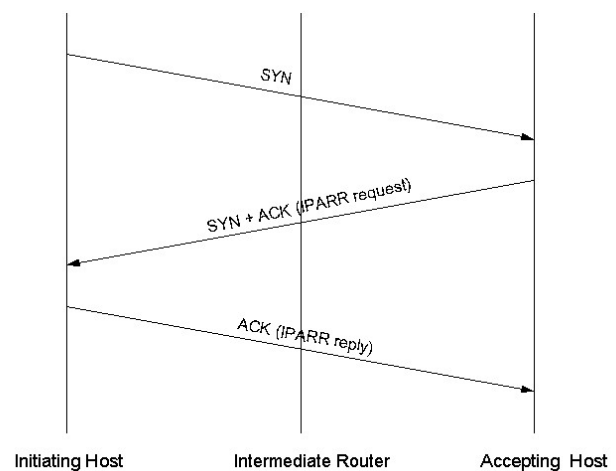


Fig. 1. Connection Set up using the 3-way handshake

- 2) The host receives a connection request (SYN) from a client. If the initiating host requires intermediate processing, it sets the IPARR option in the SYN segment (IPARR request). The host receiving an IPARR(request) message

copies the list of IP addresses in SYN+ACK (IPARR reply) message in the next outgoing segment to the requesting host.

- 3) The SYN segment is passed to the IPP layer since IPARR option is set in the IP options field. 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 not, the router removes its IP address from the list of addresses in the IPARR option. If the router is ready to support intermediate processing, the initial sequence number is noted. This would enable the router to order segments for the connection.
- 4) When the accepting host sends a SYN+ACK, it may also set the IPARR option to request intermediate processing.
- 5) The initiating host receives the acknowledgment and sends an ACK in return and enters the ESTABLISHED state. When the accepting host receives the ACK, it enters the ESTABLISHED state.
- 6) 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 a set of variables and data structures for the connection. At the same time, the router starts off a reservation timer.

V. ROUTER RESERVATION AND CODE TRANSFER

A network administrator may monitor the CPU usage of a router over a period of time and configure the router as active. The reservation at a router is the reservation of its CPU cycles and buffer. The reservation is a forward, soft state reservation in that the requesting host must keep refreshing the reservation periodically. Reservation timeout may happen at a router before it receives such a refresh message at which time the router deletes the reservation state without informing the 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 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. 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 provide a minimal bandwidth for Reservation Refresh messages to protect them from congestion losses.

After the sequence numbers have been synchronized, scripts are transferred to respective routers. In a sender(receiver) based system, the sender(receiver) transfers the required scripts to the routers. Fig. 2 depicts the transfer of code to routers. The host IPP forms code segments with IPARR option set and TIP option set to Code. The router IP layer passes the segment to

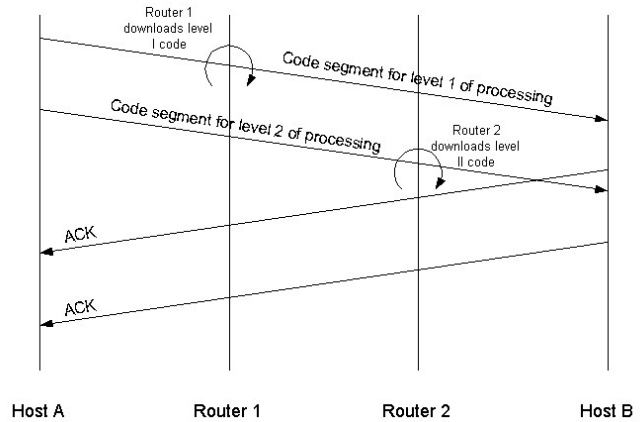


Fig. 2. Code transfer to the active routers

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 committed to by the router. The code segments reach the other end host which then acknowledges the receipt of segments.

VI. PROCESSING AT THE ROUTERS

The reserved routers perform computation on the connection data using the scripts supplied by the application. A processing timer is started when processing begins on connection data. The router CPU performs the computation and packs the resulting data into new segments. If the processing is complete before the processing timer goes off, the processing timer is stopped. If the processing timer goes off, the router halts processing for the connection and sends an error message to the sender. The resulting segments are placed on the transmission queue in the order they arrive. These packets are scheduled to be dispatched to the IP layer as would be done normally. During processing, 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.

It is possible that the size of resulting data is different from that of input data. To account for this, the resulting IPP segments are all given the sequence number of the first byte of data used in that round of computation. These segments are called logical segments. Sub-sequence numbers are used to differentiate between segments of a set of logical segments. A sub-sequence number for a segment in a set of logical segments actually serves as the order of the first byte of data contained in that segment. The subsequent bytes in that segment can be ordered based on this sub-sequence number. A set of sub-sequence numbers for segments is local between pairs of active nodes. This is because sub-sequence numbers are only used to order the segments containing the results of computation but having the same original sequence number. Precise mechanism for performance of computation at the router depends on the implementation.

VII. SEGMENTATION RULES

An IPP segment is used to carry both code and data for a connection. There are certain constraints to how the data and code have to be arranged in a segment and limits to the information header options are allowed to carry. These rules have been framed to enable the routers to decide whether data (processed or unprocessed) 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 and processed data must be sent in separate segments.
- Parts of data requiring different scripts must be sent in separate segments.
- Code and data must be sent in separate segments.
- Data that has been processed fully may be merged with other fully processed data for a connection.
- The data should be in a form that allows the active nodes to 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.

VIII. ACKNOWLEDGMENTS AND RETRANSMISSIONS

The acknowledgment for a segment is really an 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. For an active host or router, the predecessor is responsible for data buffering and possible retransmission. In this case, there is only local acknowledgement of data: between successive active nodes. The acknowledgement consists of sequence number and the sub-sequence numbers of the bytes of data acknowledged.

Example: Suppose there are four IPP enabled nodes, M1, M2, M3, and M4 in a system and possibly some non-IPP nodes. Out of these four nodes, M2 and M3 are routers and the other two are end hosts of an IPP connection. At an instant, M1 is serving as a sender and transmits data to be processed at M2 and M3 to be ultimately delivered to M4. At this time, M1 starts its retransmission timer. When M2 receives the data segments up to those needed to complete one round of computation (one or more segments), it sends a local ACK to M1. M1 records the receipt of this ACK and the acknowledged sequence number. M2 may send a delayed ACK by combining ACKs for several segments. In the case that M1 receives an ACK before its retransmission timer goes off, M1 removes the acknowledged sequence numbered data from its retransmission queue and updates its count of acknowledged sequence numbers for the connection. In the case that retransmission timer at M1 runs out before an ACK (if at all) is received, M1 retransmits the data to M2 and starts its retransmission timer. Between M2 and M3

similar process of local acknowledgment and buffering takes place. Receipt of an acknowledgment would influence variables such as receiver's 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 active router, identifying the lost data segment by means of (sequence number, sub-sequence number) tuple and the connection identifier. 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.

IX. CONTROL MECHANISMS

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 an ACK. The previous router stores this value of window update and it is used at the time of next transmission to the router for that connection. At the time of transmission, an active node takes into account the receiver window size of the next active node and its own congestion window. The flow control is local between pairs of active nodes. 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 (possibly via many normal routers) 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]. The connection is not broken at every node, but only at the active routers for the connection that are expected to be few in number in practical cases.

X. CONNECTION TEAR OFF

At the time of connection tear down, the hosts do not inform the routers as the routers themselves can delete the reservation information when the Reservation Timer for the connection expires.

A. Sender based system

For a sender initiated CLOSE, 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. For a receiver initiated CLOSE, the receiver does a CLOSE as it normally would. The receiver does not have any reservation at the routers.

B. Receiver based system

For a receiver initiated CLOSE, the receiver indicates that it does not have any data to send and closes the connection normally. For a sender initiated CLOSE, by initiating the FIN, the sender indicates that it has no more data to send. The receiver keeps refreshing the reservation at the routers unless it receives the byte sequence numbers till the FIN segment sequence number.

XI. SECURITY

Emergence of active networks presents new security challenges due to the delegation of data processing to the routers. Cryptographic techniques in use in conventional networks are inadequate in providing security for full-fledged deployment of active network features. Thus, mechanisms for protection against hacking, code and data misuse, malicious routing and processing are required. In this work, we make recommendations about the procedures that could be used in addition to other security safeguards for use in IPP.

A. 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 many ways:

- 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 for 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.

There has to be a mechanism at the routers so that any abnormal use of router resources is identified and curbed. 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 an indefinitely long computation. An upper limit to the size of input queue for a connection ensures that no application can occupy an 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. Processing should be aborted if the code tries to access any part of user or kernel memory outside its allocated space. The third problem is curbed due to a router reservation being a soft state reservation.

B. Host security

To make use of intermediate processing on a byte stream of data, the end hosts supplies a set of scripts to the routers which may pose security threats:

- 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. Scripts should be executables ideally impossible to reverse-engineer. This is so that a malicious router does not modify the script to use it to its advantage. Connection data may be encoded so that only the scripts for the connection may be able to process it.

XII. 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, fault recovery mechanisms are required. Since the active routers also modify the data, and the receiver receives data different from what the sender dispatched, there is scope for loss of data without the knowledge of hosts if one of the nodes along the route crashes. The effect of when one or more of the reserved routers fails is different from when one or more of the end hosts fails and the recovery mechanism has to be devised accordingly.

A. 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.

B. Router failure

In this case, the end hosts have the knowledge of the sequence numbers 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 hosts must reroute the connection. Based on RFC 793, 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 hosts must make sure to use sequence numbers larger than those recently used.

XIII. TYPES OF MESSAGES

The IPP uses the following messages:

- 1) *Reservation Request messages:* They are sent with IPARR option set. This option is set in the SYN or

SYN+ACK segment at the time of connection set up and hence does not require a separate message to carry this information.

- 2) *Reservation Reply Messages*: A host receiving a segment containing the IPARR option (request) set would reply to the message by setting the IPARR (reply) option of the next outgoing ACK segment. These messages confirm whether the reservations of routers requested by the other host were successful or not.
- 3) *Reservation Refresh messages*: They are used to maintain the soft state reservation at the routers. Out of the two hosts, the host requiring intermediate processing sends out these messages. These messages may be sent with zero data and IPAPR option set.
- 4) *Non-existent Reservation Error Messages*: These messages are generated by an active 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 host that is responsible for the computation. If this host had done an active opening of the connection (like connect()), it may try to reconnect. If this host had passively accepted the connection (like accept()), it may reset the connection and wait for new connection requests. A host should ignore any error messages for a non-existent connection.
- 5) *Processing Error messages*: These messages are generated at an active router and sent to the sender of the data on which processing failed.

XIV. EXPERIMENTAL IMPLEMENTATION OF IPP

An experimental implementation of IPP was done on Linux 2.2.17 to demonstrate the working of IPP highlighting the concepts of connection set up, router reservation, TIP, IPARR and IPAPR options, data transfer, intermediate processing and reliable communication. A set up of four Linux machines was used. A machine located at University of Washington, Seattle, acts as a remote server for the two client machines located at Iowa State University, Ames. Another machine acts as a router between the clients and the server. All these machines have IPP enabled on them. There are other (non-active) routers on the route between the server and the clients. The set up was used to demonstrate communication using network packet analyzer tools "tcpdump" and "Ethereal Network Analyzer". Preliminary tests were done for two factors:

- *Server availability*: Delegation of computation across routers is expected to raise the availability of a server host. This way, it may be able to service more requests in a given period of time.
- *Client Latency*: A client may incur a delay during connection set up due to code transfer and router reservation. This delay should be within acceptable limits and is expected to be so because code transfer overhead would be a one-time delay for a connection.

For both the factors the results were promising and development work is being pursued to enhance the IPP to a more robust state where precise performance figures may be established.

XV. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a novel approach to computing within IP routed networks to seamlessly integrate two major paradigms - Active Networks and Internet protocols. IPP defines a simpler and more feasible approach to active networking without requiring drastic changes to the way Internet already works. The advantage of such an approach is the achievement of computing power and flexibility in the Internet with active and non-active nodes in perfect co-existence. This field of research is relatively new and there are important issues such as security and router resource management that need to be resolved before such a system becomes available for commercial use. Further research will investigate the expanded security requirements of IPP, identify the constraints particular to IPP and develop new or extended security and resource control techniques needed to meet the unique requirements and constraints of IPP.

REFERENCES

- [1] J. Postel. Internet Protocol, RFC 791, Information Sciences Institute, University of Southern California, September 1981.
- [2] J. Postel. Transmission Control Protocol, RFC 793, Information Sciences Institute, University of Southern California, September 1981.
- [3] J. Postel. Internet Control Message Protocol, RFC 792, September 1981.
- [4] J. Baker. Requirements for IP version 4 Routers, RFC 1812, Cisco Systems, June 1995.
- [5] M. Allman, V. Paxson, W. Stevens. TCP Congestion Control, RFC 2581, April 1999.
- [6] Dave Wetherall and David Tennenhouse. Towards an Active Network Architecture, Computer Communication Review, 1996.
- [7] Dave Wetherall and David Tennenhouse. Active IP option, Proceedings of the 7th ACM SIGOPS European Workshop, Connemara, Ireland, September 1996
- [8] S. da Silva, D. Florissi and Y. Yemini. Composing Active Services in NetScript, ARPA Active Networks Workshop, Tucson, Arizona, March 9-10, 1998.
- [9] Tobias Oetiker. MRTG - The Multi Router Traffic Grapher, LISA 98 Systems Administration Conference, Boston, Massachusetts, December 6-11, 1998.
- [10] 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.
- [11] 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.
- [12] S. Bhattacharjee, K. Calvert and E. Zegura. Active Networking and End-to-End Arguments, IEEE Network Magazine, 1998.
- [13] K. Calvert, E. Zegura, J. Sterbenz. CANES: A Modest Approach to Active Networking, Presented at IEEE Computer Communications Workshop, September 1997.
- [14] Ulana Legedza, David J. Wetherall, and John Guttag. Improving The Performance of Distributed Applications Using Active Networks, IEEE INFOCOM, IEEE, March 1998.
- [15] Beverly Schwartz, Alden W. Jackson, W. Timothy Strayer, Wenyi Zhou, Dennis Rockwell and Craig Partridge. Smart Packets for Active Networks, Presented at Openarch, March 1999.
- [16] Micheal Hicks, Pankaj Kakkar, Jonathan T. Moore, Carl A. Gunter and Scott Nettles. PLAN: A Packet Language for Active Networks, Proceedings of the 1998 ACM SIGPLAN International Conference on Functional Programming, pages 86-93, Baltimore, MD, 27-29 September 1998.