T00988

Architectural Framework for Active Networks

Acc. No. (TMS) T-988

Developed By

Adnan Iqbal

Supervised By

Dr . M.Sikandar Hayat Khiyal

Dr. Muhammad Sher

Department of Computer Science

Faculty of Applied Sciences

International Islamic University, Islamabad.

(2004)

**In the name of Almighty Allah,
The most Gracious,
The most Merciful.**

Department of Computer Science
International Islamic University, Islamabad.

Final Approval Date: 22/4/04

It is certified that we have completely read the thesis, titled **"Architectural framework for active networks"** submitted by **Adnan Iqbal** under university registration number 36-CS/MS/01. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic university, Islamabad, for the degree of **Master of Science.**

Committee

External Examiner
Dr. Abdul Sattar [illegible]

Internal Examiner
Dr. Tauseef-ur-Rehman [illegible]
Assistant Professor,
Department of Computer Science,
International Islamic University,
Islamabad.

Supervisors
Dr. M. Sikandar Hayat Khiyal S Hayat K
Head,
Department of Computer Science,
International Islamic University,
Islamabad.

Dr. M. Sher [illegible]
Assistant Professor,
Department of Computer Science,
International Islamic University,
Islamabad.

Dedication

Dedicated to my Parents.

A dissertation submitted to the
Department of Computer Science,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for the award of the degree of
Master of Science

Declaration

I hereby declare that this software, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that I have developed this software entirely on the basis of my personnel efforts made under the sincere guidance of my teachers. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Adnan Iqbal
36-CS/MS/01

Acknowledgments

All praises to Almighty Allah, the most Merciful, the most Gracious, without whose help and blessing, I was unable to complete the project.

Thanks to my parents who helped me during my difficult times. And it is due to their care and love that ii am as this position today.

Thanks to my project supervisors Dr. M. Sikandar Hayat Khiyal and Dr. M. Sher, their sincere efforts helped me to complete my project successfully.

I acknowledge all the persons for their help in this project.

Adnan Iqbal

Project In Brief

Project Title:	Architectural Framework for Active Networks
Objective:	To develop a software based router that could serve as Active Router.
Undertaken By:	Adnan Iqbal 36-CS/MS/01
Supervised By:	Dr. M. Sikandar Hayat Khiyal Head, Department of Computer Science, International Islamic University, Islamabad. Dr. M. Sher Assistant Professor, Department of Computer Science, International Islamic University, Islamabad.
Technologies Used:	Microsoft® Visual C++ 6.0
System Used:	Pentium® IV
Operating System Used:	Microsoft® Windows 2000 Professional
Date Started:	1st December, 2002
Date Completed:	20th August, 2003

Abstract

The networking now a day is not simply the connection between multiple computers and the goal is not only the data transfer from one location to another. Day by day new services are being introduced in the networks. The problem is that these services are not easy to implement. These services require standardization that is a lengthy process. We have tried to find out a way to eliminate this standardization process. This can be done by using active approaches to networking rather than classical passive approach. In this model we have emphasized that the computations performed by the intermediate nodes are not standard but the standard is computing environment where multiple types of computations can take place. As a proof of concept, we have developed software simulating active routers and hosts. It is successfully tested and it provides option to create different type of active applications and test them.

Introduction

1. Introduction

1.1 Active Networks

The term active networks is considered as relatively new term but it is not as new as it is considered but as a matter of fact it is being given reasonable consideration now a days. The concept of active networks emerged from DARPA[5] meetings. The term active network means that a network has the capability of executing code inside intermediate nodes. Current networks are essentially passive networks as intermediate nodes are not capable of executing coed being brought up by the data packets going through them.

The philosophical difference is that current networks have standardized the computations performed on the packets data and active networks do not standardize these computations instead the intent is to standardize the execution environment.

Active networks are packet-switched networks in which packets can contain code fragment that are executed on the intermediary nodes. The code carried by a packet may extend and modify the network infrastructure. The goal of active network research is to develop mechanisms to increase the flexibility and customizability of the network and to accelerate the pace at which network software is deployed. Applications running on end systems are allowed to inject code into the network to change the network's behavior to their favor.

Most networks currently have a topology where 'smart' hosts sit at the edges of the network, and are connected by dumb switches.

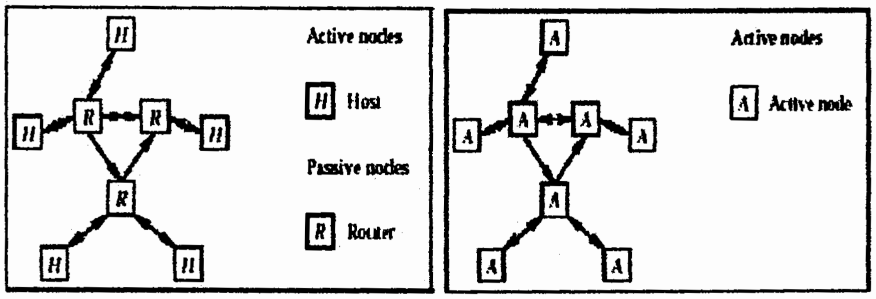


Figure 1.1 (a) Current Networks (b) Active Networks

In reality, these switches are not so dumb: they are computers just as powerful as the hosts, but this computing power is mostly used to push packets around. The switch managers decide what software should run on the nodes, so adding new features to the network requires the agreement of all the switch manufacturers and managers.

An alternative model is to allow users access to the computing power in the switches, and to run their own software. The network has some dumb switches and some smart switches, and we can run a virtual network to hide the dumb switches from the user.

1.2 Approaches in active networks

Until recently, active networking research concentrated on two distinct approaches: "programmable switches" and "capsules". These two approaches can be viewed as the two extremes in terms of how program code is injected into network nodes. Programmable switches typically upgrade by implicit injection of code by a network administrator. Research in the area of programmable switches focuses on how to upgrade network devices at run time, on upgrades introduced by administrators which support end system applications (e.g. congestion control for real-time data streams), or on a combination of both.

Capsules, on the other hand, are packets carrying small amounts of program code, which is transported, in-band and executed on every node along a packet's path. This approach introduces a totally new paradigm to packet switched networks. Instead of "passively" forwarding data packets, routers execute the packet's code. The result of that computation determines what happens next to the packet. Applications include simple proof-of-concept ping applications, network diagnostic tools, active multicasting and more. This approach has the potential for an enormous impact on the future of networking. However, in the near future, security constraints will cause severe performance problems for capsule-based solutions. Capsules commonly make use of a virtual machine that interprets the capsule's code to safely execute it on a node.

Recently, convergence between the pure "programmable switch" and the pure "capsule" approach became visible. Most of the research groups involved agree that some sort of code caching makes a lot of sense. The main motivation for this convergence is the realization that potential capsule code is more application specific than user specific. In the same way, users usually do not write their own applications but use off-the-shelf software. They are not expected to inject their own programs into the network, but use code from a set of code modules written by specialists. This allows for various optimizations in the form of caching. We will also show how our approach aggressively builds upon this same realization.

1.3 Applications

Active networking is relatively new concept and its importance is not well understood. Therefore its applications are currently very limited although research is being done rigorously to find out the areas where active networks can be applied successfully.

Different areas are identified where active networks can be used for example, Network management, Web caching, Congestion control etc. the details of all these applications are described in "Applications of Active Networks".

Example applications include self-learning web caches, congestion control, on-line auctions, and sensor data mixing. Since the code is injected out-of-band, programmable switches provide no automated, on-the-fly upgrading functionality.

Another very important observation is that the deployment of multimedia data sources and applications (e.g. real-time audio/video, IP telephony) will produce longer lived packet streams (flows) with more packets per session than is common in today's Internet. Especially for these kinds of applications, active networking offers very promising possibilities: media gateways; data fusion and merging; and sophisticated application specific congestion control. Both our hardware and software architectures support the notion of flows. In particular, the locality properties of flows are effectively exploited to provide for a highly efficient data path.

Active Networks can also be used to enhance security and to cope with absolutely unseen and unpredictable events. Bernard Cole has described the need of Active Networks in the scenario of unforeseen events very well.

Currently, network management is achieved by using a polling mechanism. In this method management stations routinely poll the managed devices for data, looking for anomalies. This technique has served us well in the past. However, due to the increase in the number and complexity of nodes in the network, now it has become problematic.

Management centers become points of implosion, busy with large amount of information. This information is very often redundant, as the packets that arrive may simply report that there was no change in the state of the monitored part of the network. Also, in case of a problem, the round-trip delay that is needed for the information to reach the management center and the reply to return back to the affected part of the network is sometimes significant and the action undertaken is not up to date any more. It is essential that network management employs techniques with more immediate access and better ability to scale.

Active networks are the natural answer to the above problem. By making the internal nodes of the network active we can move the management centers right in the "heart" of the network and thus reduce both delays from responses and bandwidth utilization for management purposes. Also, we can inject special code in the packets that can act as "first aid" in case they encounter a problematic node. This code can be executed in the affected node and change its state automatically instead of waiting for a reply from a management center. Other packets can act as patrols, constantly looking for anomalies as they trace the network. Finally, since a management center sends programs to the managed nodes, it can request real-time tailoring of the information to be returned in order to meet its current needs. This will reduce the back traffic and processing time of the information after it is received by the management center. To sum up, by using active networking for network management:

- Problems are tracked quickly or are reported automatically without the need of polling.
- Management centers can be in the "heart" of the network, thus delays from responses and bandwidth utilization for management purposes are reduced.
- "Patrol" and "first aid" active packets can respectively track a problem and deal with it at once.
- Information content returned to the management centers can be tailored to the current interests of the center so that back traffic and processing time are reduced.

- Management policies can change easily as administrative requirements change, thanks to the inherent flexibility of active networks technology.

Packet switching networks are network of queues. At each node, there is a queue of packets for each outgoing channel. If the rate at which packets arrive and queue up exceeds the rate at which packets can be transmitted, the queue size grows without bound and delay experienced by a packet goes to infinity. All this process is called Congestion.

This problem is unlikely to disappear in the near future. Therefore, it is essential to find efficient algorithms to deal with it. Congestion is a prime candidate for active networking. Also, it often takes a considerably long time for congestion notification information to propagate from the point of congestion to the user, so that the latter can self-regulate in order to reduce congestion. As a result, either there is a period of time during which congestion is augmented -- since applications have not learned about it -- or the notification arrives so late that there is no longer any congestion and self-regulation is not needed. On a descriptive level, one can find many examples where the added functionality of active networks can help in dealing with congestion control. Here are some examples:

- An active node can monitor the available bandwidth and control the rate of a data flow accordingly. Of course, buffering is needed in this case, so instead of putting the buffers in the switch, we can put them in the active node.
- In case of many data flows with different congestion requirements, an active node can control the relevant rate of each flow in addition to the total rate. Also, it is possible to adapt to dynamic changes of the requirements.
- The transformation of data at a congestion point is also a powerful capability. In fact, applications sometimes produce data according to the congestion situation if they are aware of it. Therefore, we can perform the above transformation right in the place where it is needed and only if it improves the performance. However, we should expect that from a computational point of view, a transformation may have a significant cost.
- Selective dropping of units, packets or cells can be held very efficiently. In case of congestion, we prefer to drop less important units than more important ones.

The importance of a unit depends on the amount of information it carries. A classic example here is the case of MPEG compressed video where if we lost an I frame; there is no point in keeping the P and B frames that depend on the lost I frame.

- Finally, we can have a multi-stream interaction in the following sense: e.g., if a user is receiving video and audio and there is a loss in the video, audio units should receive extra priority to assure that the user will still get some information.

A substantial fraction of network traffic in the Internet comes from applications like the World Wide Web, where information is retrieved by clients from servers located anywhere in the network. The caching of objects at locations close to the clients can decrease both the network traffic and the time needed to retrieve the information. Active networks can be used to provide a smart caching scheme wherein smaller overall storage capacity is needed and higher reduction in network traffic and latency can be achieved. Traditional approaches to network caching are to place large caches at specific points in the network. The key point in these schemes is how to choose these specific points. One option is to cache at transit nodes (transit-only caching). Since a large fraction of paths in the network have to go through transit nodes, they are prime candidates for caching. Another policy is to cache in stub nodes that are connected to transit nodes because the former have to be traversed in order for a node inside a stub domain to access the rest of the network. Therefore, cache nodes can be located near the edge of the network or at strategic points within the network organized with a hierarchical scheme wherein clients are manually configured to access a particular cache in the hierarchy.

An interesting idea would be to balance the hierarchy by repositioning not only the cached information but also the cache nodes. In this scheme, each node or a set of nodes decide whether to cache the information that returns from the server to the client. Obviously, the effective organization of the location and the content of the caches are not trivial. Nodes should be smart enough to cache objects that nearby clients will request in the future and to coordinate with each other to avoid caching the objects that are already cached in neighbor nodes. Active networks technology may help deploy a mechanism of coordinating the nodes. Also, because a significant fraction of Web pages are dynamically computed, active technology may support the storage and execution of programs that generate these pages in nodes near the clients. Recent work at the Georgia Institute of Technology considers the benefits of associating caches with nodes throughout the network, and self-organizing cache contents in an active way. The proposed scheme, called Self-Organizing Wide-Area Network Caches, yields round-trip latencies that are smaller than or equal to the more traditional approaches, while requiring much smaller caches per node. The basic idea is to obviate the need to decide where to place caches by considering that all nodes of the network can cache objects and relying on active technology to maintain a uniform distribution of caches within the network. Nodes make local decisions in a way that resources are used effectively overall. The first approach described is called modulo caching. A distance measure, called cache radius, is defined, measured in transmission hops. The caching policy uses the radius as follows: on the path from the server to the requesting client, information is cached in nodes that are cache radius apart. We therefore end up with a distribution of caches located a "cache radius" away from each other. The second approach uses some of the cache space in each node to store locations of information objects. Each node's cache is divided into levels. Level 0 contains locally cached objects; level 1 contains objects cached in nodes one hop away, etc. When a request message for an object is processed, the levels are searched in sequence beginning with level 0. This approach is called look around algorithm. The number of levels of adjacent caches maintained and checked in this algorithm is a parameter of the policy and, as with the cache radius, might be set globally, on a per-object basis, or even locally.

Simulation results show that active mechanisms outperform traditional methods in case of correlated accesses. By correlated accesses, we mean that an initial access will cause future accesses involving the same client and server pair. In case of uncorrelated accesses, transit-only caching performs a little better than active mechanisms, but this sort of caching fails to adapt to correlated accesses

1.4 Related Work

Active networking research has been ongoing for several years. Various research labs have described and implemented interesting approaches. In this section, we give an overview of some of these efforts. Related work is going on in many universities of the world. Following universities are doing reasonably useful and prominent research in the field of Active Networks.

MIT, BBN, Georgia Tech, University of Pennsylvania, University of Arizona and Columbia University are some of the universities carrying out research about active networks and most of the research about active networks is funded by DARPA.

This work is in different dimensions. Both the approaches are being researched i.e. programmable switches and capsule based approach. Different supporting technologies are being developed like operating systems supporting Active Networks and compilers supporting Active Networks.

1.5 Our Work

Our work is a step by step procedure. The objective to start this project is to create a culture of Active Networks at International Islamic University. Of course we started with zero. The steps followed or will be followed to fulfill main objective are given below.

1- To study the basics and underlying philosophy of Active Networks.
2- To study current work being carried out about Active Networks at international level.
3- To study the applicability of Active Networks.
4- To propose a practically usable Active Network.
5- To implement proposed Active Network prototype.
6- To study, compare and analyze different thoughts and propose new thoughts.
7- Enhance that prototype in a step by step procedure and to introduce more features in it.
8- Use developed prototype to provide different services, using it as a building block.

All these tasks are to be done in a step by step procedure; we have decided to complete first four tasks in this particular project and to keep on building on these steps in the next coming projects.

Analysis

2. Analysis

2.1 Important questions

When we start analysis then we come to know that in case of an Active Network, manual system is the network infrastructure available commonly and we can name that network infrastructure as passive networks at the start of analysis we face different questions and answers to these questions are the key to clear and fruitful analysis phase. Some of the possible questions are given as.

- **What are the differences between Passive Networks and Active Networks?**
- **What are the ingredients of an Active Network?**
- **What is the purpose of building an Active Network?**
- **In what scenario it will be used?**
- **Should we develop an Active Network absolutely orthogonal to current networks?**
- **How can we build an Active Network?**
- **What tools should be used?**

Our analysis phase is based on these questions and we tried to find out clear, precise and well defined answers to these questions. This document is based on these questions and their answers and the design of the software depends upon the answers given to these questions. Next section of this chapter discusses these questions in detail and provides answers to these questions.

2.2 Answers

2.2.1 What are the differences between Passive Networks and Active Networks?

A passive network is a network in which data packets pass through intermediate nodes without being altered or even without being examined. For example an IP network has different devices such as personnel computers, routers, switches etc. personnel computers are at terminals and switches, hubs and routers are intermediate devices. Whenever a packet is generated from one terminal computer for another terminal computer, this packet has to go through different intermediate nodes like routers and switches. These routers, switches or hubs can perform different operations on this packet for example a hub does not perform any operation and just passes the packet opaquely to all connected nodes other then the sender. A switch finds out destination requested by sender and just passes the packet towards destination. A router does a similar job. It finds requested destination and forwards the packet towards destination. Some devices perform more computations and the can restrict the forwarding of packet to other destination. Even then the scope of computations is very limited. Number of computations that an intermediate device supports and is allowed to do are very limited and all these computations are based on the header of packet and none of the computation is performed on the data of packet itself.

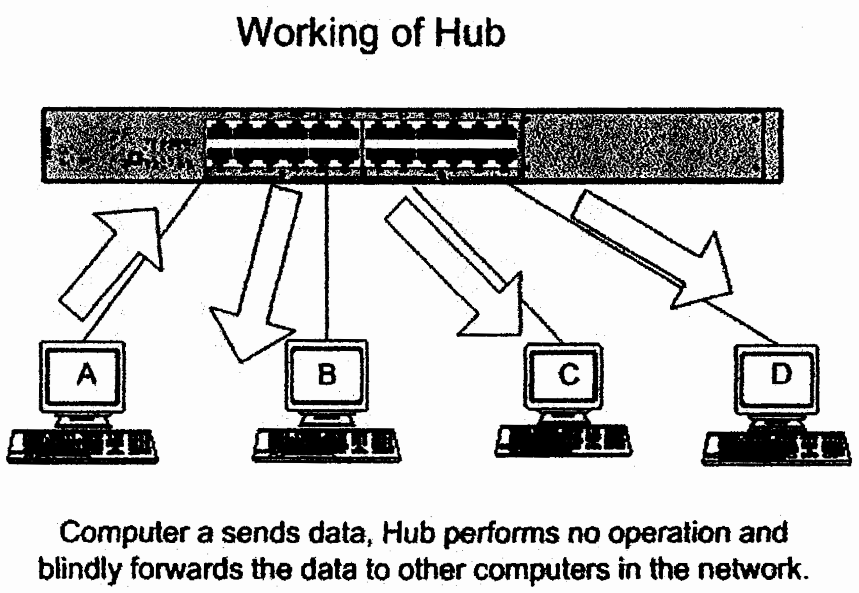
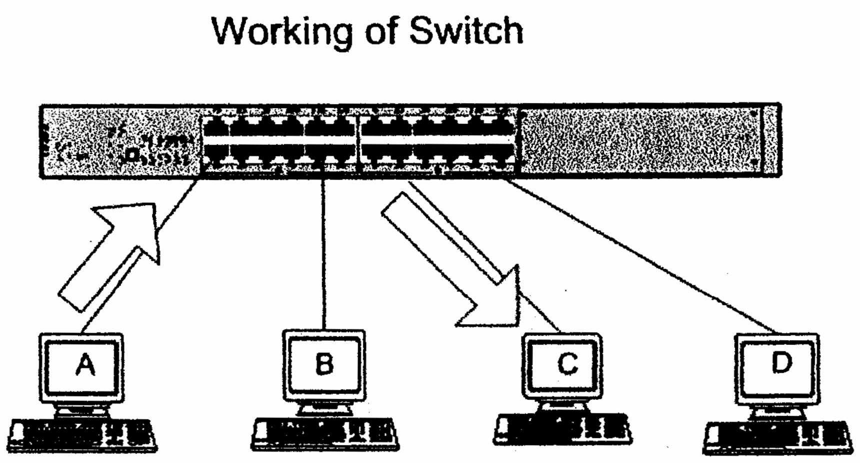

Working of Hub
A
B
C
D
Computer a sends data, Hub performs no operation and blindly forwards the data to other computers in the network.

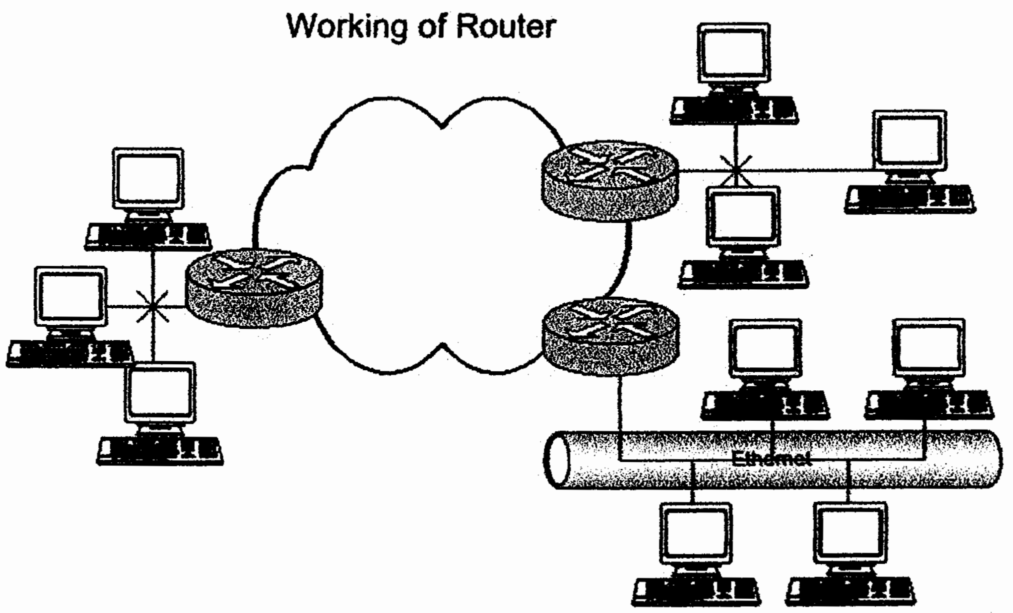
Figure 2.1 A Hub broadcasts data to other devices



Computer a sends data, Switch performs small operation on the header of data, finds out destinatination and forwards the data to requested computer.

Figure 2.2 A switch unicasts data to intended destination.

Active Networks follow a totally different approach. In a passive network, number and type of computations are standardized and none of the vendor can deviate from these computations. In an Active Network neither number of computation and nor the type of computations are standardized instead it is being tried to standardize the computing environment.



Routers perform more complicated tasks. Routers receive a packet, find its required destination and search a suitable path and then forward the packet towards that suitable path.

Fig 2.3 Routers are internetworking devices.

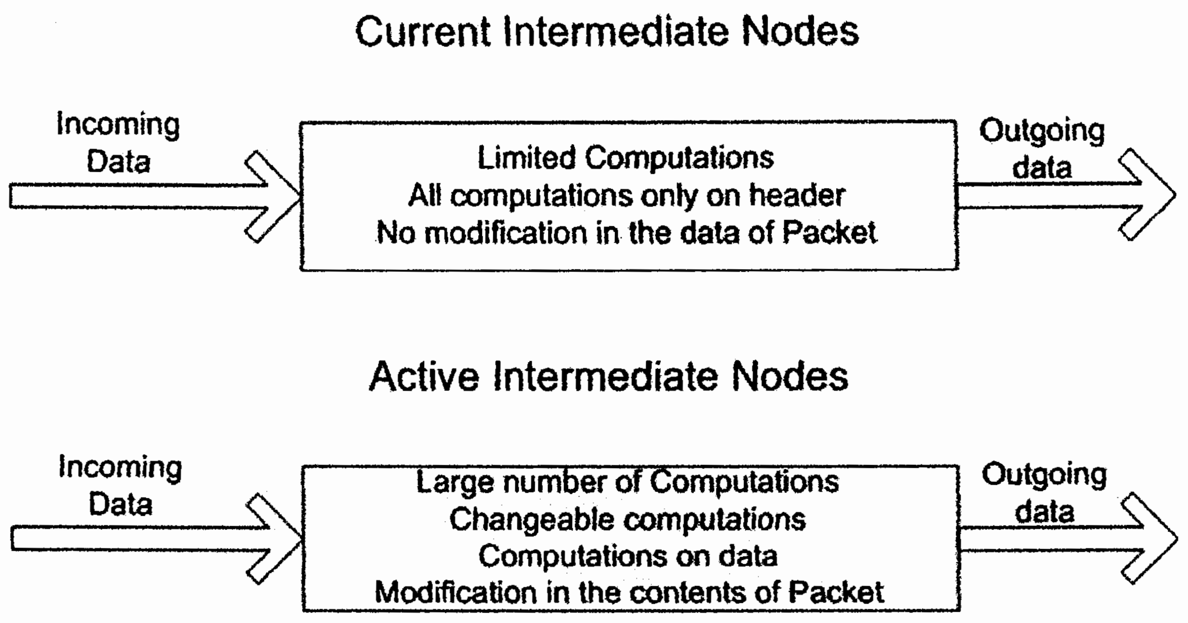

Current Intermediate Nodes
Incoming Data
Limited Computations
All computations only on header
No modification in the data of Packet
Outgoing data
Active Intermediate Nodes
Incoming Data
Large number of Computations
Changeable computations
Computations on data
Modification in the contents of Packet
Outgoing data

Fig 2.4 Difference between Active and Passive routers

In an Active Network computations performed by an intermediate node can be user specific, application specific or the vendor specific as well. Intermediate devices allow the packets that go through them to have executable code or some kind of mechanism that could initiate some action from the intermediate devices. So the packets in active network do not only have passive data, they can also have executable code as well and that code can be executed by the intermediate devices.

2.2.2 What are the ingredients an Active Network?

A passive network consists of terminal devices and intermediate devices. Similarly an active network consists of terminal and intermediate devices. Terminal devices can be the personnel computers or any device that can send data end to end. Intermediate devices are not the end points of the network. These devices lie in the middle of the network and a packet has to go through at least one of them to reach the destination.

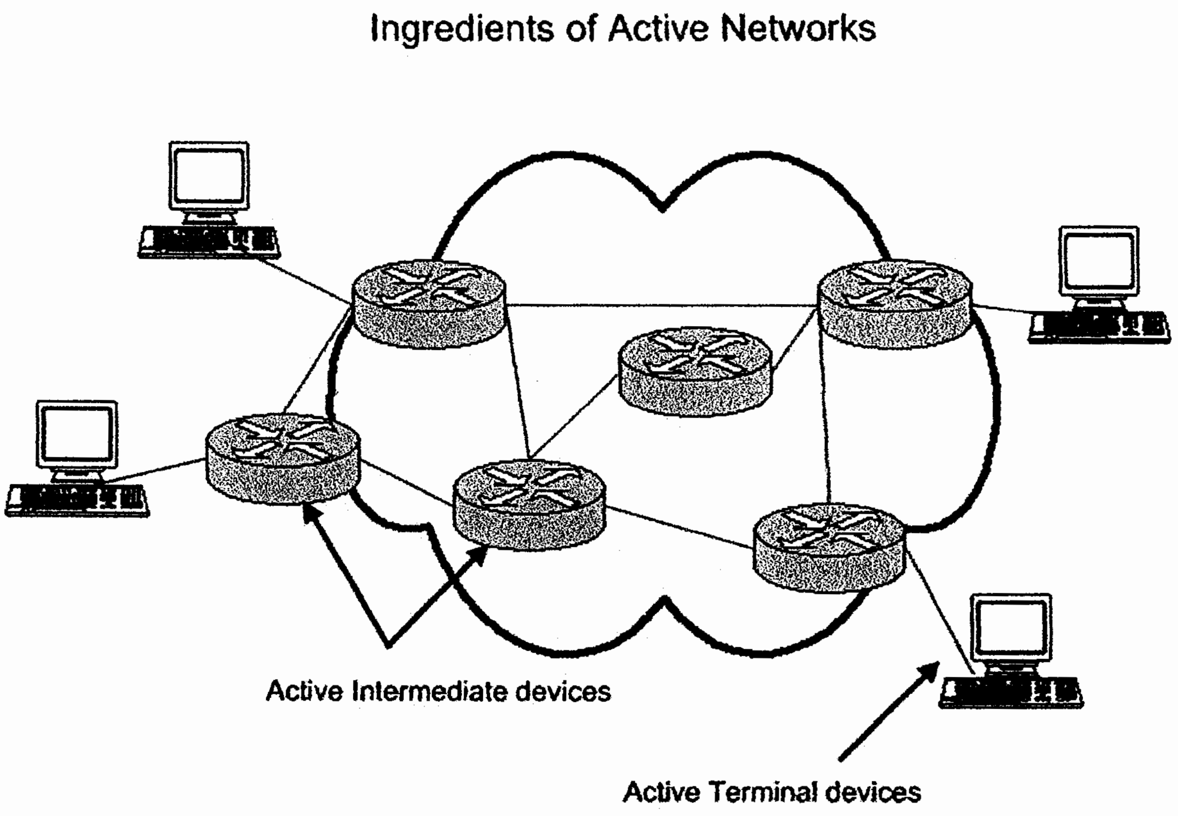


Fig 2.5 Active intermediate devices and terminals make an Active Network

The most important thing is that the terminal devices must be capable of creating packets that contain executable code and intermediate devices must provide the facility of executing that code when packets pass through them.

2.2.3 What is the purpose of building an Active Network?

There can be many reasons to develop an active network. it can be developed for network management, caching, congestion control, multicasting and many other things. Our reason to build an active network is to provide a platform for next coming researchers so that they may not start from zero instead they take up the work done and build on it.

It is basically being developed as a proof of concept system. Our attempt is to create a simplest possible architectural framework for active network and then to enhance it step by step. Very small work is done in this field and especially in Pakistan almost no work is done before. Most of the work done is in USA, Japan and Canada.

The Active Network created will be able to support newer services required by the user of the network and as a matter of fact user will not require to standardize the services instead he will have to create only the functionality required and incorporate that functionality inside intermediate nodes.

2.2.4 In what scenario it will be used?

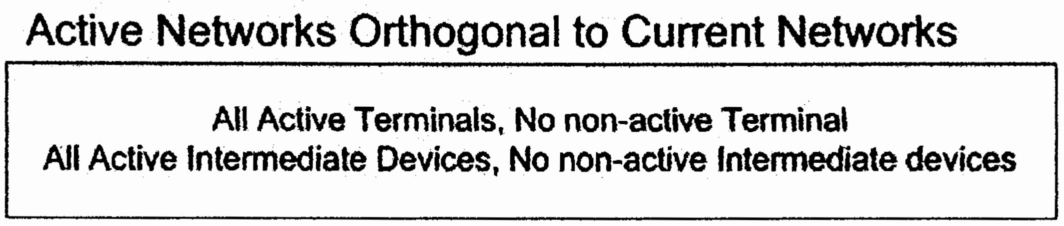
As it is discussed earlier that active networks can be used in different scenarios. Active Networks can solve many problems and many active applications can be written to take benefits from active networks.

It is not in the scope of this project to write any of such applications as the target of this particular project is to make an active network infrastructure that can be used to develop such applications.

Even then we will try to make at least one application that will elaborate a scenario where it can be used. We will write an application that will be used to manage the network by obtaining current network information in the form of routing tables.

2.2.5 Should we develop an Active Network absolutely orthogonal to current networks?

Answer to this question is not a straight forward one. It is a lot easier to develop an active network absolutely orthogonal to current networks. On the other hands such kind of network can only be used for research purpose it is far from expectation that such a network will ever replace the current networks because current passive networks have spread over the world and it is absolutely impossible to shift them to absolutely new paradigm.

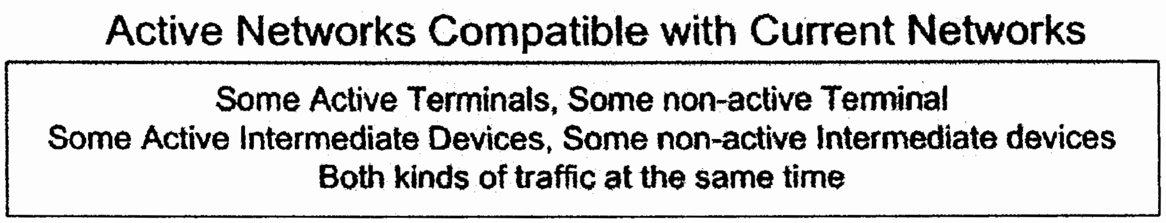


Active Networks Orthogonal to Current Networks

All Active Terminals, No non-active Terminal
All Active Intermediate Devices, No non-active Intermediate devices

Fig 2.6 Orthogonal Active Networks consist of active devices only.

A better approach can be to use existing network infrastructure as a building block and to provide the functionality of active networks. In this case newly born network will exhibit the properties of both the networks passive as well as active. The advantage is obvious; we can shift to such kind of networks without lot of major changes. it will allow the use of existing networks and there will be no or little impact on current applications and services. As the active networks grow in demand and become better and better in their functionality, existing network will keep on changing their shape into active networks. This is slow but an acceptable approach and because of this flexibility we have adapted to use this second approach.


Active Networks Compatible with Current Networks

Some Active Terminals, Some non-active Terminal
Some Active Intermediate Devices, Some non-active Intermediate devices
Both kinds of traffic at the same time

Fig 2.7 Active Networks can also work with current networks

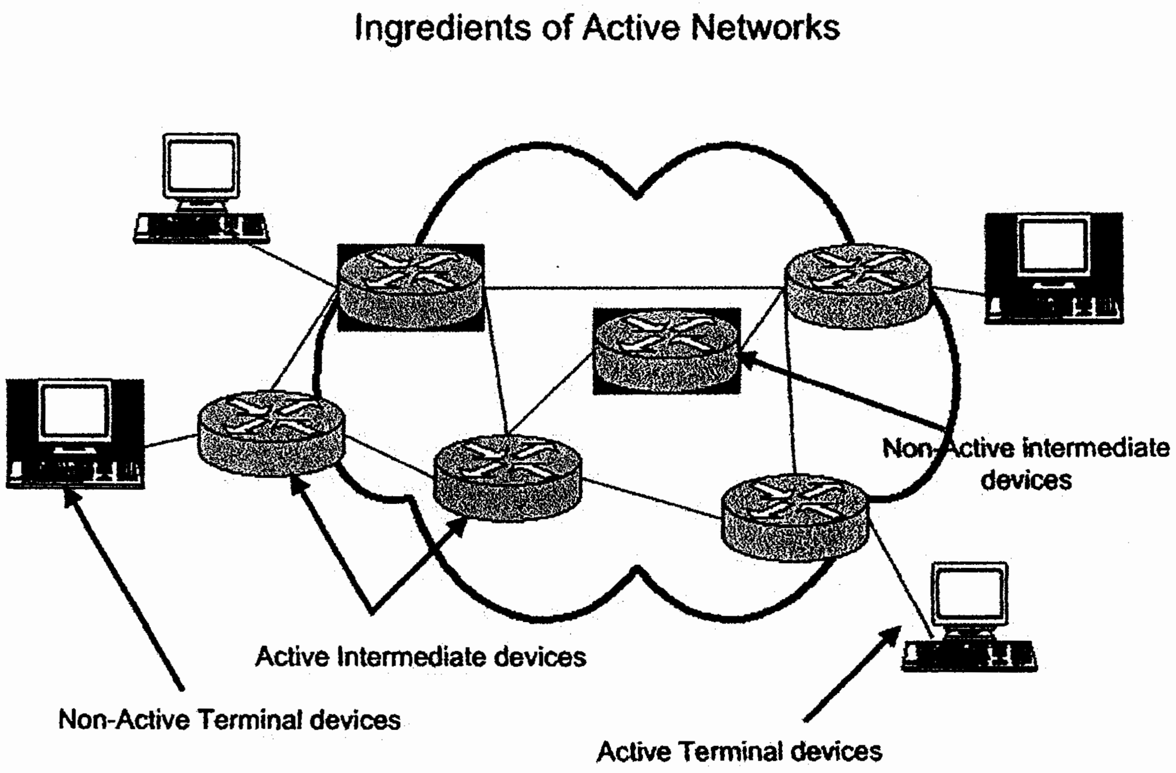

Ingredients of Active Networks
Non-Active intermediate devices
Active Intermediate devices
Non-Active Terminal devices
Active Terminal devices

Fig 2.8 A non orthogonal Active Network

2.2.6 How can we build an Active Network?

Answer to this question demands us to study the requirements for the active networks in detail. If we want to develop an active network then there can be multiple ways and requirements. Some requirements and methods are consistent with each other and others totally opposite and cannot be used with each other at all. Let us define these requirements and the methods.

The most important part of an active network is the intermediate device. This intermediate device can easily be called as the heart of active network. This intermediate device is most probably a router but it can be a switch as well. These routers are called active router because these provide the functionality of code execution and computation of different kinds.

Another part of the active network is active terminal. This is an end device or terminal computer that has the capability of creating packets that are in accordance with active router and contain executable code along with data.

2.2.7 What tools should be used?

Different tools can be used for the development of an active network. The most appropriate one is the C++. The platform used is Microsoft Windows 2000. The compiler used is Microsoft Visual C++.

2.3 Analysis Methodology

The methodology used to analyze active networks architecture is object oriented analysis technique. Object oriented analysis offer many advantages over structured analysis technique therefore it is proffered over structured analysis methods.

2.2.1 Use Cases

Many use cases were found during the course of analysis. Description of all the use cases is given below.

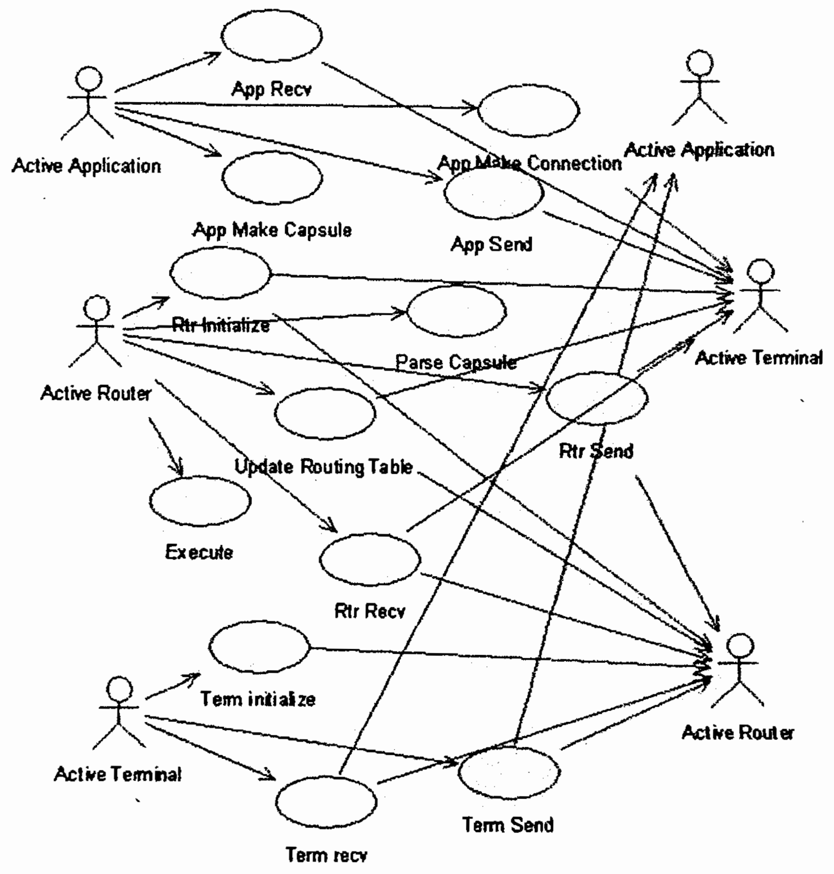


Fig 2.9 Use Case Diagram

2.2.2 Use Case Description

2.2.2.1 Terminal Initialize

This use case describes the working that how a terminal is initialized. First of all a terminal finds out the port, from where it can get data or send data to an application. It then finds out the interface with the network link. It initializes the buffers. It sends its information on the network and starts the receiver to continue working.

2.2.2.2 Terminal Send

This use case is used when the terminal wants to send a capsule to the network or to the local application. To send the data first it picks a capsule from buffer and finds out the requested destination and then sends data to requested destination.

2.2.2.3 Terminal Receive

This use case describes the working that how a terminal continues to receive data. It waits for data to arrive from any of the interface, when data arrives; it copies that data to an appropriate buffer.

2.2.2.4 Router Initialize

This use case describes the initialization phase of a router. First the router finds number of interfaces available to it. It initializes all the buffers available to it. It sends its information to all of the requested links and it also initializes the routing tables. It initiates receiver and controller.

2.2.2.5 Router Send

This use case describes the procedure of sending data from router to the routers and terminals associated to it. It is invoked by a controller or by processor. In either cases it is given with the capsule and destination. The sender finds out the path to the destination by using routing tables and forwards the data as found.

2.2.2.6 Router Receive

It is responsible for receiving all kind of data. Data can arrive either from routers or from any terminal. It is initialized by the initializer and waits for data. When it receives data it copies that data into appropriate buffer.

2.2.2.7 Router Update Routing Table

This use case describes the procedure of updating the table when a change occurs in the neighboring routers. Neighboring devices can send a specific message to this router. That message is received by receiver and store din the buffer. Controller picks the message, gives it to updating module and it updates the routing table. It also sends new routing entries to its neighboring devices.

2.2.2.8 Router Parse Capsule

This use case describes the process by which a received capsule is converted into a meaningful thing for a router. It gets data from controller and divides the data into different meaningful parts by separating data and executable code.

2.2.2.9 Router Execute Capsule

This capsule describes the execution of a capsule. The parsed capsule has two parts data and code. These parts are taken by the execution manager and executable part is divided into function and these functions are executed. After execution the capsule is reassembled and sent to the required destination.

2.2.2.10 Application Make Connection

Every application that wants to use activeness needs to make a connection with the terminal. It cannot send or receive directly from the network. It makes a connection from the terminal by sending request and terminal acknowledges the request.

2.2.2.11 Application Make Capsule

Every application is responsible to send data in proper capsule format. It is not the responsibility of the network to convert the data into proper capsule format therefore every application should have a procedure to make data into proper capsule format and then transfer it to the terminal.

2.2.2.12 Application Send

After establishing a connection with the terminal the application can send capsules to the network using terminal. For this purpose only requirement is that the capsule should be proper.

2.2.2.13 Application Receive

Every application can receive data from the terminal and for ths purpose they must have a receiving module.

2.2.3 Activity Diagrams

2.2.3.1 Activity diagram of Initialize Terminal

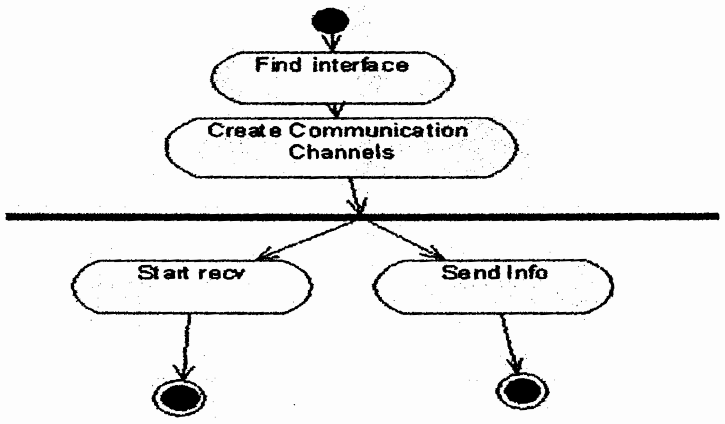


Fig 2.10 Initialize Terminal

2.2.3.2 Activity diagram of Terminal Send Data

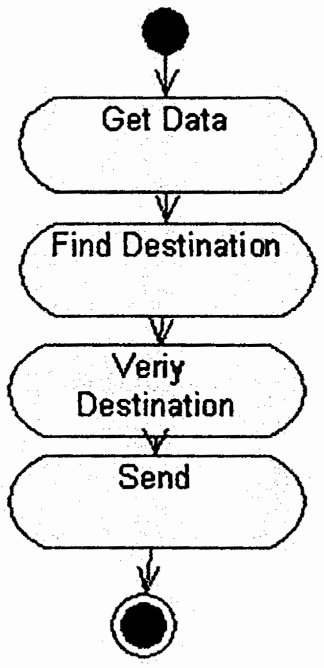


Fig 2.11 Terminal Send Data

2.2.3.3 Activity diagram of Terminal Receive Data

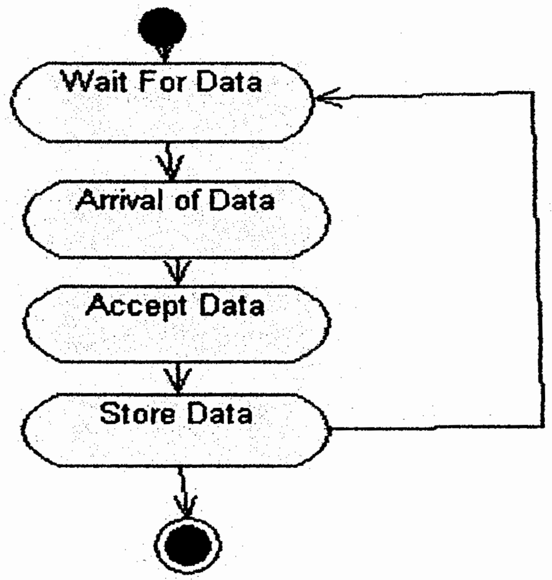


Fig 2.12 Terminal Receive Data

2.2.3.4 Activity diagram of Update Routing Table

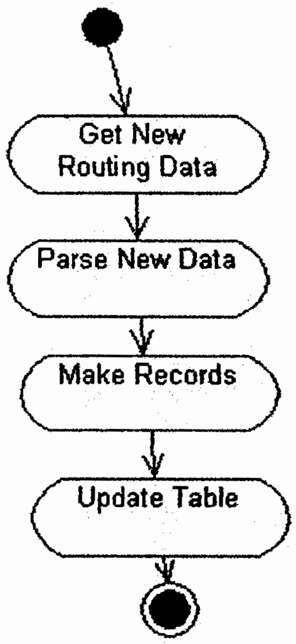


Fig 2.13 Update Routing Table

2.2.3.5 Activity diagram of Execute Capsule

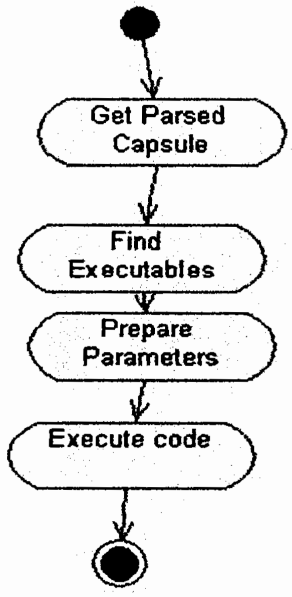


Fig 2.14 Execute Capsule

2.2.3.6 Activity diagram of Make Capsule

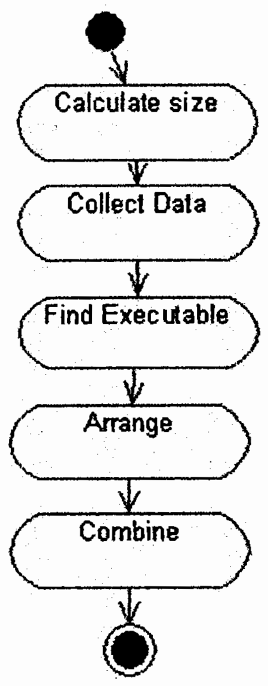


Fig 2.15 Make Capsule

2.2.3.7 Activity diagram of Application Connection

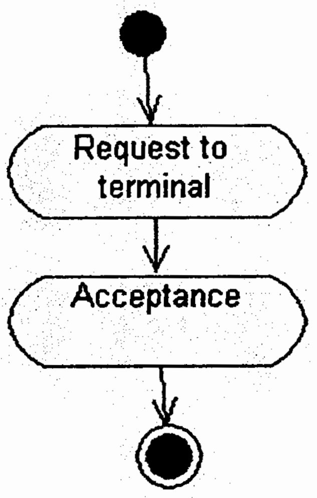


Fig 2.16 Application Connection

2.2.3.8 Activity diagram of Parse Capsule

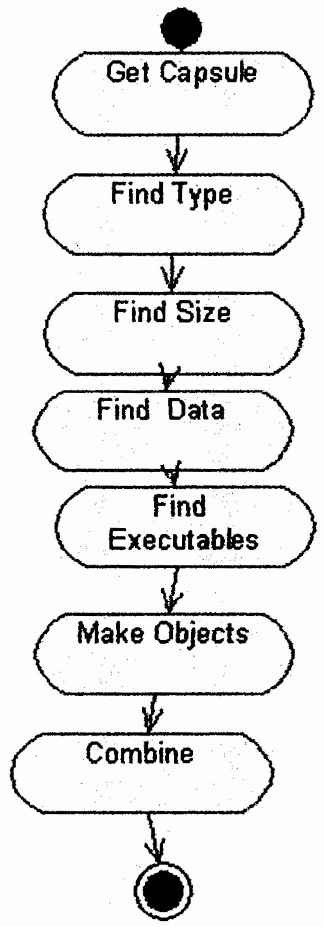
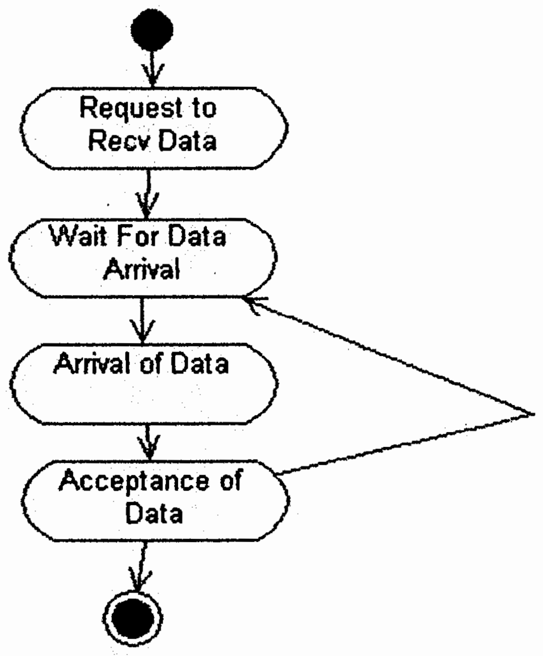


Fig 2.17 Parse Capsule

2.2.3.9 Activity diagram of Application Receive Data



T-988

Fig 2.14 Application Receive Data

2.2.3.10 Activity diagram of Application Send Data

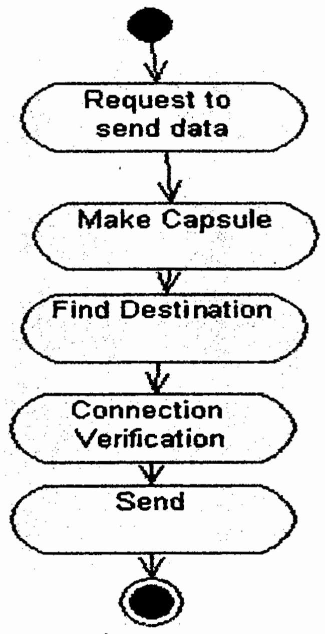


Fig 2.18 Application send Data

2.2.3.11 Activity diagram of Initialize Router

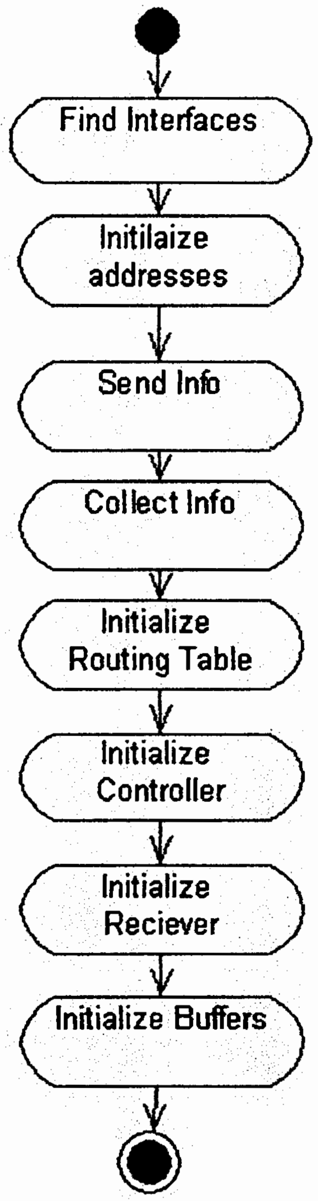


Fig 2.19 Initialize Router

2.2.3.12 Activity diagram of Router Receive Data

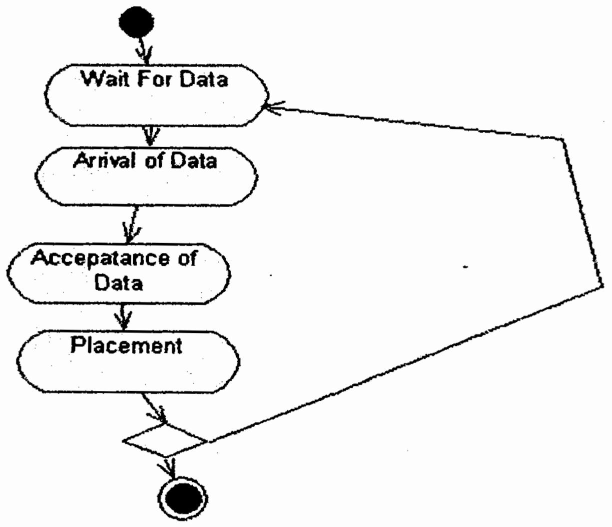


Fig 2.20 Router Receive Data

2.2.3.1 Activity diagram of Router Send Data

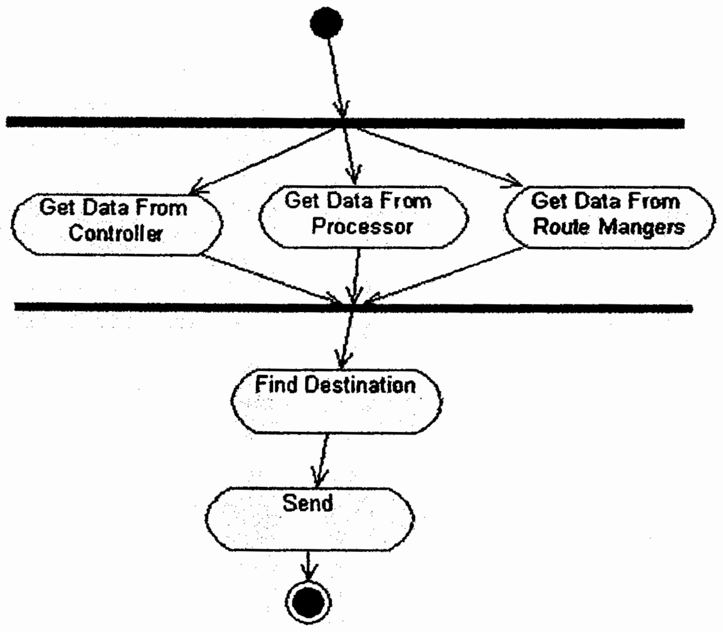


Fig 2.22 Router Send Data

Design

3. Design

The importance of design is very clear to every one. The important question is that what design methodology to be used. We have decided to use Object Oriented Design because of many reasons. The most important one is that our work is of the nature that it has to be extended by the successors and for this purpose design has to be extremely extensible and this property is well satisfied if we use object oriented design. Moreover the prototype developed by us has to be used by the people as a building block. Therefore our application has to be highly reusable. This property is also well satisfied by object orientation.

Next sections of this chapter describe different object oriented tools used to define our design. These include core concepts, classes, description of classes and sequence diagrams.

3.1 Classes

Classes are basis for object orientation. We have defined classes that can be used in our design by finding concepts.

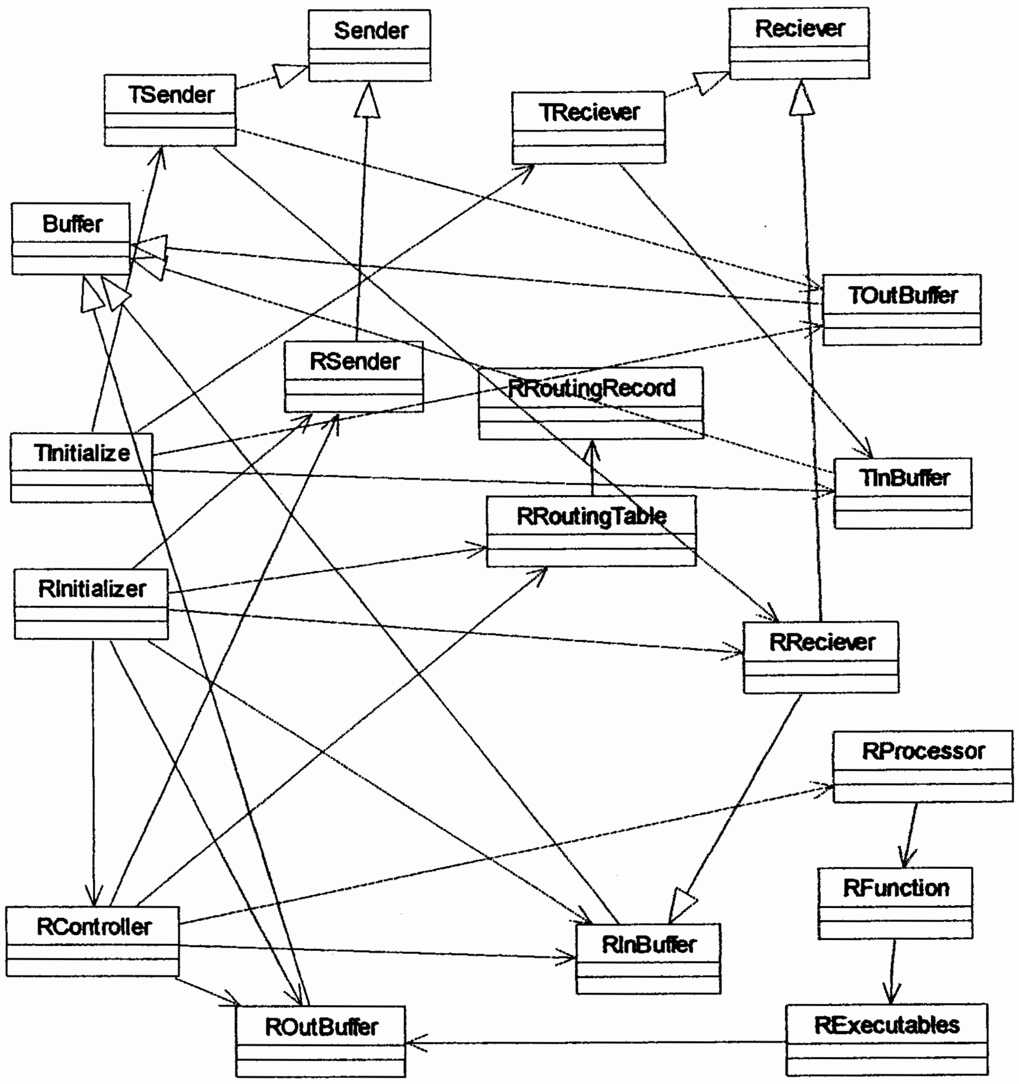


Fig 3.1 Class Diagram at initial level

During the analysis and early Design phase different concepts were identified. These concepts were mapped to classes. A description of each class is described in the following lines.

3.1.1 Buffer

This is high level class for every class used to store data in any form. It is used for the buffering at input and output. The basic operations that it provides are to add records and to get records. Different classes are derived from it depending upon the requirement that whether these are used in terminal or in router.

3.1.2 Sender

This is high level class for any class that is used to receive data either in the terminal or in the router.

3.1.3 Reciever

This is high level class for any class that is used to receive data either in the terminal or in the router.

3.1.4 TSender

This class is used to send data on a terminal. It has different attributes like IP address to which it can transfer data along with its own IP address, the port through which it transmits to applications and the port to which it has to transfer to the network. Its major operations are to get data and to send them.

3.1.5 TReciever

This class is used to receive data on a terminal. It has different attributes like IP address from which it can receive data along with its own IP address, the port through which it receives from applications and the port from which it has to receive from the network. Its major operations are to receive data and to store it.

3.1.6 TInBuffer

This class is storage for the data coming in from different locations like the applications and the network. It has operations like data storage and data retrieval. It stores full capsules along with its sender.

3.1.7 TOutBuffer

This class is storage for the data going out to different locations like the applications and the network. It has operations like data storage and data retrieval. It stores full capsules along with its intended receiver.

3.1.8 TInitializer

This class has the functionality to start a terminal in a right fashion and it also has all the data needed for the proper functionality of terminal.

3.1.9 RController

This class has functionality to control all the router functionality. It has information about incoming buffers as well as outgoing buffers. It continuously monitors the router. It can send data directly to sender and it can take data from a receiver buffer. It also monitors the changes in the routing records received. It then asks routing table to be changed.

3.1.10 RRoutingRecord

This class is basic building block for the routing table. It has format in which a routing record can be received and kept. This format is the interface, destination and hop count. In future implementations hop count can be replaced with the sum of weights of all edges in the graph.

3.1.11 RRoutingTable

This is composed of routing records and has complete picture of the network. It provides the operations of finding the path. It tells whether a path exists between current node and the requested one or not. It also tells about the shortest path.

3.1.12 RSender

This class is used to send data on a router. It has different attributes like its own IP addresses of different interfaces attached to it. Whenever it has to send data, it finds the path using routing table. It itself has no information about the current picture of the network. Its major operations are to get data and to send them.

3.1.13 RReciever

This class is used to receive data on a router. It has different attributes like IP address from which it can receive data i.e., all directly connected nodes whether these are routers or terminals along with its own IP addresses. Its major operations are to receive data and to store it in an in buffer.

3.1.14 RInBuffer

This class is storage for the data coming in from different locations like the terminal and the routers. It has operations like data storage and data retrieval. It stores full capsules along with its sender.

3.1.15 ROutBuffer

This class is storage for the data going out to different locations like the terminals and the routers. It has operations like data storage and data retrieval. It stores full capsules along with its intended receiver.

3.1.16 RInitializer

This class has the functionality to start a router in a right fashion and it also has all the data needed for the proper functionality of a router. It has the operations to find out available interfaces. It has operations to initialize all the buffers. It has functionality to start controller and receivers.

3.1.17 RProcessor

This class is heart of the whole system. It is responsible to provide the functionality of processing the capsules available inside the router. This class provides multiple options like getting a capsule from the inner buffer. It has the functionality of parsing the capsule as well. It also performs error checking on a capsule.

3.1.18 RFunction

This class has the functionality to convert code part of a capsule into recognizable functions. It also sets the parameters of all the functions present inside the capsule. This class is utilized by the processor class.

3.1.19 RExecutable

This class has all the executable function. It can be said as the data base of functionality available on a router. It provides the functionality of executing any of the available function. It also provides operations to search a function whether it exists on the router or not.

3.2 Sequence Diagrams

Sequence diagrams are important tools to describe a system. These describe activities in a system time wise. Following are the sequence diagrams identified in our system.

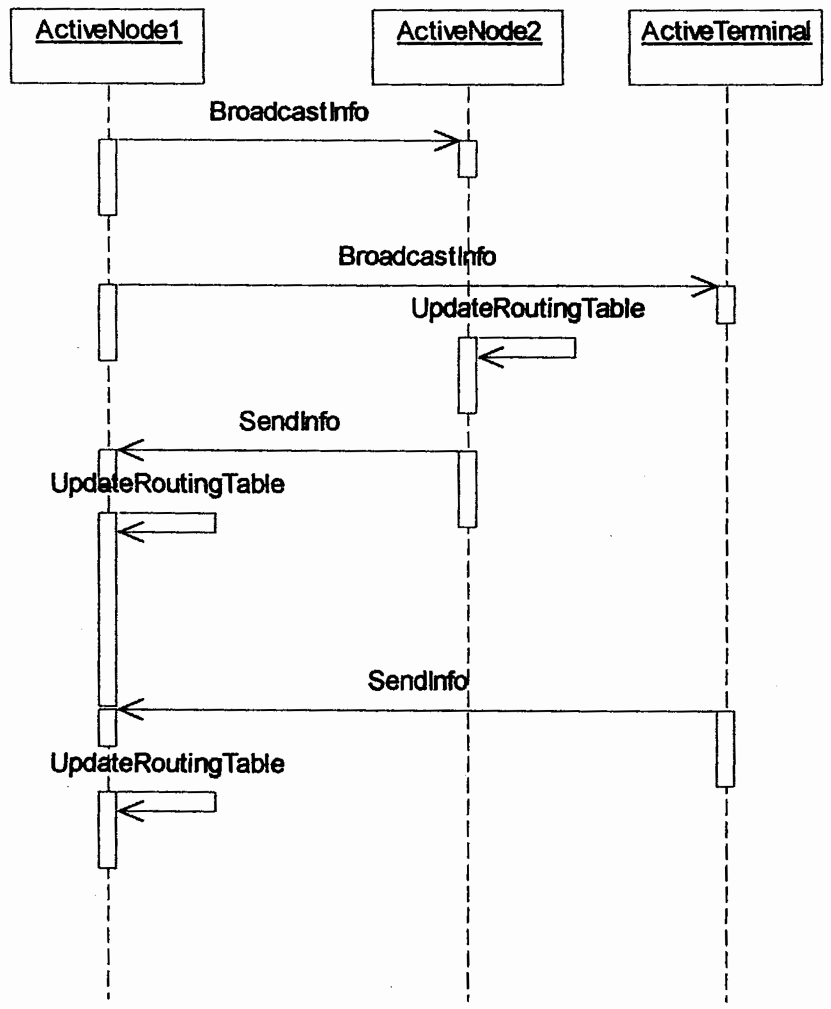


Fig 3.2 Establish Routing Table

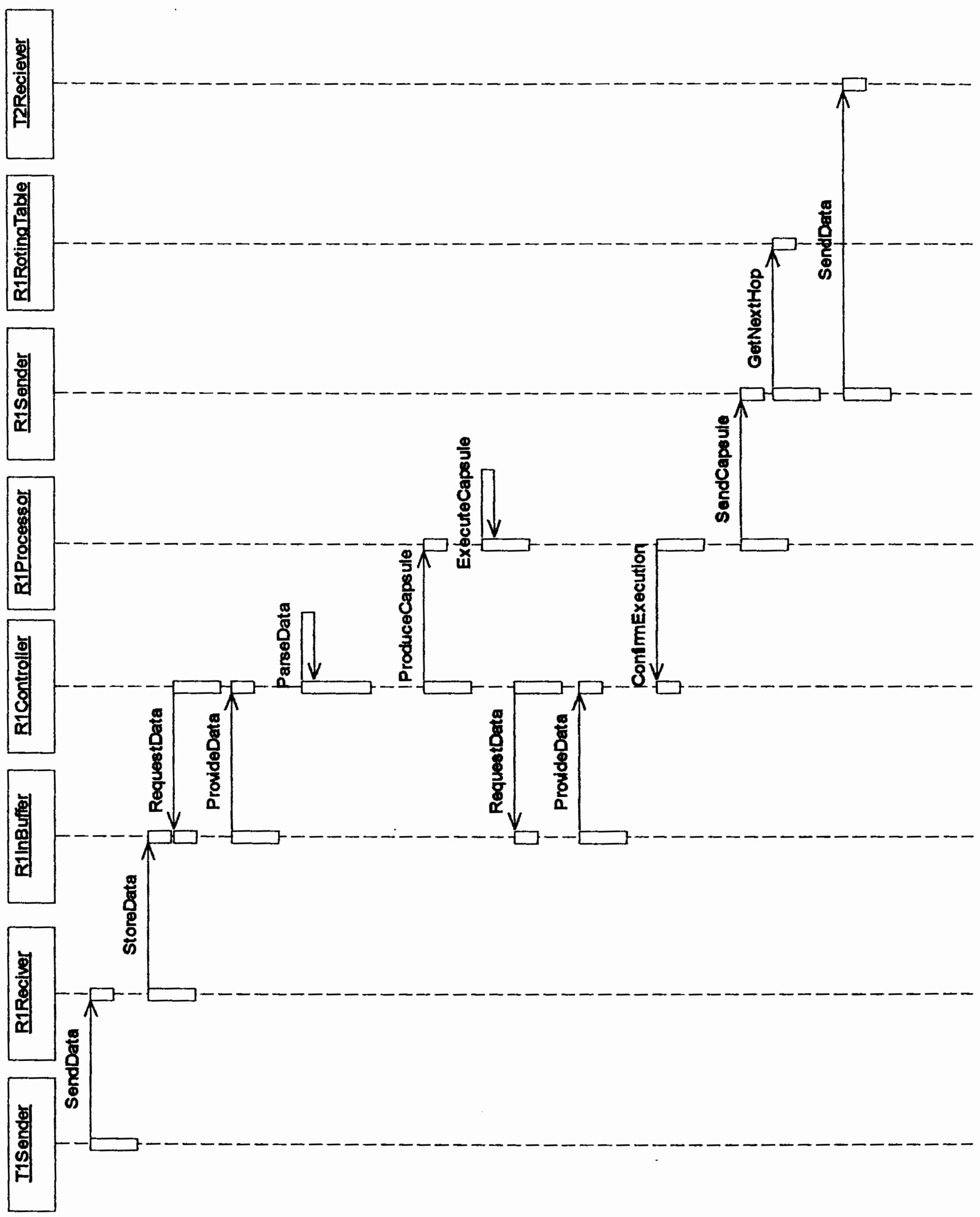


Fig 3.3 Execute Capsule

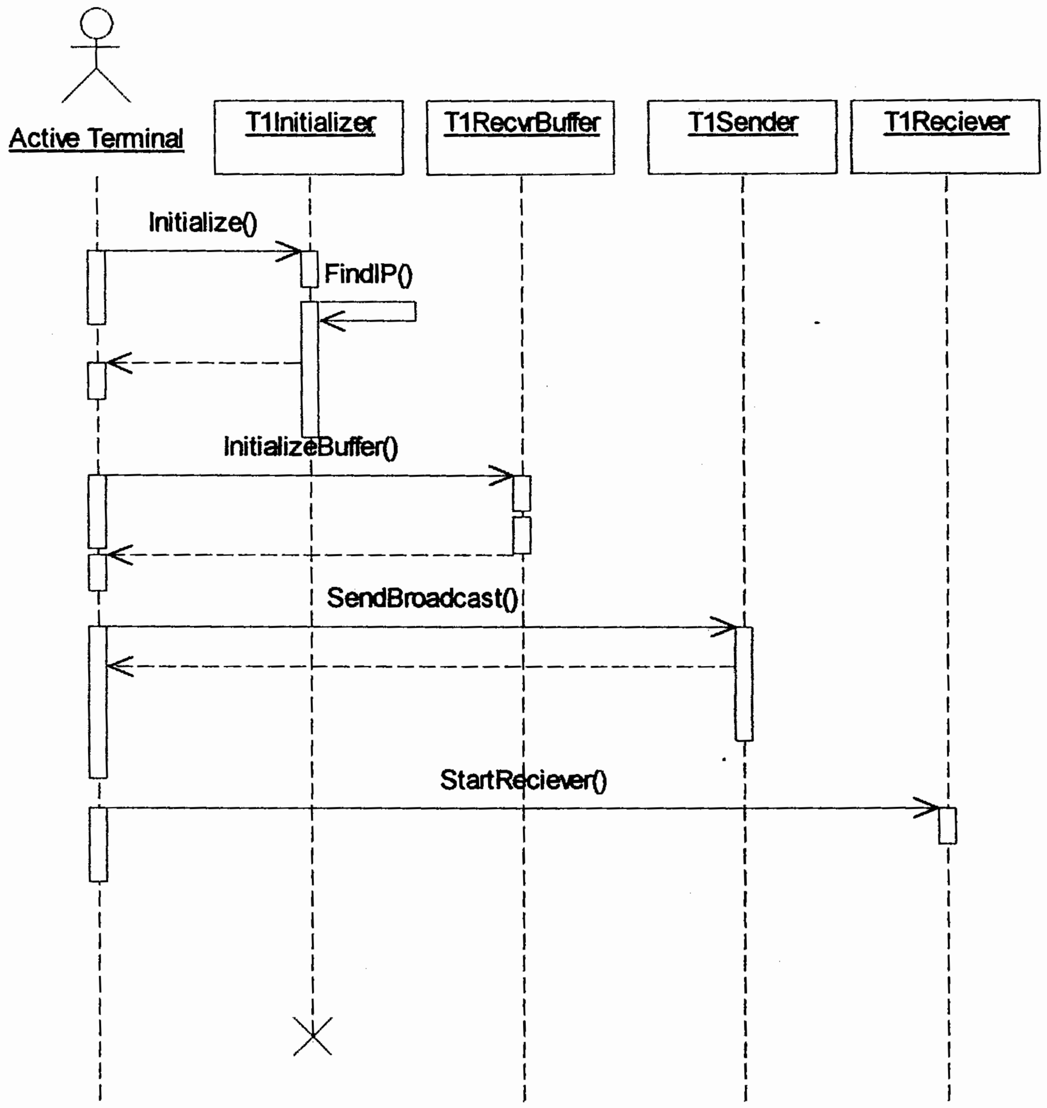


Fig 3.4 Terminal Initialization

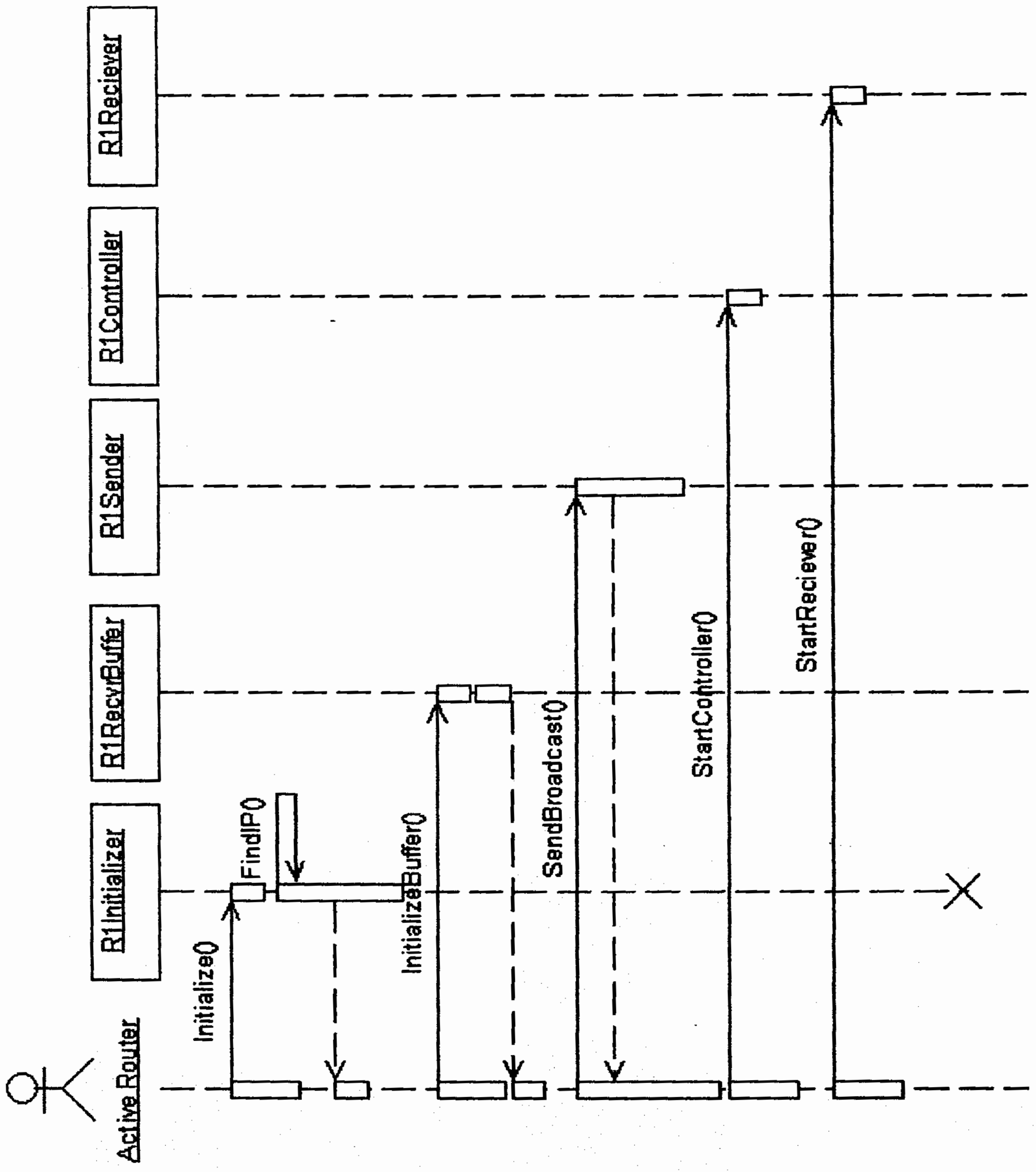


Fig 3.5 Router Initialization

Implementation

4. Implementation

One of the final and very important stages in the development is implementation. After a good analysis and design it is possible to implement the software using a variety of tools. Even then the choice of good tool and platform is very important. Wrong choice can make the life of developer very difficult. It is also possible that someone may opt for a tool that never provides the required functionality. Following is the discussion of selected platform and the tools.

4.1 The Platform

After a careful analysis we have chosen Microsoft Windows 2000 as a platform. This is done to fulfill many requirements. One of the very basic reasons is that Windows is the most common operating system in this part of world. Our ambitions are that our software must be used by maximum number of persons. By making it available for Windows means that it is available to most of the community.

We want substantial number of applications to be developed and more and more work to be done on this software after completion by the successors. To fulfill this property, the software has to be very easy to use. If we use an easy to use operating system then half of the difficulty is gone right at the start and if we use a difficult operating system then the difficulty is multiplied.

4.2 The Tool

We have chosen C++ as the language and Visual C++ as the compiler. The language, C++ is chosen because of design constraints. We need an object oriented language to implement because design is object oriented. There are multiple options. A very obvious alternative is Java. But Java language is ruled out because of many reasons. The most important reason is the speed. We are not implementing our software for a heterogeneous environment; instead our design goals specify only a single operating system therefore for the time being we do not need the compatibility provided by Java.

The choice of Visual C++ as a compiler is obvious because it is without any conflict the best available compiler for C++ in Windows. Not only it provides C++ implementation, it also provides many more options. To utilize maximum from the operating system support it is necessary to use a tool that can grab every option provided by the platform. Visual C++ does the same.

4.3 The Technology

As we have already described that it is necessary to utilize available options as much as possible therefore we have used already available building blocks. We have used WinSock for basic communication mechanism between routers and terminals. Winsock is also used as a building block for inter process communication on the terminal between the terminal and user application.

WinSock API provides many advantages for a programmer. Its specifications are same for every vendor. Different vendors can develop different implementations. Only limitation is that the implementation must adhere to the specification. More important it also has socket APIs described as Berkeley sockets specification.

4.4 Transmission Entities (Packets)

As it is clear from the previous discussion that our software is a prototype active network therefore most of its job is to send and receive packets. These packets are named as capsules. There are different types of capsules possible in this system. We have defined three types. These are control, routing and processing capsules. Control capsules are used to manage only the network connections and other links related tasks. Routing capsules are used to describe a change in the routing table. Processing capsules are the one that are used to carry executable code in them.

4.4.1 Capsule Formats

First thing required is to define capsule formats. All above described capsules are defined properly. Each type has its own well specified format. There is only one thing common in a capsule i.e., its first byte, that defines the type of capsule and therefore serves as the de multiplexing key for the capsule. Full format of each type is given below.

4.4.1.1 Process Capsule

The process capsule is the largest size capsule. It has following fields.

1- Type

2- Sub Type

3- Size

4- Source Address

5- Destination Address

6- Source Port

7- Destination Port

8- Data Length

9- Data

10- Number of Functions

11- Length of Function Name(s)

12- Function Name(s)

13- Number of Parameters

14- Type of Parameter(s)

15- Value of Parameter(s)

As described earlier, Type field is common in all types of capsules. It can have different values. In case of a processing capsule it should be used to identify capsule as a processing capsule. The value of type field is 'p' in this case. Subtype field defines sub type of capsule. Multiple sub types are possible for a capsule. These types are

a- Normal

b- Data less

c- Code less

Normal capsule describes that the capsule has both data and code. Data les defines only the code in the capsule and code less defines the capsule as having the data only and no executable in it.

Size field describes the size of capsule and it spans on two bytes. That means that the limit of size on a capsule is 64 KB. Source Address defines the address of the capsule originator and Destination Address defines the intended destination. Size of both fields is 4 bytes. Each address is 32 bit IP address. Source Port defines the port of originator and Destination Port defines the port of Destination. Size of each field is 2 bytes.

Data Length field defines the length of Data segment of the capsule in bytes. Its size is 2 bytes. Data length can contain zero if the capsule type is code less. Data field defines the data contained in the capsule. Its size is equal to number of bytes described in the Data Length field.

Number of Functions field defines total executables in the capsule. Size of this field is 1 byte. This field can have a value equal to zero in that case the capsule is a code less capsule.

If Number of Functions field has value equal to n then following fields will repeat n times.

1- Function Name Length
2- Function Name
3- Number of Parameters

Function Name Length field describes the length of function name in bytes. Its size is 1 byte; therefore maximum function name length is 255. Function name is actual function name. Number of Parameters field describes total parameters involved in the function. Its length is 1 byte. If number of parameter field has a value n then following two fields will repeat n times.

1- Type of Parameter
2- Value of Parameter

Type of parameter defines the type and its size is 1 byte. Currently supported types are byte, character, short, unsigned short, integer, unsigned integer, long, unsigned long, float, double. Apart from these values it is possible to have previous functions return value as parameter.

4.4.1.2 Routing Capsule

The Routing capsule has following fields in it.

1- Type
2- Level
3- Sender Address
4- Number of Entries
5- Destination
6- Cost

First field of a Routing Capsule is Type that defines it to be a routing information capsule. Its second field is Level. Its size is 1 byte. Level tells the receiver about the level of information that it has inside the capsule. Third field is Senders address. This field describes 4 byte IP address of the sender. Number of entries field defines number of routing records in the capsule. If this field has a value n then there must be n records in the capsule. Each record has following two fields.

1- Destination Address

2- Cost

Destination Address is 32 byte IP address and defines the address of router or terminal that is reachable from the sender. Cost field defines number of hops of destination from the sender.

Routing capsules are transferred only among the routers. Terminals do not participate in these type of activities.

4.4.1.3 Control Capsule

Different control capsules are possible. For the time being only one such capsule is defined. This capsule is for keep alive messages. It has following fields.

1- Type
2- Sub Type
3- From

First field is Type field. Second field Sub Type defines this capsule to be a keep-alive capsule. Third field is 32 bit IP address of sender. This type of capsule is used to communicate presence of a terminal or router in the network.

4.5 Components

The prototype is composed of different smaller modules, these modules work successfully with each other. These are

1- The Router
2- The Terminal/Host
3- The Application

Router is the most important part. It is the heart. This software runs on the systems that are designated as routers. To take benefit from our software one has to use at least the Router part. The terminal part is also necessary and this software runs on the systems designated as the host. Application is the part for which all this is done. All of our system is to support applications. These applications are to be written by the user.

4.5.1 The Router

As described earlier, this module works on the device designated as router. The system starts with initialization phase. In the initialization phase first of all software tries to find out number of interfaces available. Following code can be used to find out number of IP addresses on the system. This code works even if you have more than one ether net cards and IP addresses.

```
struct hostent *phe=gethostbyname("localhost");
        phe=gethostbyname(phe->h_name);
struct in_addr *pinaddr;
for( int i=0; ; i++)
    {
        pinaddr=(LPIN_ADDR)phe->h_addr_list[i];
        if (pinaddr==NULL)
            break;
        AfxMessageBox(inet_ntoa(*pinaddr),0,0);

    }
```

Note that in above code the address obtained are not being stored. Every newly calculated IP address overwrites the previous one. To use this code rightly, each IP address should be stored.

Initialization phase is not yet done. The router creates sockets for every IP address found in previous step. These sockets are created using socket function call. An example call is shown below.

```
int MySocket = socket( AF_INET,SOCK_DGRAM,PF_INET);
```

First argument AF_INET defines the address family, AF_INET defines internet address family. Second argument describes type of service used. In this case we are using SOCK_DGRAM. It is for datagram (UDP). Last argument is for protocol family, in our case it is internet. If successful, function returns a positive integer otherwise -1.

The router then initializes buffers. These buffers are for both incoming and outgoing data. The router sends its information to its nighbours. This is done by sending its IP addresses. The call is completed using following function.

send(socket_identifier,data,size_of_data)

First argument defines the socket on which data is to be sent. Second argument defines the data that is to be sent. Third parameter defines size of data in previous argument. On success function returns number o bytes actually transmitted.

Then router starts controller and receiver. Both these parts of router work in parallel and continuously. So basically these are threads. To create a thread following function is used.

Createthread(…);

4.5.1.1 Controller

This module keeps an eye on every part of router. It monitors incoming buffer and if it finds some data in those buffers, gets it and performs appropriate action. Working of controller can be described in following pseudo code.

```
If (DataFoundInIncomingBuffer)
    {
    GetDataFromBuffer();
    FindDataType();
    If (DataType == ControllerData)
        {
        OperateHere();
        }
    else if (DataType == RoutingData)
        {
        UpdateRoutingTable();
        }
    else if (DataType == Processing)
        {
        ParseAndExecuteCapsule();
        }
    }
```

When controller finds a capsule for itself i.e., Type of capsule is controller then it parses the capsule further to find out sub type. Currently only keep alive sub type is supported. If a capsule is found with a sub type keep alive than controller finds out sender and sender port from the capsule. If sender and sender port already have a connection with the router hen this capsule is discarded other wise a connection is established and an acknowledgment is sent. Upon receiving the acknowledgment the sender completes the connection. Along with this, an entry is passed to routing table as well.

4.5.1.1 Processor

This module is invoked by controller. When controller finds out a capsule in buffer, it checks the type. If type is a processing capsule then processors parse function is called. The processors parsing function works in the following manner.

```
GetType();
GetSubType();
GetSize();
GetSourceIP();
GetDestIP();
GetSourcePort();
GetDestPort();
GetDataLength();
GetData();
GetNumberOfFunctions()
GetFunctionsAndParameters()
```

When processor completes this task, it has separated the data and functions. Now functions represent the code to be executed which is present inside the capsule. Before execution certain other tasks are also done. First of all it is checked whether the function exists or not. If function exists only then it can be executed other wise discarded.

After completion of execution it is possible that the end result capsule may be differ from the original capsule received earlier. It is the reason that instead of sending a copy of incoming capsule directly a new composition is generated and that new capsule is sent on the link to next hop. To send this capsule, the processor finds out destination using the routing table and sends it on the link.

4.5.1.1 Receiver

Just like controller, receiver also works full time but it is totally different from the controller. There is only one controller in the whole router but there are more then one receivers in the router. In fact number of receivers is equal to the number of interfaces on the router. Each receiver waits for data on its port. As soon as it finds some data, it places that data in an associated buffer. So number of buffers it also equal to number of interfaces on the router. Now it is responsibility of controller to pick data from the buffers and perform accordingly.

4.5.2 The Terminal/Host

This module works as a link between network and application. Applications can not directly communicate with the router or the network. Each application will have to send its data to the host running on local system and that host is responsible for sending data to next hop that is if any router exists. In this way application developers have to do very little work in terms of finding next hop. In fact application developer needs no such information. It needs only to concentrate on core logic and capsule preparation.

4.5.3 The Application

An application is used as the building block of services to be used in the network. The applications are to be developed by users on their own. Each application should have some properties in common. The applications must be able to make proper format. The format is defined in previous sections. Application must forward their capsules to locally running Terminal/Host. Applications need not to communicate with the routers directly.

A sample application is developed by us. This application works like a route tracing program. It is defined in appendix A in detail.

User Manual

Appendix A-User Manual

Overview

The software is built as a simulator and it can serve as the basis for the development of Active Applications. The whole software comprises of three parts.

1- The Router
2- The Host
3- The Application

All these parts are described below one by one. The software assumes some things from the network. These are described below.

1- All computers must be connected directly to each other using cross cables.
2- There must be at least one router computer.
3- Router should be in the middle and not on the edge.
4- Hosts should be on the edges and not in the middle.
5- Routers must provide more than one interfaces i.e., should have more than one configured network cards.
6- There is no limitation, which system (Host or Router) is invoked first. An Example topology is described below.

Router

This is the heart of system. It has different windows. These are main window, configuration window, interface and routing window.

Main Window

When it is stated main window is displayed. Main window has following components.

1- Interfaces (List Box)
This list describes the number of interfaces (Ports) on a router. List displays an information like If0..192.168.0.3, where If0 denotes interface and next is the IP address of that interface. A node can have maximum 4 IP addresses.

2- Routing Table (List Box)

It describes current routing records available to the router. Each record has following format. 192.168.0.3... 192.168.0.5..1, where firs IP address is the destination, second is the interface on the router through which destination can be reached and third one is the cost. Cost denotes number of hops.

3- Activity (List Box)

This list box describes current activity. If some data is received or transmitted by the router on any of the port, the activity will be shown in the list.

4- Capsule Buffer (List Box)

This list displays the information about all the capsules received by the router.

5- Capsule Activity (List Box)

This list shows the activity being performed on a capsule.

6- Configure (Button)

The router functionality starts with this button. By clicking this button, configuration dialog box is opened.

7- Connection (Button)

This button displays information of connection on interfaces.

8- Routing (Button)

This button displays information of routing table.

9- Executable (Button)

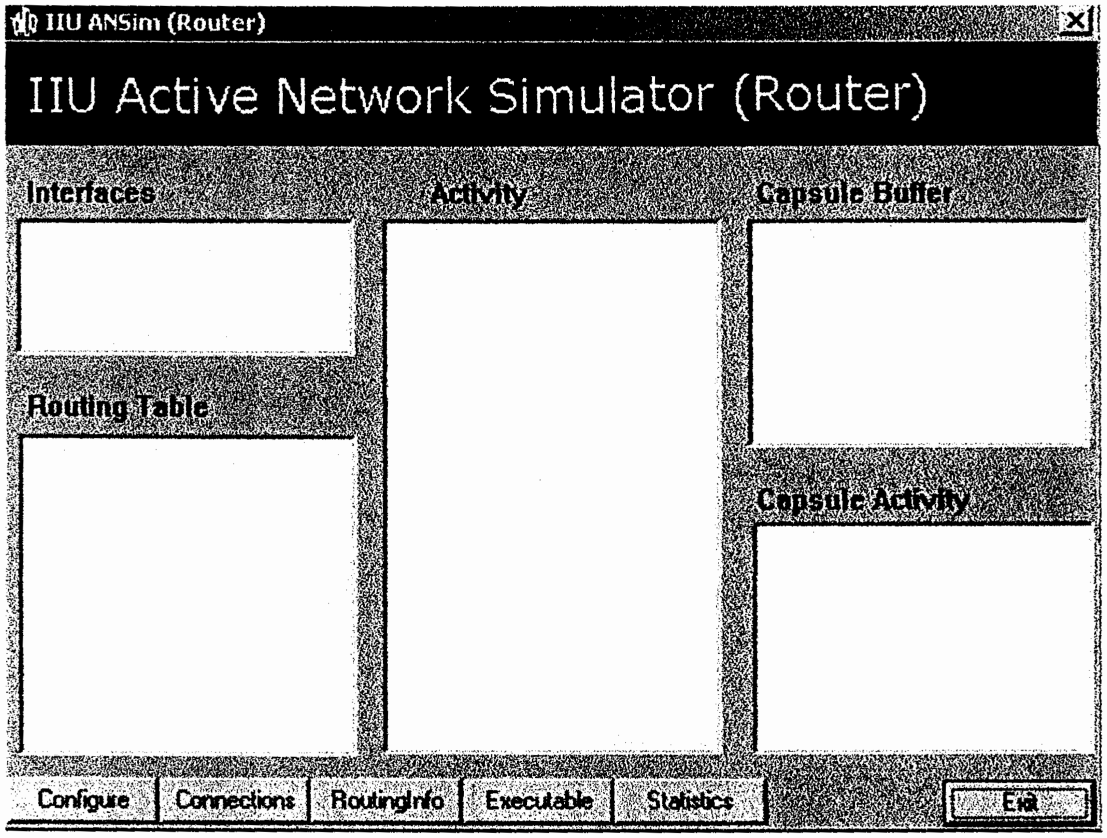
This button displays information of currently available APIs for execution.

10- Statistics (Button)

This button displays some statistics..

11- Exit (Button)

This is used to exit the program.



Configuration Window

When configuration window is displayed, it shows following components.

1- IP Address 1

User provides IP address of first interface, the interface 0.

2- IP Address 2

User provides IP address of second interface, the interface 1.

3- IP Address 3

User provides IP address of third interface, the interface 2.

4- IP Address 4

User provides IP address of fourth interface, the interface 3.

5- Set Address (Button)

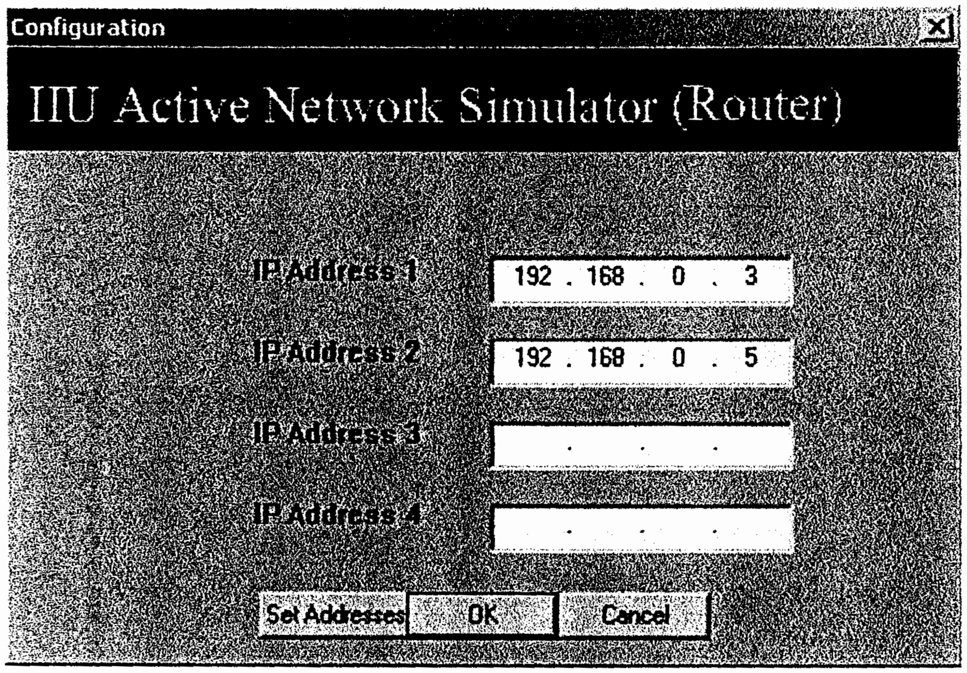
By clicking this, the addresses are set in the router.

6- OK (Button)

It is used to exit from the window.

7- Cancel (Button)

It is used to exit without saving.



IP Address 1: 192 . 168 . 0 . 3
IP Address 2: 192 . 168 . 0 . 5
IP Address 3: . . .
IP Address 4: . . .

Interfaces and connection Window

When configuration window is displayed, it shows following information..

1- All interfaces

It describes the interfaces available in the router.

2- Addresses of Interfaces

Every available interface has an IP address. It shows those addresses.

3- Logical port used by addresses

The port used for connection and to receive and transmit data is shown here. By default, it is 4000.

4- Connection on the interface

If an interface has got a connection with peer, it will display yes other wise no. It is not a TCP connection. It simply means that there exists a peer on this interface.

5- Peer Address

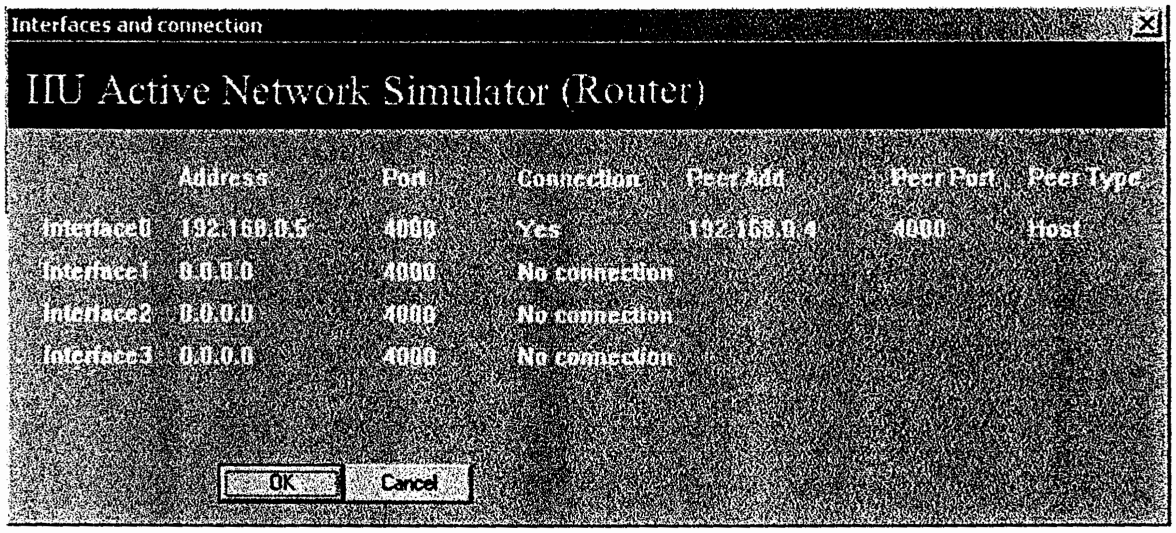
If a peer exists than it should have an address. This field describes the address.

6- Peer Port

Peer port describes the port which should be used to send data to the peer.

7- Peer Type

It describes whether the peer is a router or a host. If it is a router , it will be provided with routing tables otherwise not.



IIU Active Network Simulator (Router)

	Address	Port	Connection	Peer Add	Peer Port	Peer Type
Interface0	192.168.0.5	4000	Yes	192.168.0.4	4000	Host
Interface1	0.0.0.0	4000	No connection			
Interface2	0.0.0.0	4000	No connection			
Interface3	0.0.0.0	4000	No connection			

Routing Table Window

When Routing Table window is displayed, it shows following information.

1- Destination

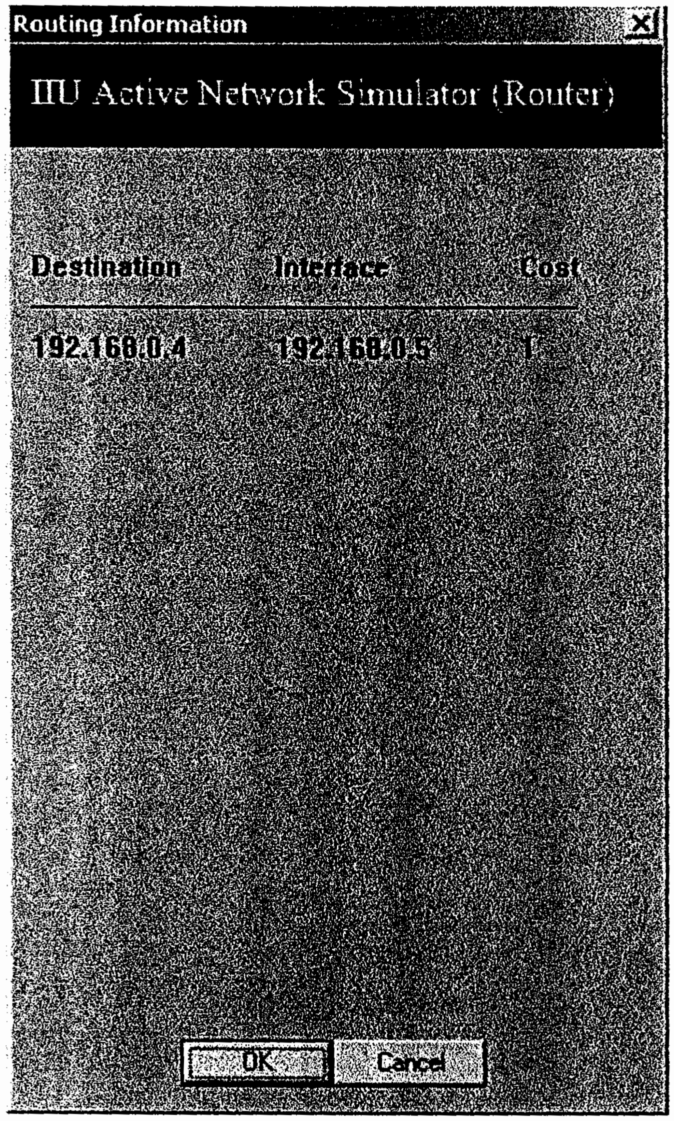
It displays the IP address that can be reached through this machine.

2- Interface

It describes the IP address of interface through which above mentioned Destination can be reached.

3- Cost

It describes the cost of reaching destination through described host. The cost is in terms of number of hops.



Destination	Interface	Cost
192.168.0.4	192.168.0.5	1

Host

It is used on edge machines. It provides the facility to actual applications to use active networks. It has only one window. This has following components.

1- IP Address

It is used to provide IP address of local system.

2- Set (Button)

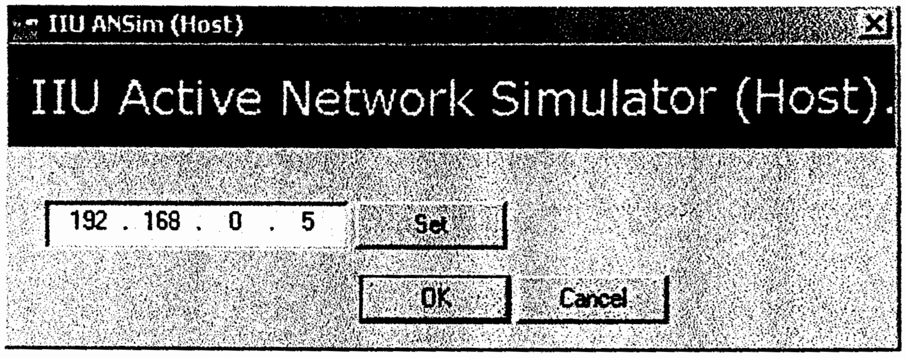
This button starts all activity. It assigns address to the required variables and then starts searching for peer routers and waits for incoming data.

3- OK

It is used to exit from the application.

4- Cancel

It is used to exit from the application.



Application

Application development is a concern of user. Still we have developed an example application that could help user to understand. The example application has a behavior like trace route. It has two parts, Trace route sender and receiver. The receiver application waits for data to arrive at him from any trace route sender. The sender makes capsules. The capsule goes through host and routers. At each router it executes and as a result, its data portion gets appended the IP of both interfaces through which it came in and went off. Its both parts sender and receiver are described below.

Trace Route Sender

It has only one window. The window has following components..

1- Destination IP

It describes the IP address of Trace Route Receiver

2- Destination Port

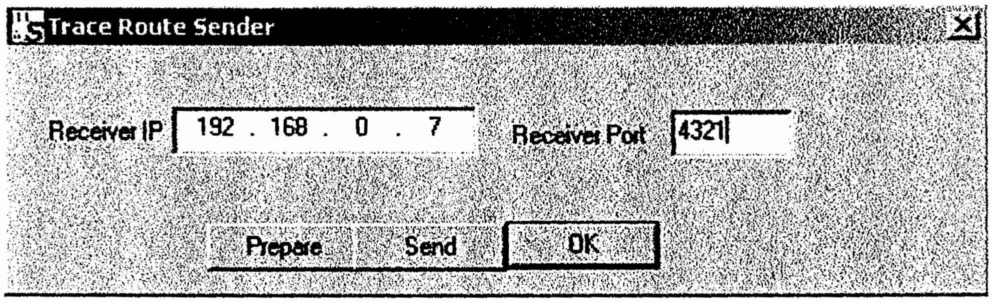
It describes the port of Trace Route Receiver

3- Prepare

By clicking this button the capsule is prepared.

4- Send

By clicking this button the capsule is sent to Host running on this machine.



Trace Route Receiver

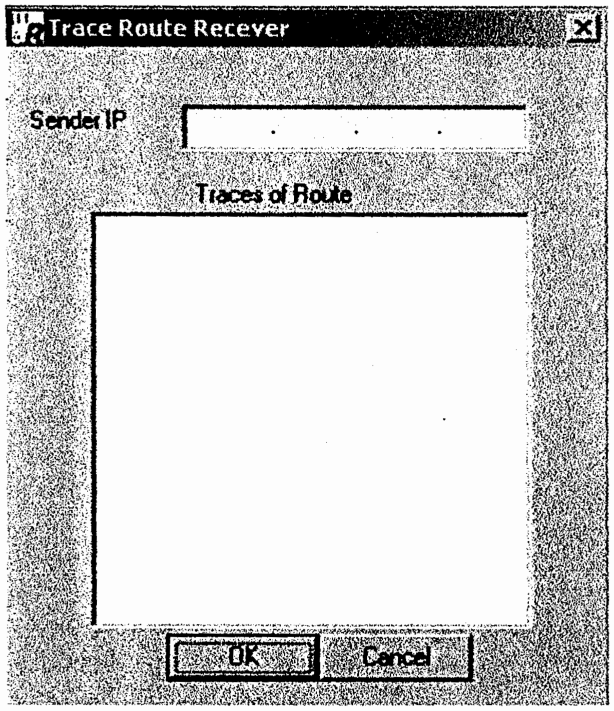
It has only one window. The window has following components..

1- Sender

When data is received the application finds out the sender and displays it here.

2- Received data

It describes all IP addresses traced by the capsule in order.



Glossary

Appendix B-Glossary

Active networks: Packet switched networks that allow passage of packets having executable code and intermediate nodes can process this code along with the header.

Broadcasting: Data transmitted by one sender is received by all receivers in a specific domain.

Caching: It is the process of storing data temporarily so that minimum requests could be sent outside.

Capsule: A packet having executable code along with data.

Congestion: State of an intermediate node when it is unable to store more data coming from network link.

Congestion control: Techniques used to remove and lessen congestion when it has occurred.

Latency: Time taken by a packet to reach the destination from sender.

Look around algorithm: An algorithm used to cache objects inside intermediate nodes. Objects are cached using a hierarchy having subdivisions called levels.

Modulo caching: An approach used to cache requested objects inside intermediate nodes efficiently.

Multicasting: Data transmitted by one sender is received by all receivers in a specific group, Mostly members of group are scattered.

On the fly up grade: It is the ability of intermediate node to upgrade its states dynamically.

Passive Network: A packet switched network in which intermediate nodes perform only header processing on the packets.

Programmable switch: A switch whose computations can be changed by the administrator of the network.

Round trip delay: Time taken by a packet to come back to the sender after being sent by that sender to any destination.

Stub node: The node not involved in the forwarding of packet. It is simply the node at the edge.

Transit node: It is the node involved in the forwarding of packet. It is simply the node in the middle.

Unicasting: Data transmitted by one sender is received by only one receiver.

References

References

[1] David. D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proc. SIGCOMM'88*, pages 106–114, Stanford, CA, Aug. 1988. ACM.

[2] J. H. Saltzer. D.P. Reed and D.D. Clark, End-to-End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, 1984.

[3] D. L. Tennenhouse and D. Wetherall. Towards an Active Network Architecture. In *Multimedia Computing and Networking 96*, San Jose, CA, Jan. 1996. A revised version appears in CCR Vol. 26, No. 2 (April 1996).

[4] E. Nygren et al. PAN: A High-Performance Active Network Node SupportingMultiple Code Systems. In *2nd Conf. on Open Architectures and Network Programming (OPENARCH'99)*, pages 78–89, New York, NY, Mar. 1999. IEEE.

[5] Y. Yemini and S. da Silva. Towards ProgrammableNetworks. In *Intl. Work. on Dist. Systems Operations and Management*, Italy, Oct. 1996.

[6] D. J. Wetherall and D. L. Tennenhouse. The ACTIVE IP Option. In *7th SIGOPS European Workshop*, Connemara, Ireland, Sep 1996. ACM.

[7] D. Wetherall. *Service Introduction in an Active Network.* PhD thesis, Massachusetts Institute of Technology, Feb. 1999.

[8] Bernard Cole. Active Networks and Cyber Terrorism, www.Embedded.com, October 2001.

[9] Jonathan T. Moore, Michael Hicks, Scott Nettlesy, Practical Programmable Packets, Computer and Information Science, Electrical and Computer Engineering, University of Pennsylvania. University of Texas at Austin.

[10] Danny Raz Yuval Shavitt, An Active Network Approach to Efficient Network Management Bell Laboratories, Lucent Technologies.

[11] An Active Router Architecture for Multicast Video Distribution
Ralph Keller, Sumi Choi, Marcel Dasen, Dan Decasper, George Fankhauser, Bernhard Plattner Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
Applied Research Laboratory, Washington University, St. Louis MO, USA

[12] Adam Wolisz, Christian Hoene, Berthold Rathke, Morten Schlager. Proxies, Active Networks, Reconfigurable Terminals: The cornerstones of future Wireless Internet. Technical university of Berlin, Telecommunicat Networks Group.

[13] Samrat Bhattacharjee, Kenneth L, Calvert, Ellen W. Zegura. Self Organizing Wide Area Network Caches. Networking and Telecommunications Group, College of Computing, Georgia Institute of Technology, Atlanta.

[14] AN Composable services working group, Composable Services for Active Networks, September 1998.

[15] Samrat Bhattacharjee, Kenneth L, Calvert, Ellen W. Zegura. Congestion Control and Caching in canes. Networking and Telecommunications group. College of computing, Georgia Institute of Technology, Atlanta.

[16] Y Chae, S Merugu, S. Bhattacharjee. Exposing Network support for topology sensitive Applications.

[17] Samrat Bhattacharjee, Kenneth L, Calvert, Ellen W. Zegura. On Active Networking and Congestion. GIT CC 96/02.

[18] Danny Raz Yuval Shavitt, An Active Network for Efficient Distributed Network Management Bell Laboratories, Lucent Technologies.

[19] David J, Wetherall and David L, Tennenhouse. The ACTIVE IP Option, Telemedia Networks and Systems Group. Laboratory for Computer Science Massachusetts Institute of Technology.

[20] Improving the Performance of Distributed Applications Using Active Networks Ulana Legedza. David Wetherall and John Guttag. Software Devices and Systems Group. Laboratory for Computer Science. Massachusetts Institute of Technology.

A simple practical Active Network architecture

Adnan Iqbal, M. Sikandar Hayat Khiyal, M.sher
Department of Computer Science,
International Islamic University, Islamabad.

Abstract: Active Networks are programmable networks. These networks provide the facility of executing code inside intermediate nodes. As compared to existing networks, these networks make user more powerful. User has the capability to perform computations of his own choice during the transmission of data. In this paper focus is on the simplest approach to develop an active network node. The design of node will provide the facility of programmability to the node.

Keywords: Active Network, Active Node, Programmability, Capsules, Congestion control.

1. Introduction: In an active network, the routers or switches of the network perform customized computations on the messages flowing through them. These networks are active in the sense that nodes can perform computations on packet, and modify the packet contents. Moreover, these computations can be customized. In contrast, the role of computation within traditional packet networks, such as the Internet, is extremely limited. Routers perform computations on packet header and they can change the header of packets flowing through them but do not change the contents of packets.

The computations performed by router on packet header are fixed and independent of applications that are involved in communication. In an Active Network computations are not fixed and not always independent of the applications.

Several problems with today's networks are identified for example the difficulty of integrating new technologies and standards into a shared network infrastructure, poor performance due to redundant operations at several protocol layers, and difficulty accommodating new services in the existing architectural model. Several strategies, collectively referred to as active networking, emerged to address these issues. Two things motivate research in Active Network: Increasing user requirements in this regard and enhancing technology day by day. User requirements are in the form of firewalls, web proxies, multicast routers, mobile proxies, video gateways etc. For all above applications computations of heterogeneous nature are required inside the network. Our goal is to define rules and basic principles for the development of a simple active network and then to develop a sample using same principles.

Next section describes the issues related to the development of Active Networks. Third section describes our approach and principles defined by us. Fourth section describes an example implementation and last section describes results of our research.

2. Issues in the development of Active Networks: Active Network technology encompasses different fields of computer sciences like networks, operating systems and language processing. Many different aspects must be kept in mind while developing a new architecture for active networks like security, interoperability, code execution and services introduction. Our motive is simplicity while fulfilling maximum requirements.

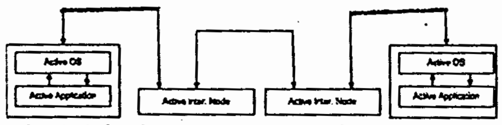


Fig 1. Ideal Active Network

3. Simplest Possible Architecture: In our approach the ingredients of an active network are active intermediate nodes and active applications. Active intermediate node is a router that has the capability of executing code contained in a packet flowing through it. Active application is a user program that generates and transmits packets containing executable code. Such a packet is termed as capsule [1].

3.1 Capsule:

A capsule must have information like executable code, data and address information.

Different options are available about the type of code to be transmitted and obvious ones are machine dependant object code, intermediate code and high level language constructs. All have different benefits and drawbacks. Object code is the easiest option to be used. If we use it then there is no need to compile or interpret the code at executing machine, so speed of execution will be fast. There is no need to develop any new compiler. Portability is its major drawback. This type of code will work only on one type of operating system and on one type of architecture. Another possibility can be to use any High Level Language. This will provide the benefit that the programs will become smaller. Principal disadvantage is that at every node program will be compiled and then it will be executed so speed of execution will be slower then binary code. Speed of execution and portability, both can be improved by using intermediate code and on the fly compilers.

Address information describes the addresses of sending and receiving hosts. It is implementation dependant.

3.2 Active Applications

Active applications are user programs that communicate with other user programs. These applications generate capsules, which have executable code. Active applications can define the behavior of intermediate node on these capsules by putting code of their choice. The flexibility in the behavior definition is dependant on the type of code used in the capsules.

3.3 Active Intermediate Node: It is the device used to connect multiple active applications with each other. We have identified different activities associated with an active intermediate device like receiving and forwarding of capsules, routing capabilities and most important execution of capsule code. To perform these functions active intermediate node can be divided in different parts like link manager, routing manager and processor.

Link Manager receives capsules from network links and forwards capsules to the network link with the help of routing tables. Buffers must be maintained for any kind of traffic through the node. Buffers must have all currently available capsules in the active node. It must have following knowledge, Source and destination of capsule, position of capsule in the active node and status of the capsule. Status will show whether the capsule is under execution or not.

Processor is responsible for execution. It must have knowledge about the position of executable code in the memory. After processing it has complete information of capsule that has gone through the execution cycle. It removes all such information except some very limited information kept as soft state.

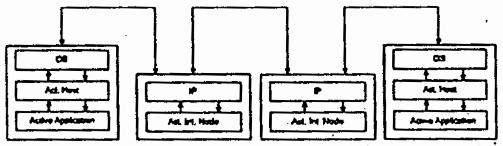


Fig 2. Active Network architecture that can work along with current IP networks.

Resource Manager has entry of all the resources being used by the executing transmission entities. These resources may be routing table or any hardware resources.

When an active intermediate node receives a capsule then it places the capsule in the buffer. After this controller will find the vacant area of memory so that it could place the code there. If it finds that place then it will place the code and execution will start. During execution if code needs any type of resources then these will be made available to it and entries will be made in resource manager. After the completion of execution all resources will be released and there entries will be removed from the resource manager. The capsule will be forwarded to the next hop after completion of execution.

4. Implementation: We have used above described rules to develop a sample network. We have developed it in such a way that it works over existing IP networks. Such a methodology is used because of testing purposes. We have chosen to implement it on Microsoft Windows 2000 and Intel processor. The language that we have chosen is C++. The compiler used for this purpose is VC++.

An important issue is type of code. The code used by us is in the form of high level function names and their parameters. Each intermediate node stores the executable code in the form of functions and each capsule has different function name and parameters values.

```
ReceiveCapsule();
FindType();
if (Type == executable)
        Execute_and_Forward();
else
        Forward();
```

Fig 3. Pseudo code for intermediate node

The scenario of transmission is given as: Active node will be running on a system on a specific port. Applications using activeness of the active node send data to some other node on the network. That data is in the form of capsule that is combination of code and data. When applications send capsule, it is received by active node working on that system. Active node of that system performs necessary operations. In this way activeness is achieved over passive architecture. If any active node exists in the mid way, it receives capsules. It parses the capsule, separates data and code and then executes the code. After execution the capsule is reformatted and transmitted towards next hop using a link.

4.1 Capsule Format: We have defined the packet format having the following fields. It is also described in figure 2.

- Type
- Sub type
- Source IP
- Source Port
- Destination IP
- Destination Port
- Length of Data
- Data
- Length of Code
- Executable Code

In this packet format executable code is further divided into functions. Each function has three parts length of name, name and parameters. The parameter field is further divided in parameter type

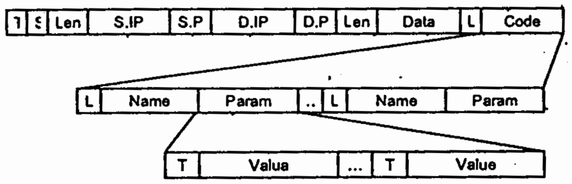


Fig 4. Capsule Format

and value.

4.2 Active Applications: Every active application that sends data to any other active application makes use of sockets to communicate with active node on the same system. When data is transmitted from an active application, it is given to active node rather than to be transmitted on the network link.

4.3 Active host

Active nodes receive capsules from applications and from network link. When capsule comes from a local application then it is transmitted over the network link. If it receives a capsule on a network link, it forwards it to intended application on a specific port.

This active host is necessary because we are implementing a totally new architecture over an existing architecture. By implementing active host, it becomes very easy to run our architecture without disturbing the current one.

4.4 Active intermediate node

In the beginning it initializes itself by setting up all interfaces on the system and buffers. After this it informs all neighbors of its presence and gets ready to receive capsules and other types of packets. Other packets can be routing or network control packets.

As it receives a capsule, it parses the capsule and separates code. Executable code is in the form of functions and each function contains names and parameters. Each function is executed one by one. After the completion of execution the capsule is reassembled and sent to next hop after consulting the routing tables.

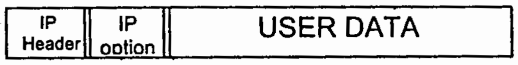


Fig 5. IP options in Active Networks

5. Results

Different attempts have been made previously to develop such an active network that can accommodate in an existing infrastructure. Such an attempt is described in [4] named as Active IP Option. The have used IP options of IP data gram as a method to implement activeness. There approach has different drawbacks because of limited size of IP options.

Our implementation of Active Networks has provided with the great deal of programming inside the network. It is better then previously done work as it provides generic facility of carrying the code and not limited in terms of size. Moreover, it can be applied inside current networks without disturbing them. We have developed a small application that takes benefit of activeness and works like route tracing application. We plan to develop more complex application as it has potential even for applications like multicasting and protocols conforming to user requirements. No portability problem as far as the type of code is concerned.

References:

1 Towards an Active Network Architecture David L. Tennenhouse and David J. Wetherall Telemedia, Networks and Systems Group, MIT

2 A Survey of Active Network Research David L. Tennenhouse, Massachusetts institute of Technology Jonathan M. Smith, University of Pennsylvania W. David Sincoskie, Bell Communications Research David J. Wetherall, Massachusetts Institute of

Technology Gary J. Minden, University of Kansas.

3 Practical active network, Nygren , Department of Electrical and omputer engineering, MIT.

4 The Active IP Option David J watherral and david L Tennenhouse. Telemedia networks and System group, Laboratory of CS, MIT. Sept 1996.

5 From Internet to Active Net David L tennenhouse, S J Garland, L sharira, M F kashoek, Laboratory of CS, MIT.

6 A survey of semantic techniques for Active Networks. Alan Jeffrey IanWakeman November 28, 1997

7 AN vision and reality,lessons from a capsule based system David Wetherall Department of Computer Science and Engineering University of Washington

8 An architecture for active networking Samrat Bhattacharjee, Kenneth L. alvert, Ellen W. Zegura Networking and Telecommunications Group College of Computing, Georgia Institute of Technology,

9 Towards practical programmable packets, Jonathen T Moore, Scott M Nettles.

This message is not flagged. [Flag Message - Mark as Unread]

Subject: CIT 2004: paper acceptance notific
From: BWerner@computer.org Add to Address Book
Date: Tue, 1 Jun 2004 15:05:24 -0700

Dear Author,
Congratulations! Your paper has been accepted for publication by the IEEE Computer Society Press in the Proceedings of the 2004 international Conference on Computer and Information Technology (CIT 2004) to be held 14-16 September 2004 in Wuhan, China. Please read these instructions carefully, and, importantly, remember to fax your signed copyright form to us by the due date, filled out correctly and legibly. Also, note you are no longer required to submit an abstract of your paper.
Paper ID: 446
Paper title: A simple practical Active Network architecture
Authors: Adnan Iqbal, M. Sikandar Hayat, M. Sher

IMPORTANT NOTE:
Please send a PDF or Postscript version of your paper, ONLY. If you Prepare your paper using Latex (see below regarding macros), please output your Latex file to postscript, and send the .ps version. Due to incompatibility problems we can no longer accept Word (.doc) files. Thank you.

COPYRIGHT FORM:
¤ Please FAX your signed copyright release form to Bob Werner at +1 714 761 1784.
¤ We cannot publish your paper without this permission.
¤ PLEASE NOTE: The copyright form is an attachment to this e-mail.

DEADLINES:
PLEASE NOTE: Your paper is due by 25 June 2004.
PLEASE NOTE: A signed copyright release form is due by 25 June 2004.
Please put the conference name - CIT'04 - on all correspondence, on the copyright form, and on any mailing envelope. This will assure that your correspondence is directed immediately to my desk.

ELECTRONIC SUBMISSION:
If using a web browser, please submit your paper electronically to our FTP site.
¤ The paper should be uploaded at:
ftp://ieeecs@ftp.computer.org/press/incoming/proceedings/cit04/
If you are uploading your paper using FTP transfer protocol software, you will need to login into our ftp site at: ftp://ftp.computer.org.
Logon as: ieeecs
Password: benefit
FILE NAME:
Please refer to the attached author list text file for your paper number,name, and paper title. Please name your conference paper file per this sample:

    <paper ID number_last_name_first_initial.pdf>
ATTACHMENTS:
There are some customized attachments to this author kit. They contain:
¤ AUTHOR GUIDELINES
    Instructions on formatting your paper for the proceedings.
¤ COPYRIGHT RELEASE FORM
    A copyright release form that MUST BE SIGNED AND RETURNED WITH YOUR
    PAPER. Note that you are stating
    that the material in your paper is original and you have not previously
    released copyright to it. We cannot publish your paper without this
    properly filled-out and signed form.
¤ REPRINT ORDER FORM
    To use if you wish reprints of your paper. Please make checks for
    reprints out to the IEEE Computer Society.
    (The reprints.pdf attachment below will allow you to type your
    information directly into the form before you print it out.)
There are three attachments in Adobe Acrobat PDF file format. They are instruct.pdf, copyright.pdf, and reprints.pdf.
If attached files become unusable during transmission for any reason, you may also access uncustomized copies of them at our FTP site. Available at:
ftp://pubftp.computer.org/Press/Outgoing/proceedings/

LATEX MACROS:
LaTeX formatting macros are also available on our FTP site.
¤ Proceedings 8.5 x 11 format - Filename: <LaTex macros.zip
Available at:
ftp://pubftp.computer.org/Press/Outgoing/proceedings/
If you have any questions, please feel free to contact me.
Sincerely,
Bob Werner
IEEE Computer Society Press
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1264
Fax: +1 714 761 1784

Are you a member of the IEEE Computer Society? For membership information,
visit http://computer.org/join

Attachments:
(See attached file: instruct.pdf) (See attached file: reprints.pdf)(See attached file: FORMAT.pdf)
(See attached file: cit04 copyright form.pdf) (See attached file:
cit04 accepted papers.txt)

This message is not flagged. [Flag Message - Mark as Unread]

Date: Wed, 2 Jun 2004 20:03:45 +0900
To: ai_hr2000@yahoo.com
From: hwang@u-aizu.ac.jp Add to Address Book
CC: hwang@u-aizu.ac.jp
Subject: CIT 2004 Author Notifications

Dear Adnan Iqbal,

The IEEE Computer Society Press have already sent the author kit to you. According to the schedule and guidelines written in the author`s kit, please DIRECTLY send the final version of your paper to IEEE CS PRESS before the deadline.

The authors/presenters are likewise required to abide by the following rules:

1. Each accepted paper should be registered and presented at CIT 2004. The paper should be registered based on the fees in the attached registration form. This applies to all papers, regardless of whether the presenter is a student or not. If the presenter is going to present multiple papers, each paper should be registered as required.
2. Each accepted paper should be registered by June 25, 2004. If we do NOT receive the paper registration on this date, your paper will be automatically withdrawn from the CIT 2004 Proceedings.
3. A full paper can have at most Eight (8) pages. Only 8 pages will be printed even if the paper is longer than the imposed limit. To some authors from China: the order of your name should be "First_Name Family_Name" as posted on the Program.
4. Please contact with Dr. Xuhui Li at Wuhan University (Email address: lixuhui@whu.edu.cn) about the hotel reservation for CIT 2004 attendees, or book hotel online by following the link http://www.sinohotelguide.com/wuhan/.
5. The conference Advanced Program (Preliminary) has been posted on the web. Any mistakes, email us.

An invitation letter will be sent to participant(s) who wish to get such a letter for purposes of supporting the approval of your business trip, visa application, etc.

Thanks all. We look forward to receiving your CIT 2004 registration soon.

Best regards,
CIT2004 Program Committee Chairs
Hui Wang, Zhiyong Peng, Atsushi Kara
==
The 4th International Conference on Computer and Information Technology (CIT2004)
Wuhan, China, 14-16 September 2004,
http://sm1svr01.u-aizu.ac.jp/~cit2004/