Parallel Processing of Data on the Fly in Active Networks

Akbar[1], Buhari[1], Kalamullah[3], Sahalu[2], Saleem[1]
[1]Lecturer, King Fahd University of Petroleum and Minerals, Saudi Arabia.
[2]Assistant Professor, King Fahd University of Petroleum and Minerals, Saudi Arabia.
[3]Researcher, Univeristy of Duisburg, Germany.
Email: mibuhari@ccse.kfupm.edu.sa

Abstract:

In traditional packet networks, like the Internet, the amount of computation performed on the packets is extremely limited or is performed in an ad hoc manner. As the Internet grows there is an increasing demand by users to perform user-driven computations or services inside network nodes. In many cases, these services are implemented at nodes, such as firewalls, which adopt the facade of routers but perform application-specific processing that trancends conventional architectural guidelines. In an active network, the routers or switches of the network perform customised computations on the messages flowing through them.

The programs that drive the customised computations are either injected into the routers or are carried inside the packets in addition to the data inside the packets. Using the active networks technology, customisable, user-specific computations can be performed in a systematic rather than ad hoc way as in unconventional networks. The use of active networks potentially exposes a huge amount of computations taking place inside network routers. Parallel processing can be used to speedup those potential computations.

We use active networks and parallel processing in order to exploit the benefits of both worlds; use active networks to introduce user-customised computations and use parallel processing in the network nodes to speedup the computations while optimising resource usage. We have done experiments in two ways by writing a real-time application and also by simulation using OPNET tool. For real-time application we use Java for our active networks implementation because of its support for network programming and PVM (Parallel Virtual Machine) for Parallel processing.

Keywords: Active Network, Parallel Processing, Matlab

1. Introduction

The emergence of Active Network technology has attracted many academics and researchers to become involved in the development of this technology. Active network has potentially much to offer the Information Technology world. First, exploiting this technology can reduce the time required for the standardization process of new network services. Second, active network shifts the conventional network paradigm: from a passive node that only transfers bits to a more general processing engine like an end station which supports customised computation on user‘s data. Furthermore, active networks can also be used for enabling on-the-fly modification of network functionality, for example to adapt to changes in link conditions.

The use of active networks potentially exposes a huge amount of computations taking place inside network routers. Parallel processing can be used to speedup those potential computations. We envisage that parallel processing will be a powerful tool when used in conjunction with active networks especially in a LAN environment: when the load distribution on the nodes in a LAN is skewed with some machines heavily loaded and others lightly loaded these two technologies can be used to good effect taking advance of low-communication latency of a typical LAN.

In this article, we set out to use a synergy of these two techniques in a LAN environment to investigate the prospects of adding user-customised computations inside networks. Another goal of this paper is to use parallel processing to improve the performance and resource utilisation of the underlying active networks system in a LAN setting.

The remainder of this paper is organised as follows. We outline the related work and general architecture of our active networks system in Section 2. The PVM architecture is described in Section 3. The way these two architectures are combined and put to work together is then presented in Section 4. The experimental analysis using the simulation system is described in Section 5. The limitations of simulated experimental setup, is discussed in Section 6. The experimental analysis using the real-time system is described in Section 7. The advantages offered by implementing the hybrid technology is discussed in Section 8. Section 9 concludes.

2. Active Network Architecture

The general architecture of the active network is made of three main layers (as shown in **Figure 1**), namely, the active application layer, the executive environment layer and the node Operating System layer. The application

layer is used to add certain applications that are being processed at the node. The executive environment layer is to do the active processing on the capsule. The usage of the node Operating layer is the operating system that lies behind the active environment. The Operating system is noted here because the system that is being developed in a heterogeneous environment and so the functions that are invoked are basically dependent on the operating system too.

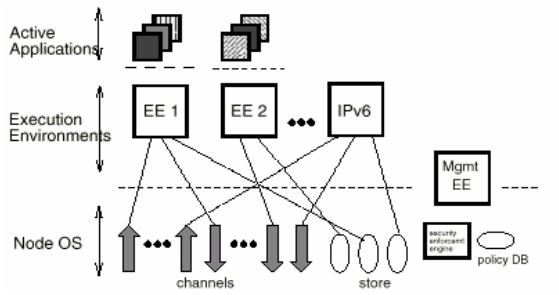

Figure 1. Active Network Architecture [Cal99]

The most important layer that contributes most to the 'active' term of active network is the Execution Environment (EE). It is in this layer a new service from the local machine is uploaded and the request from other machine for a new service is submitted and processed. EE deals with network operating system and service layer. A packet is directed to EE in order to access network services.

In Java architecture, EE lies above the Java runtime and under the service layer. From the Java point of view the architecture of Active Network is as depicted in **Figure 2**.

Active network paradigm allows program to be injected into an active node. A Program may reside in a switch/router [Bha96, Bha97, Wet98] and is activated in the EE only after the packet (i.e., capsule) from the client that carries identifiers or reference to the code, is processed and verified. Another approach lets the packet carries the whole code [Sch98, Ale97, Ban97]. In the latter approach, the node is also active in a sense that it allows computation up to the application layer.

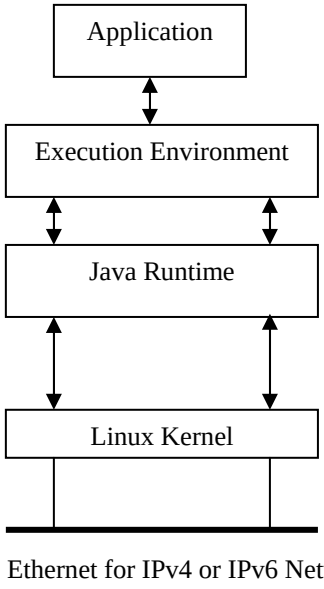

Figure 2. Java Architecture and EE

We prefer for our active network platform that the packet carries only identifiers or references to the services that reside in the node due to the following reason:

1. It maintains the packet size to minimum
2. The idea is to optimise the resources available in every machine connected to a LAN. Code can be stored in non-exhaustive machines.
3. The packet can be designed in conformance with the Parallel processing architecture interest.

3. Parallel Virtual Machine (PVM) for Resource Usage Optimisation

The essence of using the PVM in our experiments here is to avoid making a particular node as a computation hot-spot and to ensure that computing resources are optimally utilised, as much as possible. As Amdahl´s law explains, the speed of increase in the processing can be done using the parallelism.

S(n) = Execution time using one processor /
Execution time using a multiprocessor with *n* processors.
= t_s / t_p

Maximum speed up, $S(n) = ts / [fs + (1 – f) ts / n]$

$= n / [1 + (n-1) f]$

Efficiency = Execution time of one processor / [Execution time using a multiprocessor * number of processors]

As per Amdahl´s law, depicted in **Figure 3**, the processing speed can be increased using Parallelism. The whole job of operation depends on the situation and the nature of the interdependency in the operations inside a job. In the case of active networks, we are already aware of the current status of the jobs and the list of jobs that we are planning to do. Allocation of certain specific machines for certain parts of the job can be done to speed up the process.

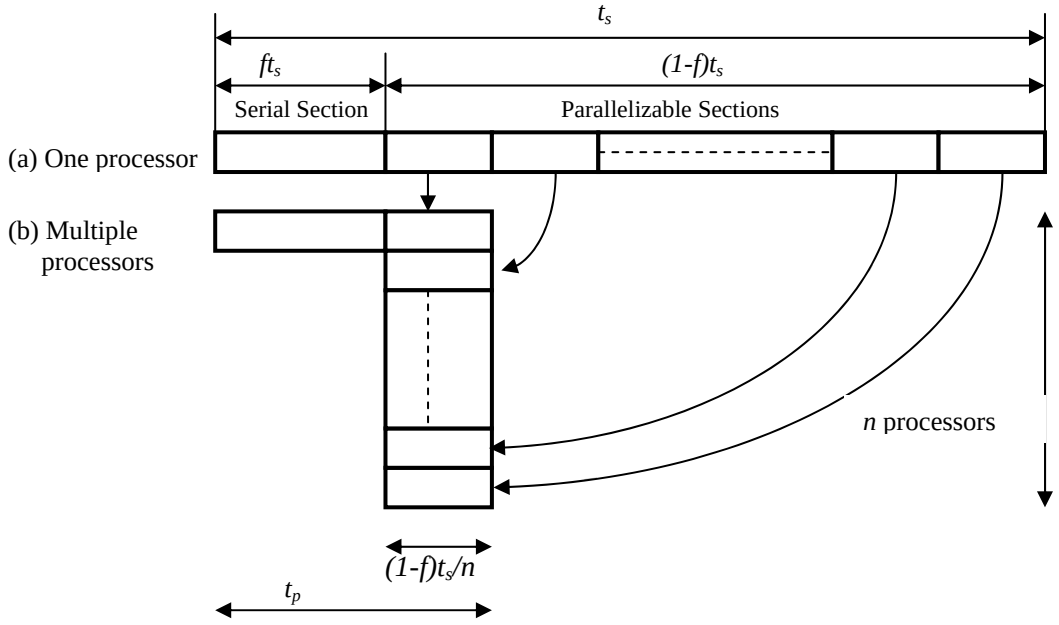

Figure 3. Amdahl's Law [Wil99]

In the case of PVM usage, the most important factor to be considered is the area of the processing. The active networks with PVM can be well implemented in a local area network (LAN) than the Internet as a whole.

3.1. Parallel processing part

The parallel processing is handled using the PVM software. This software runs both on Windows as well as Linux platform. Being a heterogeneous system, PVM helps us to do the processing faster, without any problem of heterogeneous case. The router or the switch is used to run the PVM software and the other connected machines run the PVM Daemon. The router, once it attains or receives a job to be processed, it distributes the job and executes it parallely with the help of other computers that run the PVM daemon.

3.2. Pipelining

If we are using a connection-oriented data transfer using TCP sockets in Java, then we can just use the concept of pipelining. Pipelining is the case by which one part of the job is handled by one router and the

intermediary result is sent to next router to process further This can be handled by making the routers handle different set of jobs, by sharing their respective job among their PVM Daemons.

4. Experimental analysis using OPNET (a network simulation tool)

The setup of our experiments consists of a 16-node network designed and simulated using OPNET, a powerful tool for designing and simulating computer networks. Our network model shown in **Figure 4** consists of 4 subnets each comprising of 4 nodes. We made use of OPNET's random packet generation algorithm at each of the nodes. We implemented a static routing algorithm that fixes the next router through which packets can be forwarded from the current one based upon the destination subnet and the host address present in the packet. The node part of the host and the hub are shown in **Figure 5**.

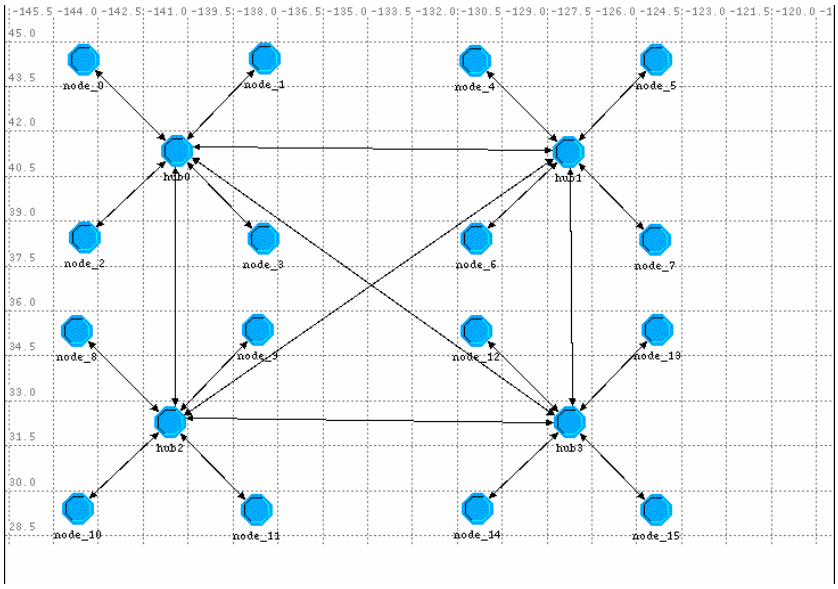

Figure 4. Our Network Architecture

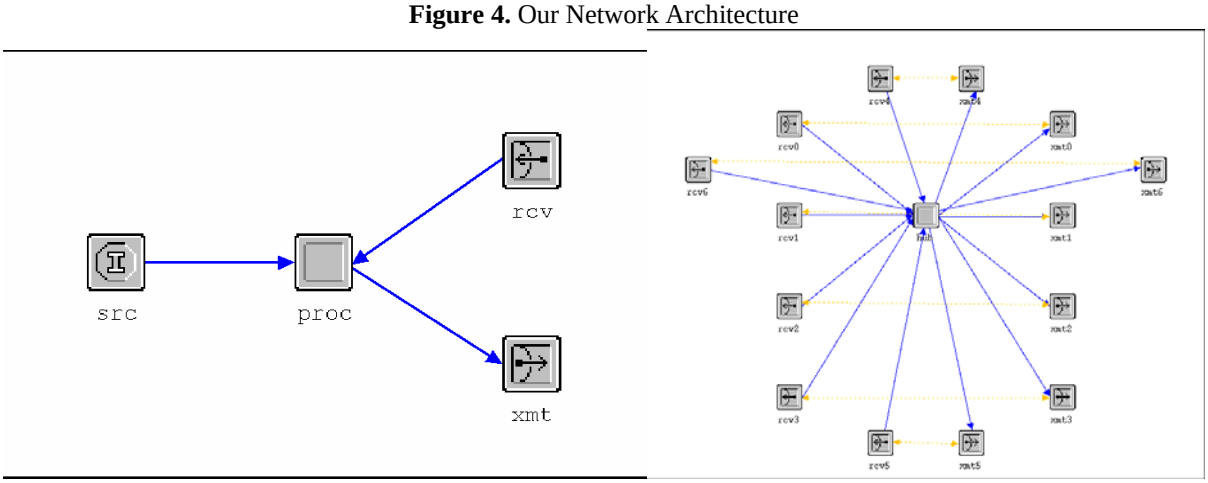

Figure 5. Node diagram of the host and the hub

- **Processing part of host and hub**

The processing portion of the host and the hub are shown in **Figure 6**. The host's "init" part is used to make the packet destination address generation to be between 0 and 15 because the network has 16 nodes only. The generated packets are then forcibly sent to the idle part. The transmitter then transmits the packet along any one stream depending upon the destination address specified.

At the hub, the checking for the destination address is done and packet is forwarded in the respective transmitter stream. Once the packet reaches the destination, the packet is now processed to identify the time it has taken to reach the destination from the source. Depending upon the time duration taken the SQ message is generated to the source.

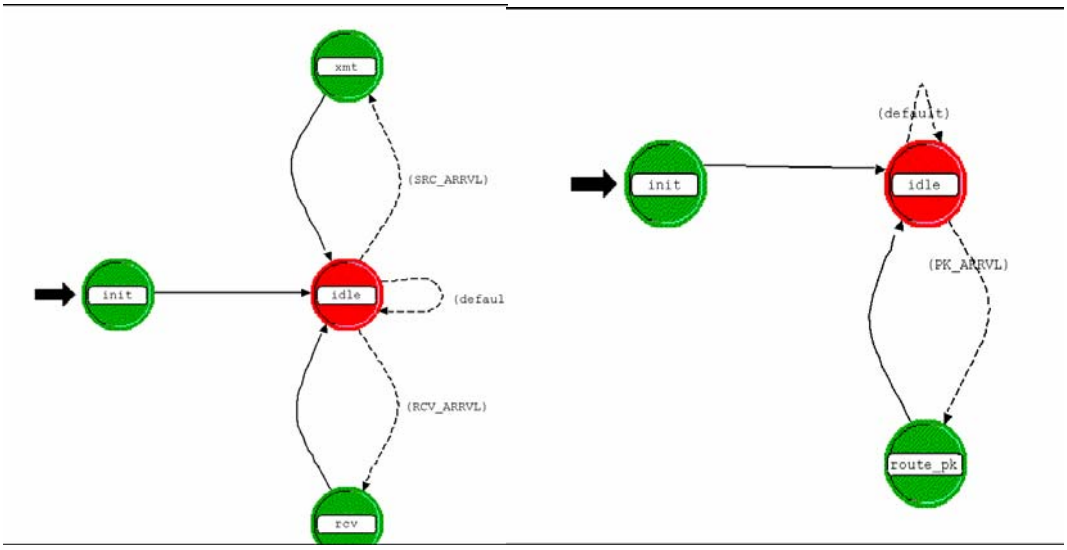

Figure 6. Process module of the host and the hub

As an example, let us suppose that the result of some computation is to be transmitted from node_0 to node_15. The data can pass either through hubs [0, 2 and 3] or [0,1 and 3] see **Figure 4**. PVM resides in active routers (hubs) and also in all the nodes. When the data is passing through first router, the router will partition the computation process to be done on the data into four modules based on availability of nodes. Each module along with its associated data is sent to one PVM process which is virtually connected and running in parallel at different nodes connected to the router. The nodes will process the data and send the result to the router. Then the router will forward the data to the next router, which will also do the same and forward the result to the destination [node_15]. Congestion is controlled in router itself, since the router is active and it can run congestion control algorithms.

In our experiments reported in this paper, we simulated the throughput and utilization of our network in order to demonstrate the potential practical usefulness of our ideas. Throughput, utilization and ETE delay are defined as:

> ***Throughput (Packets/sec)****: represents the average number of packets successfully received or transmitted by the receiver or transmitter channel per second.*
>
> ***Utilization****: represents the percentage of the consumption of date of an available channel bandwidth, where a value of 100.0 would indicate full usage.*
>
> ***ETE delay****: End to End delay is the time taken by the packet to reach the destination from the source (one end to other end). The packet created time is calculated.*

Figures 7 and **8** respectively depict results of our simulated network's ETE delay between source – hub, hub – hub and hub – destination and ETE delay between source and destination.. **Figures 7** and **8** have two component graphs; the graph in the upper part shows the result obtained when the active version of the network is simulated and that in the part shows the corresponding result for the non-active network. Apart from the active and the non-active differences in the models, the data used in the experiments is the same and all experiments were conducted under the same (as much as possible) network conditions using operating system kernel procedures.

The horizontal axes in all the figures represent the simulation time in seconds. The vertical axis in **Figure 7** and **8** represents transmission time.

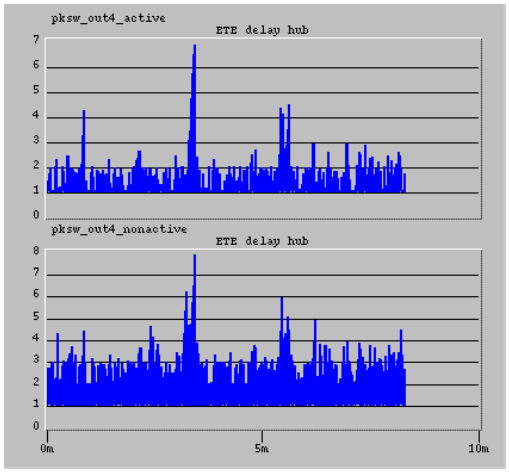

Figure 7. Comparative ETE Delay of the hub of the Active and Non-active Systems

The ETE delay in the hub is less [**Figure 7**] for each packet in the case of active system than non-active system. This is because after doing computations on the data in routers, only the results of computations are passed to the destination where the resultant data is small in size compared to the original data sent by the source. Here, destroying of packets is also done based on the following conditions.

1. The packet is destroyed if it is known that the packet may not reach the destination.
2. If the network traffic is high.
3. If *n* number of packets were sent by the source to router and in the router, after doing computations on the data in *n* packets, if only the result to be sent to the destination and the number of packets needed to carry the resultant data is less than *n*, then the resultant data is stored in the some packets among n packets and rest of the packets were destroyed.

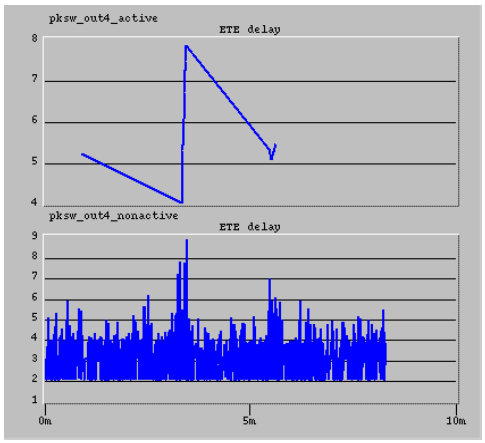

Figure 8. Comparative ETE Delay between the source and the destination of the Active and Non-active Systems

The ETE delay between source and destination is less [**Figure 8**] in active system than the non-active system because the result of computation is transmitted from the router to the destination. Since in our case the size of the resultant data is less the ETE delay is also less. The simulation time also decreases when compared to non-active system, because the computations are done in parallel. Throughput of the Active and Non-active Systems is the same.

5. Limitations of our simulation system experiment

First we note that although our system is simulated, all real computations are performed on the host computer in a separate process from the OPNET simulation, typically executing C programs and making system calls, in order to respond to the OPNET simulation tool. In a concrete (non-simulated) system the parallel computations cannot be done unless the computations are done on more than one machine. So we have designed the real time system.

6. Experimental analysis using real-time application system

We have setup an active network platform with three computers. Two computers act as an end system and another computer is treated as a router. The configuration is shown in **Figure 9**.

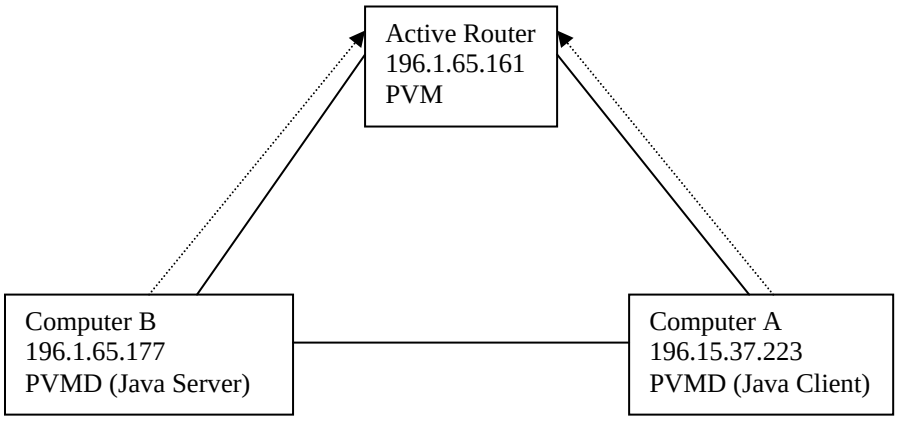

Figure 9. The Configuration Scenario

Although the machines are physically connected through the Ethernet, we design that all communication should be made through router. PVM resides in active router and Computer A and Computer B act as PVM daemon.

The current implementation is written in Java and runs as a user-level process under Linux, Windows 98 and Windows NT. We choose Java because of its support for safety and mobility. Other reasons are its flexibility as a high-level language and support of dynamic linking/loading and multi-threading.

The user interface developed is run at both the client and the server side. The server side executes the concerned Protocol (TCP or UDP). The client part finds out the source address by using the Java program. This is done so as to make sure that the user doesn't enter any invalid IP address or false IP address. Then the user is asked to supply a destination, a transmission Protocol, a flag setting based on the packet is active or non-active, and if it is an active packet, the application to be run at the router. Then, the packet is formed as shown in **Figure 10**.

Active / non-active flag	IP address destination subnet	IP address destination	File name	File contents	Capsule data

Figure 10. Packet Format

After the user provides the information for the packet, the Access control checking is performed. The Access Control List (ACL) specifies whether a user from the specific source is used to access a machine with the respective protocol. So, to do Access Control, the source and destination IP addresses along with the Protocol are used and verified for the permission or denial of permission. The updation of the ACL can be done by the administrator as it requires password. After Access Control check, the packet is checked for next hop using the routing information. Then, the packet is sent to the next hop using socket application.

Upon the arrival of the packet, the router processes the packet and identifies any processing to be done. If there is any processing, the processing is done at the router. The main advantage here is that the process currently existing can do both PVM oriented parallel job as well as Matlab applications. Thus a wide variety of applications can be done at the router itself. The processing done is mainly based on the content of the packet. The packet may also contain certain codes that are copied from the source. These source codes could also be processed at the router level. The codes that are executed at the router level can be C or Matlab programs or both. The processing of C from Java is done using Java Native Interface.

The classes that were developed for user interface and for the interaction between users and the system are listed in **Table 1**.

Classes Name	Function
AllFunctions	A Glue for other Graphical User Interface [GUI] Classes
Password	Security access to ACL
ACLCorrect	Access Limitation
ACLEntry	Entry data to ACL
ColumnLayout	Divide GUI into Sender and Receiver Part
Prompt	Java Native Interface
Java_Prompt_getLine	Java Native Interface

Table 1. Main User Interface Classes

FANS user interface consists of sender part and receiver part. The initial appearance of the user interface is depicted in **Figure 11**.

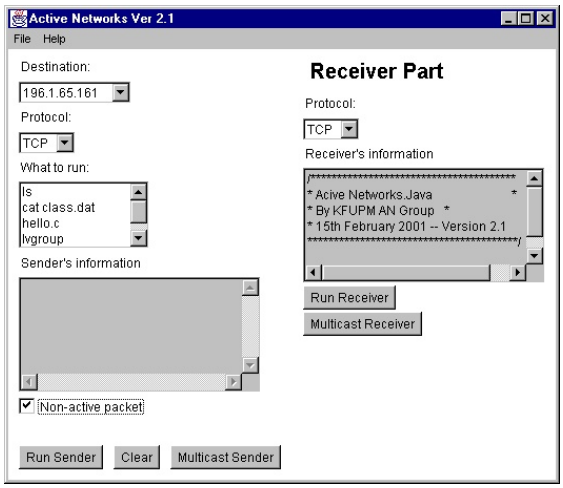

Figure 11. FANS User Interface

The active scenario is designed as follows. A user in Computer A wants to do some computation on some data by using an application (for example, Matlab) which does not exist in his machine. The user wants to send the result of the application to another computer B. In addition, the machine A has neither room to run that application nor to store the result. However, there is a similar application stored in Router, and there is enough room to store the result in Computer B. To accomplish the task, A sends an active packet to Router to activate PVM. The PVM then sets up a virtual connection between A and B. The Matlab runs parallely in router and B. The result is then stored in B. After the result is obtained and stored, the PVM is then deactivated and all the virtual connections are released.

7. Advantages

From our scenario and implementation, it is clear that the combination of Active technology and Parallel processing offers several advantages:

1. It makes use of the resources in a LAN system optimally in a distributed manner.
2. With the help of active technology, the machine that has no room to run an application can do so by sending active packet, with an identifier to the requested application, to one of the other machines that has that application and has also enough memory to run the application. In addition, it is even possible to save the result to another machine with huge memory capacity.
3. The application can run faster as it is processed parallely.
4. The application code can be injected and run in the network only if necessary and therefore efficiently use the resources.
5. Many applications may run simultaneously given that new application may be released on-the-fly.
6. Congestion control can be done in router itself since the router can congestion control routines.

8. Conclusion and Further Work

This paper shows a sample application of the generic active technology. We have also pointed out the advantages offered by the combination of active network and parallel processing in terms of performance and resource optimisation.

The future work is to address the security issues in both technologies. Both technologies are very fragile from malicious access and attack. Therefore we need to face this issues carefully and thoroughly. We further intend to make the system useful not only for programmer or expert users but also for the novice users, by having both applet and application versions.

9. References

[Cal99] Calvert, K.L, Architectural Framework for Active Network version 1.0, Active Network Working Group RFC, University of Kentucky, July 27, 1999

[Bha96] Bhattacharjee, S; Calvert, K.L.; Zegura, E.W., Implementation of an Active Networking Architecture, Technical Report, Georgia Institute of Technology, July 1996

[Bha97] Bhattacharjee, S; Calvert, K.L.; Zegura, E.W., An Architecture for Active Networking, Proc. IEEE INFOCOM '97, April 1997

[Wet98] Wetherall, David J; Guttag, J. V.; Tennenhouse, David L, ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols, Proc. IEEE OPENARCH '98, April 1998

[Sch98] Schwartz, Beverly; Zhou, Wen Yi; Jackson, Alden.W, SmartPackets for Active Networks, BBN Technologies, Jan. 1998

[Ale97] Alexander, Scott D., *et al.*, Active Network Encapsulation Protocol(ANEP), RFC Draft, Juli, 1997

[Ban97] Banch, Albert; *et al*, Multicasting Multimedia Stream with Active Networks, ICSI Technical Report 97-050