Radial Networks Reconfiguration Using Active Packets

Akbar[1], Buhari[1], Sahalu[2], Saleem[1]
[1]Lecturer, ICS Dept., King Fahd University of Petroleum and Minerals.
[2]Asst. Professor, ICS Dept., King Fahd University of Petroleum and Minerals.
31261 Dhahran, Saudi Arabia.
Phone: 966 3 860 4374
Fax: 966 3 860 2174
Email: mibuhari@ccse.kfupm.edu.sa
Published in PDPTA'2001

Abstract

Present day computer networks have become the mode not only for data transfer but also for multimedia information transfer. Distance learning, electronic publishing houses, e-Businesses and similar applications that require efficient transmission of multimedia data in real-time are now popular. To achieve such fast real-time transmission requires a network with minimal congestion and packet collision overheads. Ensuring these properties, in turn, requires the use of switching techniques which require the computationally expensive process of identifying the appropriate switching options for packets transmission.
In this paper we propose the use of active networking technology to efficiently reconfigure radial networks in order to fully utilize their bandwidth and capacity. Active network's on-the-fly processing is used to support an effective switching search to selected feasible switching combinations. Preliminary OPNET-based simulation of our system indicates about 20% performance improvements in both throughput and utilization of the network.

1. Introduction

Active technologies have been emerging in the fields of operating systems and programming languages for over ten years. Early work tended to address only one of three important issues – mobility, efficiency or safety. PostScript is an example of an early effort that stressed mobility over safety. Applications generate mobile programs that are executed at printers, which may be shared and distributed about a network [Dal96]. Active Networking (AN) refers to the addition of user-controllable computing capabilities to data networks [Sma96].

One of the key strengths of the Internet service model is that it abstracts away the details of how messages are forwarded through the network. On the one hand, this design principle is extremely powerful because it divorces applications from much of the complexity of the underlying communication system, but at the same time, it constraints applications because they cannot exploit detailed knowledge of the underlying network to enhance their performance [Ela98]. A crucial capability of active services is flexible and simple deployment of application-level computation within the network [Ela98].

In order to reconfigure a network, a dynamic reallocation policy might be considered [Xia88]. The policy requires:

- Assessment of the system behavior in order to determine the need for the next reallocation.
- Identification of the optimal assignment.
- Reorganization of the database files.

Since traffic variations will more likely take place over larger periods of time, reconfiguration is expected to be a relatively infrequent event, and each assignment of wavelengths will be sufficient enough for the scheduling of packet transmissions by the media access protocol [Ili96].

In this paper, the current state of art is discussed in Section 2. We discuss about the various issues that ought to be considered for improving the network performance in section 3. Section 4, discusses

Proposed model and its inner details. Section 5 provides the results for the test bed and finally, the conclusion.

2. State of Art

The various works done in the past have various sets of observations on network reconfiguration. These observations range from verification of single cache benefits to detailed analytical comparison of popular algorithms. A concise summary of the relevant works follows [Jos96]:

- Clark, *et al.*, wrote the seminal paper on packet processing overheads. In their discussion of TCP demultiplexing, they first suggested that, if a pointer that directly addresses the last TCB (Transmission Control Block) referenced is cached, the result is a substantial performance benefit. They further suggested that, if this optimization failed to be useful, a natural extension of this method is to hash to one of a set of TCB lists with the most recently sorted first.

- McKenney and Dove compared various protocol control block (PCB) lookup schemes using an analytical approach. They were interested in performance of TCP lookups on on-line transaction processing (OLTP) systems. Such systems characteristically have large aggregate-packet-rates, large numbers of connections, and predominantly small packets. They prove analytically that an algorithm that combines caching with multiple hash chains is "better" than other well-known alternatives. Their comparison results were in terms of expected number of PCBs searched.

- Mogul examined temporal locality of reference (i.e., referencing of PCBs) as packets depart from and arrive at a network computer. He performed experiments that traced locality at the process level using fine-grained time resolution and concluded that packets to a host usually arrive for the process that most recently sent a packet, and often with little intervening delay. He also verified the observations of Clark concerning processes that receive arriving packets.

- Kay and Pasquale measured TCP and UDP packet processing overhead for various computer architectures. Although they focus on non-data touching overhead for systems that send and receive many small packets, they provide a detailed measurement of packet multiplexing and demultiplexing. They use customized hardware and software mechanisms to measure actual processing time (in microseconds). They acknowledge that their results are not consistent with previous findings; their single-user test environment only had a single connection.

- Partridge and Pink attempted to optimize the UDP implementation in BSD4.3 Tahoe Release UNIX, which does not use a cache as arriving packets are demultiplexed. They incorporated the Reno release single-cache implementation and found that the cache had no effect on UDP demultiplexing. Further modifications to UDP packet processing showed that the use of a cache at the sender provided the greatest performance improvement.

3. Network performance issues

To measure the performance of the network, the main constraint is information retention, which means the percentage of packets that are received, verified, and successfully processed, which depends upon the following parameters [Phy98].

a. Total Bandwidth: The sum of the individual bandwidths of concurrent processes. It signifies the effective bandwidth usage for the set of all participating hosts.
b. Bandwidth/Pair of processes: The average bandwidth each individual pair of processes is able to use.
c. Time/iteration: The average time it takes to complete one iteration of roundtrip.

A 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 [Dan99].

3.1. Bottleneck Bandwidth

There is always a limit on the rate at which data can be transferred through the network. The fastest transfer rate that the path can sustain is called the bottleneck bandwidth [Ver97]. Knowing this limitation, the proper reconfiguration or assigning of packets to a queue can be done.

One key property in an end-to-end Internet route is its stability: do routes change often, or are they stable over time? Depending upon that reconfiguration may be necessary. One aspect is “prevalence” which means how likely it is to observe a particular route.

4. Our proposed model

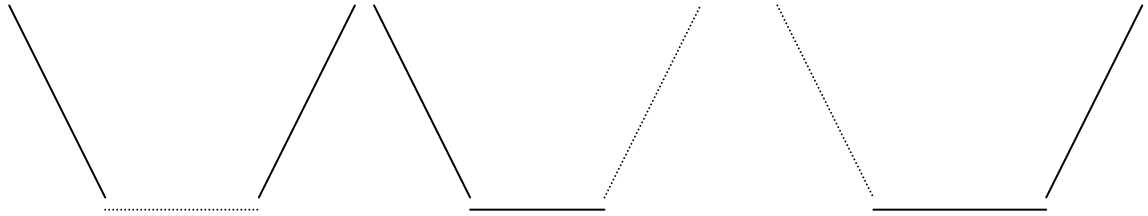
To achieve maximum throughput in a radial network system, the aim is to identify the appropriate switching options. In the proposed method, all the laterals of the network system are considered simultaneously, instead of determining the switching options on a loop by loop basis. Usually, this type of technique involves very complicated mathematical techniques and large computational time due to the combinatorial nature of the problem. However, the solutions obtained by these methods achieve a global optimum of network congestion and collision. The proposed method develops a generalized simultaneous switching algorithm, which minimizes the computational time by performing the effective search to selected feasible switching combinations.

4.1. Determination of the number of combinations

In general, any tie or/and sectionalizing switches can be closed or/and opened to perform various network reconfigurations without creating any closed loop or leaving any branch unconnected. Any reconfiguration which forms a closed loop, or leaving one or more branches unconnected, is classified as an infeasible switching combination for network configuration. To avoid any infeasible switching combinations, the connectivity from the source to all the nodes of the system is checked. If a valid path exists, then the configuration is a feasible one, otherwise it is infeasible. It is important to note that every tie switch or sectionalizing switch selected for closing or opening for reconfiguration is considered together with its two immediate neighboring switches as indicated in Figure 1. In the figure, the middle one is a tie branch and the other two are the neighboring branches of the tie branch. If a tie switch is closed, then a closed loop will be formed. To avoid a closed loop and restore the radial structure, one of the neighboring sectionalizing switches must be opened. The three possible switching combinations are shown in Figure 2 and only these three switches form a switch-group. If a sectionalizing switch is opened during reconfiguration, this open branch and its adjoining branches again form a switch group. In this way, all branches can form switch groups of their own, recursively. This recursion is stopped when a node with three or more branches is encountered. If this recursion is not stopped at some point, then the total number of combinations to be investigated will become combinatorially explosive, which results in NP completeness.

Figure 1: Three branches forming a switch-group.

The switching options in a switch-group form a trinary system of logic. That is, a switch-group can form three combinations with one of them only open for each combination. The three combinations can be named as Group-0, Group-1, Group-2, where, Group-0 represents the configuration at which the middle switch is open, Group-1 in which the right branch is open and Group-2 is the one in which the left branch is open as shown in Figure 2.



Group-0 (zero-connect) Group-1 (Left-connect) Group-2 (Right-connect)

Figure 2. Three possible combinations of a switch-group

Now, let us consider all the switch-groups in a system. If there are *n* tie switches in a system, there will be *n* switch-groups forming 3^n different switching combinations. Each switch position in a switching combination is found out by using the formula,

$$SC_i = [\, SP_{ij}, \ldots SP_{i2}, SP_{i1}]$$

The switch positions of the j^{th} switch in i^{th} combination, SP_{ij} can be obtained as,

$$SP_{ij} = [(i-1)/3(j-1)] \bmod 3$$

Where,

i: combination number (one of 3^n)

j: switch number (one of n) and

SC_i: positions of the various switches in the i^{th} switching combinations.

For each switching combination, the status of each switch in a switch-group is realized. If the status is 0, then the tie switch is open; if it is 1, the right side neighboring branch is opened and the tie switch is closed. The 0,1 and 2-status belong to Group-0, 1 and 2 respectively, and also named as zero-connect, left-connect and right-connect. For zero-connect, the configuration will remain unchanged, but for left-connect or right-connect, the configuration will change according to the left or right-connect logic. Left and right-connect logic are developed using the data flow in the adjoining branches of the tie-branch in the system. When a system is reconfigured by closing the tie-branch and opening either of the adjoining branches, the data flow in the open-branch which was flowing before reconfiguration, shifts to the newly connected tie-branch. If the left-connected is performed, then the data flow of the right branch of the tie-switch will be shifted to the tie-branch. Similarly, when the right-connect is performed, the data flow of the left branch of the tie-branch will be shifted to the tie-branch. Mathematically, this data flow transfer can be represented as,

Left-connect: $D_{jk} = D_{lk}$; D_{lk} = NULL

Right-connect: $D_{kj} = D_{ij}$; D_{ij} = NULL

The newly assigned values as shown in Figure 3 are used to calculate the data flow in the branches and to check the connectivity for identifying the feasible and infeasible combinations.

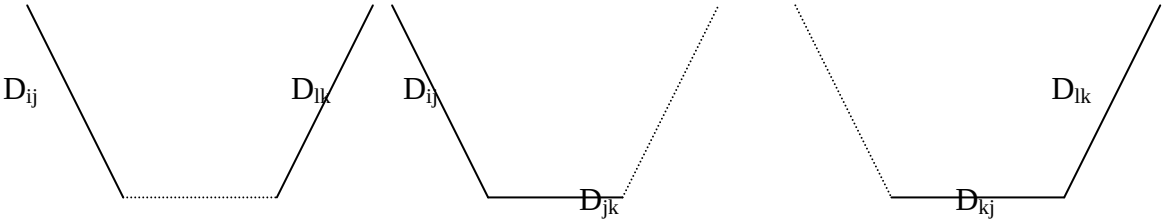


Figure 3: Data flow in three possible combinations of a switch group

4.2. Determination of Feasible and Infeasible Combinations

A network system after being reconfigured may become infeasible because there can be some loads (hosts) which will be isolated from the server. In load minimization technique, there will be certain infeasible combinations out of restructuring among the 3^n different switching combinations possible. When a branch is left unconnected due to the opening of the sectionalizing switches or a closed loop is created due to the closing of the tie-switches, infeasible combinations result. These infeasible combinations must be identified and eliminated from the total combinations. The infeasible combinations are determined by checking the connectivity of each branch in the network using the check_connect logic as described below.

Connectivity of a node or branch means the existence of physical connections of a node or branch from the source. For each combination among all the nodes, select each node one after another, check its connectivity and make its connectivity active by setting a flag if it has the feasible connectivity. This procedure is repeated for all the nodes in the network. The above steps can be represented by the following recursive procedure:

```
i = 1
CC(i)
{
 con[i] = 1;
TC[i] : CC(forward[i]);
}
if Pij > 0 then
  forward [i] = j; reverse[j] = i;
if Pij <0 then
  forward[j] =i; reverse[i] =j;
```

Where,

CC( ) : connectivity check function

TC[i] : total numbers of connection for i^{th} node

Con[i] : the connectivity of node i

D_{ij} : the data flow from i^{th} node to j^{th} node

i: the sending node and

j: the receiving node.

To find all the connections of the node i, the following procedure is used.

IF forward[i] = x and reverse[j] = y then

connect[i] = x and y; or x and i; or i and y.

Where, connect[i] is the connectivity of node *i* under consideration in an n-bus network. The connectivity check for the whole system is done as follows,

Con[i] = 1; D_i != NULL

Infeasible | con[i] = 0; i = 1 … n

Check whether the connectivity flags of all the nodes of that particular configuration are active; otherwise declare the combination as infeasible.

4.3. Finding Execution time minimization for any switching configuration of a radial network

The network reconfiguration for execution time minimization is performed by opening and/or closing the sectionalizing- and tie-switches in such a way that the radiality of the network is retained and at the same time execution times are minimized. In the proposed method, a consolidated algorithm has been developed to take into account automatically, all types of operations needed for (i) finding the total combinations of the system using the trinary logic, (ii) obtaining the feasible combinations, (iii) determining the switching status of the individual switches, (iv) estimating the times for all the switching combinations, (v) identifying the minimum time configuration which might be a local or global minimum from among the feasible switching combinations, and (vi) finally making a search to find the global optimum for execution time minimization by moving to the left or right of each of the open lines or tie lines, depending upon whether the right adjoining or the left adjoining branch of a tie-switch is open.

Once the input parameters of the base configuration, or any other configuration of a radial system under consideration, are fed to the algorithm, the algorithm automatically reassigns the branches and loads for any other switching configuration. In the input data, apart from specifying the branches, branch capacity, and tie-branches, the left and right neighboring branches to each of the tie branches and the node numbers on both sides of the tie branches indicating the extent of the search to be carried out, onto the right and left sides of the tie branches, have to be specified. The switching combinations are formed using the trinary logic concept and the status of each switch in a switch-group of various switching combinations is realized. The infeasible combinations are omitted from the total combinations using the check connect logic. These switching combinations are formed according to the switching status found from the various switching combinations. The execution time is calculated for each branch of the switching configuration under consideration, and the total execution time is estimated. Likewise, the execution time in each of the remaining configurations is calculated. The configuration that gives the minimum execution time is identified. The configuration with the minimum execution time is called the best configuration among the feasible configurations or combinations investigated. To further narrow down the search towards the global minimum, a simplified extension of the above procedure is carried out by considering the connectivity of the best-combination found above. In a reconfigured system, obtained from the base configuration, the open sectionalizing or tie switches are considered as the tie switches for the next operation for network reconfiguration. The switch-groups obtained from the best-combination are examined individually, and the switching status of each switch-group is found. If the connectivity of the switch-group in the best-combination remains in the zero switch group (zero-connect), then the switch group is unaltered. If the connectivity of a switch group in the best combination is towards the left (left-connect), the connection is made to move towards right by opening the next neighbour of the right branch of the current tie switch and closing the current tie switch forming a new group. If the connectivity of the switch group in the best-combination is towards the right (right-connect), the connection is made to move towards left by opening the next neighbour of the left branch of the current tie-switch and closing the current tie-switch forming a new switch-group. For a newly formed combination, the execution time is calculated and compared with the execution time in the best-combination estimated earlier. If it is less than that of the best combination, then the current combination is considered as the best combination and the search is further carried out in the same direction; otherwise, the best combination remains the same. The above procedure is applied to other switch-groups in succession and the best combination is found.

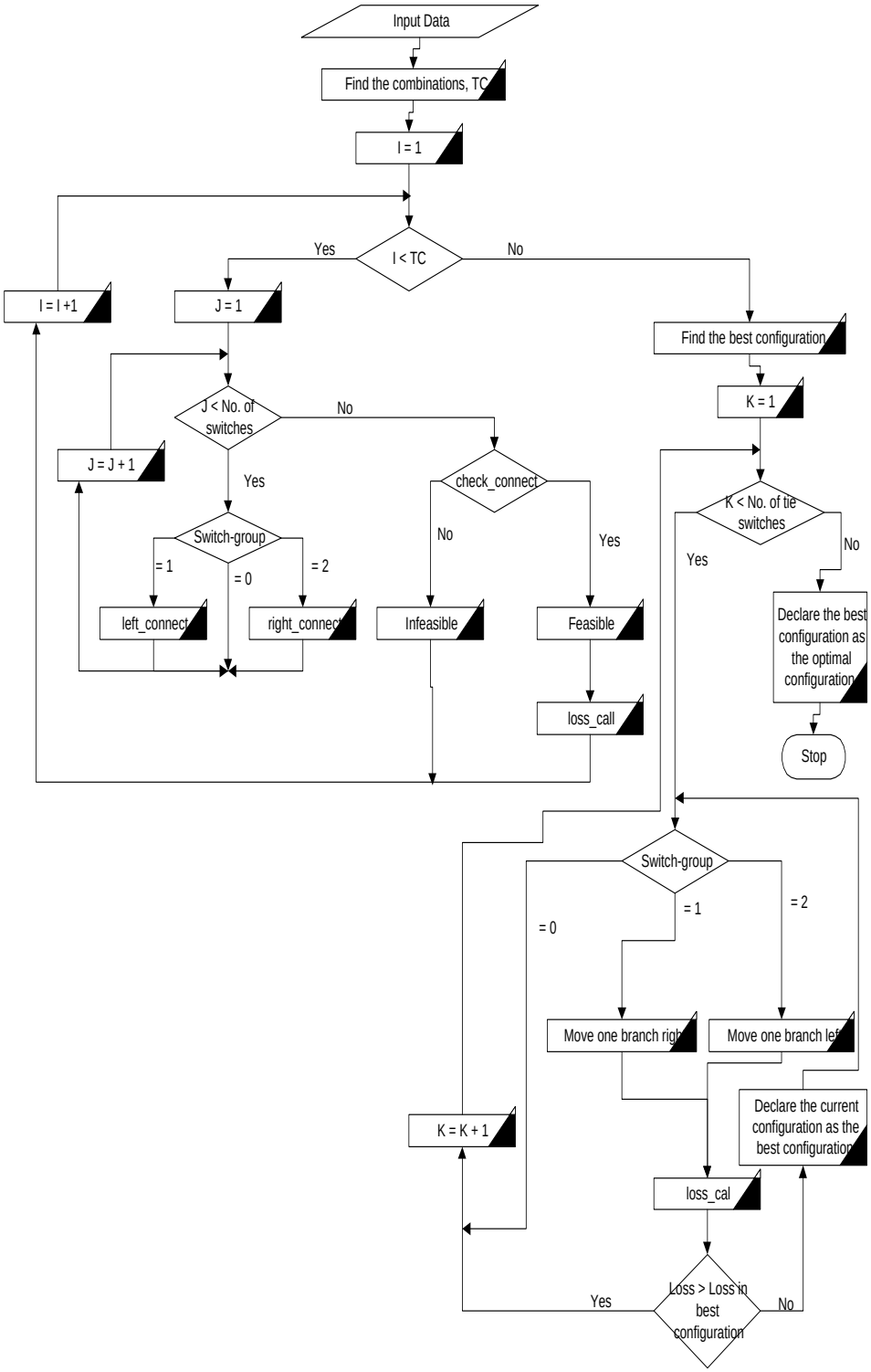


Figure 4. Flow-chart of the Proposed Model

Cell loss rate is given as a function of the traffic load offered to the network. Furthermore, average buffer overflow duration is given for various traffic loads, and it is shown that the buffer overflow duration gets larger when the load is increased. It is also shown that reducing the peak traffic rate by smoothing helps reduce both the cell loss rate and the average buffer overflow duration [Ism96].

4.4. Formula for the calculation of execution time:

Execution time for each branch is calculated using:

unit_time[i][j][k] * nooftasks[i][j][k] + 2.0 * distance[j][k] *
(totaltasks - nooftasks[i][j][k]) * nooftasks[i][j][k] ;

unit_time[i][j][k]	: Unit time for an operation (i^{th} server, j^{th} and k^{th} node).
nooftasks[i][j][k]	: Number of tasks (i^{th} server, j^{th} and k^{th} node).
distance[j][k]	: Distance between j and k node.
totaltasks	: Total number of tasks.

Normally, the major problem faced by any current day system is to identify the unit time for an operation. In the case of the active networks, the calculation of the unit time for operation, can be done based on the operation at one router. This information can be passed along with the capsule. For each router, the execution time is calculated with reference to previous router's operation speed and with the help of the Management Information Base that posses the information about the hardware criteria of the routers.

5. Results and discussion

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 5** 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 made use of one of OPNET's packets generation algorithms that randomly generates packets. We implemented a static routing algorithm that fixes the next router through which packets can be forwarded from the current one.

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. OPNET defines these statistics 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 to date of an available channel bandwidth, where a value of 100.0 would indicate full usage.*

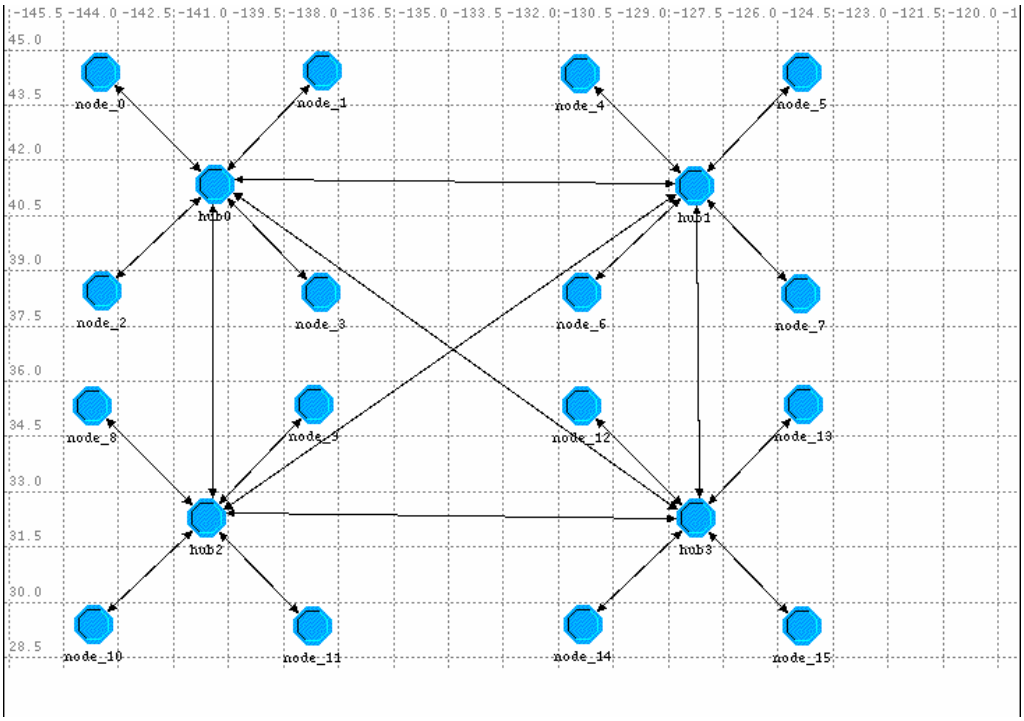


Figure 5: Our Network Architecture.

Figures 6 and 7 respectively depict results of simulating our network's throughput and utilization. Each figure has two component graphs; the graph in the upper part shows the result obtained when the active version of the network is simulated and the lower 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.
The horizontal axes in both figures represent the simulation time in seconds. The vertical axis in Figure 6 represents the average number of packets transmitted per second. The vertical axis in Figure 7 represents the percentage utilization of the network channels.

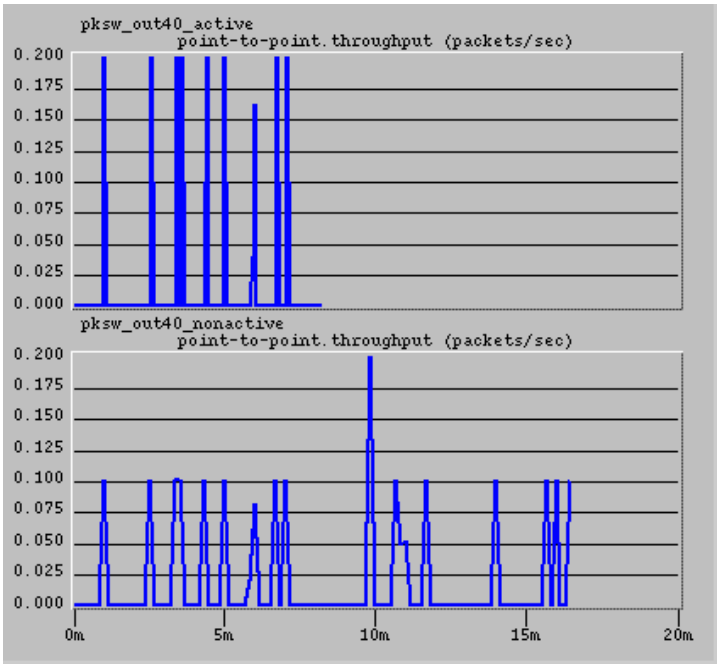


Figure 6: Showing Comparative Throughput of the Active and Non-active Systems.

Looking at these figures closely, we see that the horizontal axes in both cases indicate that in the active network case the experiments finish in about half the simulation time compared to the experiment completion time of the non-active network. This implies that the active network model has a performance improvement of about 50% over the non-active network model. Looking at the vertical axes in these figures, we see also that the throughput and utilization in the active network model double those in the non-active model throughout the program execution time. A small exception is the throughput value in the non-active system at about the 10^{th} second in the execution time when there was a big hiccup of throughput. We quickly note that this result should be taken with a pinch of salt and we explain why in the next section.

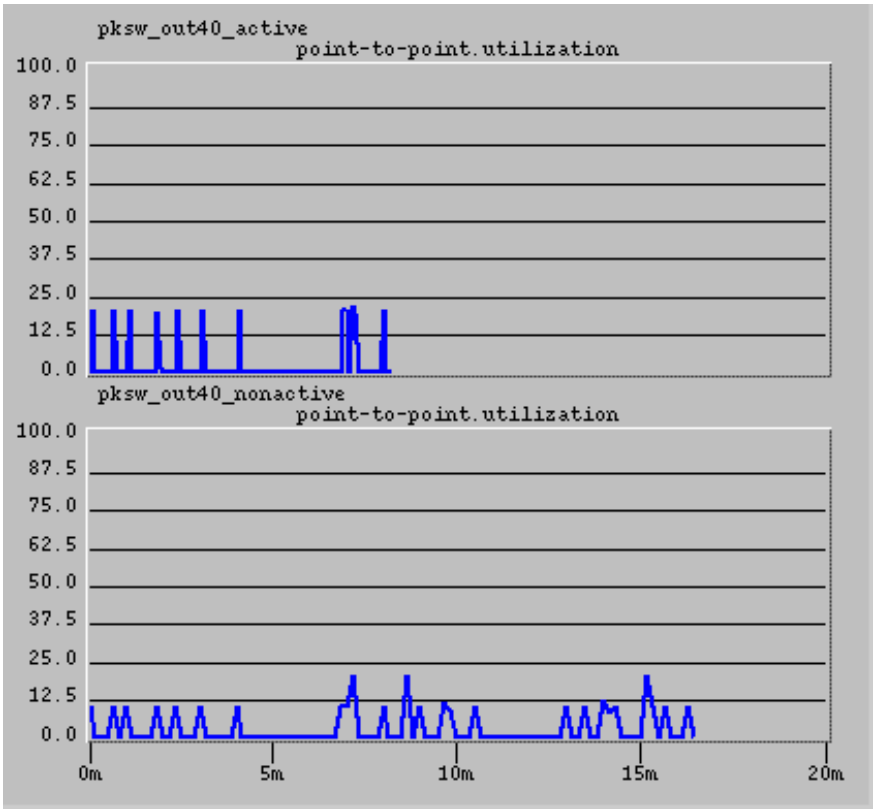


Figure 7: Showing Comparative Utilization of the Active and Non-active Systems.

Limitations of our experiments.

As we mentioned earlier our simulations are idealized in the sense that, mainly, we do not take into account communication cost. Considering the fact that the routers in our active networks now take active part in the computation process, the cost of communication when considered is envisaged to high. There is also the cost of data storage in the routers with the added cost associated with a distributed system since our active network is a distributed system with the nodes serving as computing agents. Putting all costs into account, we are confident that the performance of our system reported above will stand between 80-90% giving a performance improvement of up to 20% over the traditional non-active system.

6. Conclusion

In this paper we presented a novel approach that can be used to achieve fast multimedia data transmission in radial networks with the help of on-the-fly computation supported by routers in active networks. The model relies on the use of these dynamic computations to reconfigure the networks by helping to perform the combinatorial, computationally expensive process of identifying the appropriate switching options for packets transmission. Doing this will facilitate the reduction (or elimination) of the congestion rate and packet collision problems in the network. The net effect of these will be the exploitation of the maximum throughput and optimal utilization of the network.
Preliminary idealized OPNET simulations of a 16-node active network system reveal good potential performance improvement over traditional non-active networks. Although communication issues are not taken into account in the simulations, we believe that the performance improvement after costing all issues will be no less than 20% over traditional systems. We are in the process of implementing a real system that can be used to corroborate this hypothesis.

7. Reference

1. [Dal96] David L. Tannenhouse and David J. Wetherall, “Towards an Active Network Architecture”, Computer Communication Review, Vol. 26, No. 2, April 1996.

2. [Sma96] Samrat Bhattacharjee, “An Architecture for Active Networking”, Technical Report GIT-CC-96/20, College of Computing, Georgia Tech.
3. [Phy98] Phyllis A. Schneck & Karsten Schwan (1998). Dynamic Authentication for High-Performance Networked Applications (Tech. Rep. Grant No. DAAH04-96-1-0209). Atlanta: Georgia Institute of Technology, College of Computing.
4. [Ela98] Elan Amir, Steven McCanne, and Randy Katz, “An Active Service Framework and its Application to Real-time Multimedia Transcoding”.
5. [Dan99] Dan S. Decasper, Bernhard Plattner, Guru M. Parulkar, Sumi Choi, John D. DeHart, and Tilman Wolf, “A Scalable High-Performance Active Network Node”, IEEE Network, May/June 1999, pp. 8-19.
6. [Jos96] Joseph T. Dixon & Kenneth Calvert (1996). Increasing Demultiplexing Efficiency in TCP/IP Network Servers (Tech. Rep. No. GIT-CC-96-08). Atlanta: Georgia Institute of Technology, Networking and Telecommunications Group.
7. [Ver97] Vern Paxson (1997). Measurements and Analysis of End-to-End Internet Dynamics. Unpublished doctoral dissertation, University of California, Berkeley.
8. [Xia88] Xiaolin Du & Fred J. Maryanski (1988). Data Reorganization in a Dynamically Reconfigurable Environment. Eighth International Conference on Distributed Computing Systems ‘98, 463-470.
9. [Ili96] Ilia Baldine & George N. Rouskas (1996). Reconfiguration in Rapidly Tunable Transmitter, Slowly Tunable Receiver Single-Hop WDM Networks (Tech. Rep. No. TR96-10). Raleigh: North Carolina State University, Department of Computer Science.