Local Area Network Traffic Locality: Characteristics and Application

A Thesis

Submitted to the College of Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Department of Computational Science

University of Saskatchewan

Saskatoon, Saskatchewan

by

Neeraj Gulati

July, 1992

Distribution Notice

Local Area Network Traffic Locality: Characteristics and Application

Candidate: Neeraj Gulati

Supervisors: Richard B. Bunt and Carey L. Williamson

July 1992

ABSTRACT

Local area networks (LANs) are a popular means for connecting autonomous workstations together in a computational environment. LANs offer several advantages over traditional centralized computer systems, including better reliability, scalability, and cost. Certain limitations of LANs like their limited geographical span and the upper bound on the number of hosts that can be connected by a single LAN, can be overcome by interconnecting LANs together into an extended LAN or internetwork. Interconnection devices, such as bridges or routers, forward packets as necessary between the LANs, creating the illusion of a single large network.

The performance of network interconnection devices is key to the success of extended LANs. Interconnection devices must selectively forward packets between LANs with minimal delay, so as to transparently extend the LAN. As networks increase in size and bandwidth, the need for efficient network interconnection devices increases.

One of the goals of this thesis is the characterization of LAN traffic locality, i.e., the non-random distribution of source and destination addresses in the packets seen on a local area network. Five measures of network locality are proposed: persistence, concentration, reference density, address re-use, and host-pair locality. These measures are applied to packet traces captured from an operational LAN to determine the extent to which locality of reference is present in LAN packet traffic.

The other goal of the thesis is to determine the extent to which the locality present in network traffic could be used to improve the performance of a data link level interconnection device. The locality characteristics observed suggest a caching approach for the design of interconnection devices. Trace-driven simulation is used to analyze the benefit that can be achieved with this design. Results show that a substantial reduction in address lookup time at interconnection devices can be achieved by exploiting the locality behaviour of the traffic.

Acknowledgements

I am greatly thankful to Dr. Richard Bunt and Dr. Carey Williamson who supervised this research, for their guidance and support. Working with them was a great learning experience. I would also like to thank Dr. Derek Eager, Dr. Mostafa Abd-El-Barr and Dr. Hugh Wood, Electrical Engineering, for serving on my thesis committee and providing valuable comments.

Financial assistance received from the College of Graduate Studies and Research is appreciated.

I want to thank Dr. Jim Greer and Dr. Carl McCrosky for their guidance during the initial stages of my Master's program. Thanks to Utpal and Azam for helping out a fellow graduate student, with a fractured right arm, with all those pre- and post-graduation chores. I also thank DISCUS members Darryl, Karen, Sundaram, Elina and Heng for their comments and feedback in the group meetings. I would like to express my gratitude to Gail and Maureen for their cooperation.

I dedicate this work to my parents Mr. Sukh Dayal Gulati and Mrs. Kailash Gulati. The values bestowed in me by these two wonderful people have made it possible for me to see this day. A special thanks goes to my brother Pankaj and his wife Deepali for their support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the first two decades of their existence, computer systems were highly centralized, usually located within a single room. The users brought their work to this "computer centre", where a single large system was used by all the users. This model was inadequate for organizations that had sites spread over a wide geographical area and needed the capability to gather information from these sites, to process the information, and to distribute it. The merging of computers and communications has provided the environment suited to the sophisticated needs of such organizations. The merger of these technologies has had a profound influence on the way computer systems are organized and the old model of a single computer serving all of the organization's computational needs, is rapidly being replaced by one employing a large number of separate but interconnected computers. These systems are called *computer networks*.

Computer networks provide an environment in which all programs, data, and equipment available on a network are accessible by anyone on the network regardless of the physical location of the resource or the user. Such sharing of resources results in a reduction in cost and an increase in utilization. By having alternate sources of supply, e.g., the presence of multiple CPUs and replicated files, system reliability is improved. Furthermore, small computers have a much better price/performance ratio than large mainframes. For instance, mainframes are roughly a factor of ten faster than the fastest single chip microprocessors, but they cost a thousand times more. This imbalance makes networks of powerful personal computers more cost effective than large mainframes. Networks also offer an excellent communication medium that helps promote collaborative work and provide a means for fast exchange of information between users located far apart.

To reduce their design complexity, the logical communication model of most computer networks is subdivided into a series of *layers* or *levels*, each one built upon its predecessor. The purpose of each layer is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. The International Standards Organization (ISO) proposed a seven-layer model called the OSI (Open Systems Interconnection) Reference Model [37]. The structure of the model is shown in Figure 1.1.

The *physical layer* is concerned with transmitting raw bits over a communication channel. The design issues largely deal with the mechanical, electrical and procedural interfaces, and the physical transmission medium.

The physical layer is subject to channel noise and occasional errors, so the main task of the *data link layer* is to take a raw transmission facility and transform it into a line that appears to the network layer to be free of transmission errors. This involves error detection and error correction. This layer deals with chunks of bits called *frames*. By attaching special bit patterns, the data link layer creates and recognizes frame boundaries.

The *network layer* is responsible for routing of packets, congestion control and addressing the issue of difference in protocols when heterogeneous networks are interconnected.

The *transport layer* is a true source-to-destination or end-to-end layer. It splits

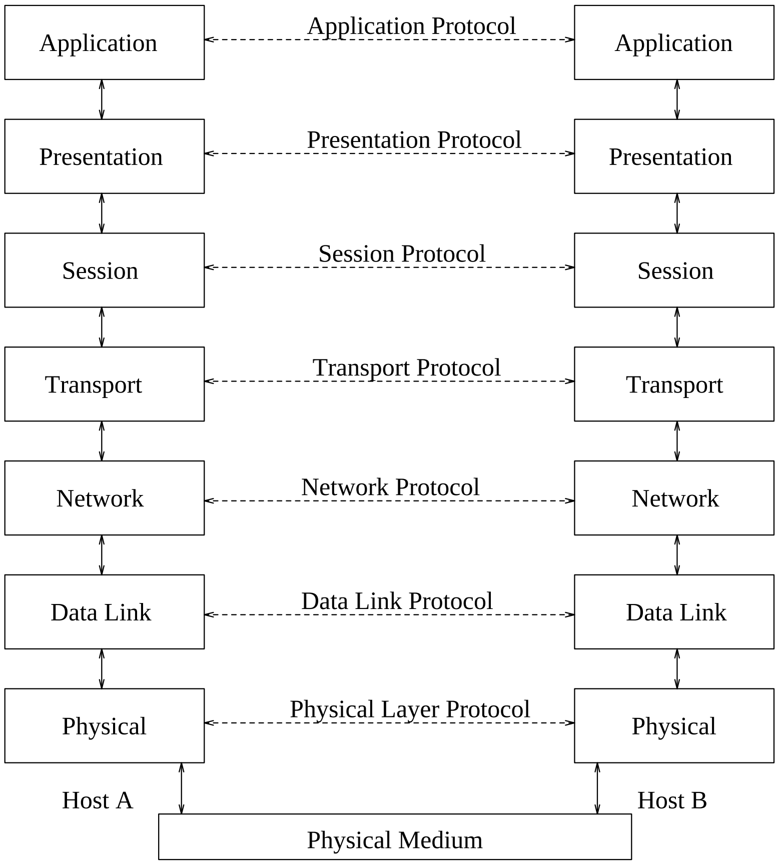

Figure 1.1: The OSI Reference Model

data coming from the session layer into smaller units if need be, passes these to the network layer, and ensures that the pieces all arrive correctly at the other end. Establishing and deleting logical connections across the network are also the responsibilities of the transport layer.

The *session layer* allows sessions to be established between different machines. A session provides services like token management and synchronization, that help in ordinary data transport or in the provision of enhanced services.

The *presentation layer* is concerned with the syntax of the information transmitted. Data conversion may be required in order to make it possible for different computers using different codes for representing character strings or integers etc., to communicate. Data compression and encryption are also performed at this layer.

The *application layer* provides useful network applications to the end user, e.g., electronic mail, remote job entry, remote login, file transfer etc. It also helps make different terminal types compatible for communication.

Within the OSI model, two approaches to network communication have evolved, namely circuit-switched networks and packet-switched networks. In circuit-switched networks, an end-to-end path is physically established before the data is sent. In packet-switched networks, blocks of data that are being transferred are stored and then forwarded by intermediate switching points (hop by hop). Packet switched technology can be divided into three broad categories:

1. Wide Area Networks (WANs)
2. Metropolitan Area Networks (MANs)
3. Local Area Networks (LANs)

These are listed in decreasing order of geographical reach. Since the work presented in this thesis deals with LANs, the following discussion focuses on issues related to LANs only.

LANs have become extremely popular because of their advantages over centralized systems. It is not uncommon in larger organizations for a number of LANs to be installed at various locations in a single complex. The need for sharing of information and resources across these LANs leads to the interest in interconnection of the LANs into a single composite network called an *extended LAN* or an *internetwork*. Such an interconnection also overcomes some of the limitations of LANs, namely their limited geographical reach and the maximum number of hosts that can be connected to a single LAN. By isolating local traffic of each LAN, better performance can be achieved as compared to a single large network. Faults can be isolated by simple manipulation of the interconnection devices. By enforcing access control requirements at the interconnection points, access to certain portions of the internetwork can be restricted, hence providing better security.

One of the important issues related to network interconnection is the design of the interconnection device. Interconnection devices must perform a number of functions. First, the interconnection device is responsible for forwarding packets to the appropriate networks. The forwarding process must take place with minimal delay. Second, when different types of networks are being connected, the interconnection device must ensure that the packets forwarded meet the protocol requirements of the receiving network. This might involve breaking a packet into smaller packets or encapsulating the received packet into a new packet for transmission. Third, the interconnection device must adapt to the topology changes in the internetwork. Finally, having various LANs interconnected increases the possibility of unauthorized attempts to access or manipulate information. The interconnection devices should be designed to thwart such attempts. Only the forwarding function of interconnection devices is considered in this thesis.

Two of the most popular technologies that have emerged to achieve network interconnection are:

1. Data link layer devices (also called *bridges*)
2. Network layer devices (also called *routers* or *gateways*)

Each of these have a set of capabilities and limitations. Since later chapters deal with the performance of a data link layer interconnection device for IEEE 802.3 LANs (a data link layer standard for LANs), this chapter provides an introduction to the operation of these devices.

The primary purpose of the data link layer is to provide error-free communications across the physical link. In addition to the basic framing and data encapsulation functions, the data link layer has a number of responsibilities that affect the internetworking operation [38]. They are:

1. The data link layer must not re-order frames within a stream of transmissions. Thus an interconnection device operating at this level must guarantee proper frame sequence.
2. The data link layer must guarantee delivery of no frame duplicates. Hence the devices operating at this level are prohibited from generating duplicate frames.
3. A data link layer is expected to deliver frames within a short period. Although an internetwork created at the data link level cannot provide as low a delay as a single LAN segment, some performance guarantees must be made. It is unacceptable for the data link layer to be allowed to deliver a submitted frame an arbitrarily long time after submission; extremely late deliveries may confuse timer-based protocols (protocols which wait some period of time for the acknowledgement to arrive before retransmitting the frame).
4. The data link layer, using an error checksum, can detect physical layer errors with a very high degree of probability. The interconnection device operating at this layer should also perform this duty and discard erroneous frames. Error recovery, in most implementations, is relegated to higher layers of the architecture.

One of the limitations of data link interconnection devices is that they cannot participate in congestion or flow control. There is no mechanism at the data link layer to reduce the sender's data transmission rate to ensure that the receiver does not lose frames due to exhaustion of internal resources (e.g., buffers). So interconnection devices at this layer lack flow control mechanisms which affects their ability to recover from network congestion. Even though these devices may have the intelligence to deduce the source of a congestion problem, there is no entity within the source stations with which to communicate and correct the problem [38].

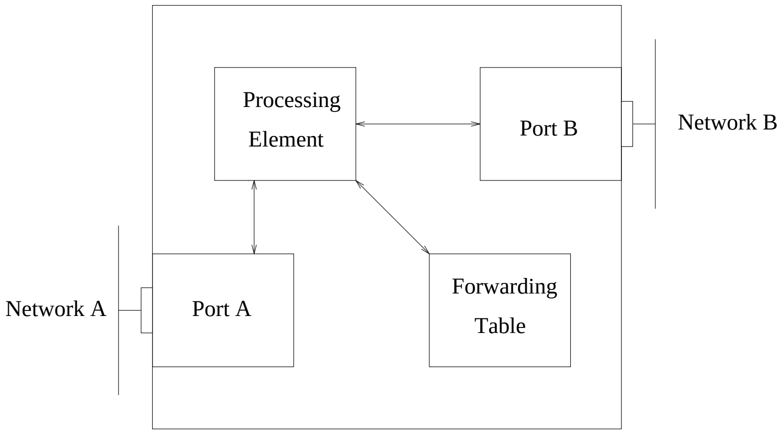

Figure 1.2: Data Link Level Interconnection Device Architecture

The data link level interconnection device is designed to extend a LAN transparently, i.e., the individual hosts on the networks it connects are not aware of its presence. Using only the services of the data link level, such a device forwards frames between LANs, providing simple low-level internetworking. The basic architecture of such a device is shown in Figure 1.2. This device contains a *forwarding table* which has a list of data link addresses along with the associated network port on which that station is accessible, as shown in the forwarding table part of Figure 1.3. There may be more than two ports, depending on the number of networks the device connects. Each port is connected to a broadcast LAN.

The frames to be forwarded by the interconnection device are generated by the applications running on the network. When a data item has to be transmitted, say a 9000 byte file to be transferred from a particular source to a destination, the data is split at each level of the protocol hierarchy, e.g., into two NFS packets, then to seven UDP packets and finally into seven Ethernet frames. The format of an Ethernet frame is shown in Figure 1.4. All the frames generated for this file transfer would have the same source and destination addresses with individual data fields containing parts of the data file. The gap between the beginning of consecutive frames is of the order of 1 to 10 milliseconds on average. The gap between a request and the corresponding

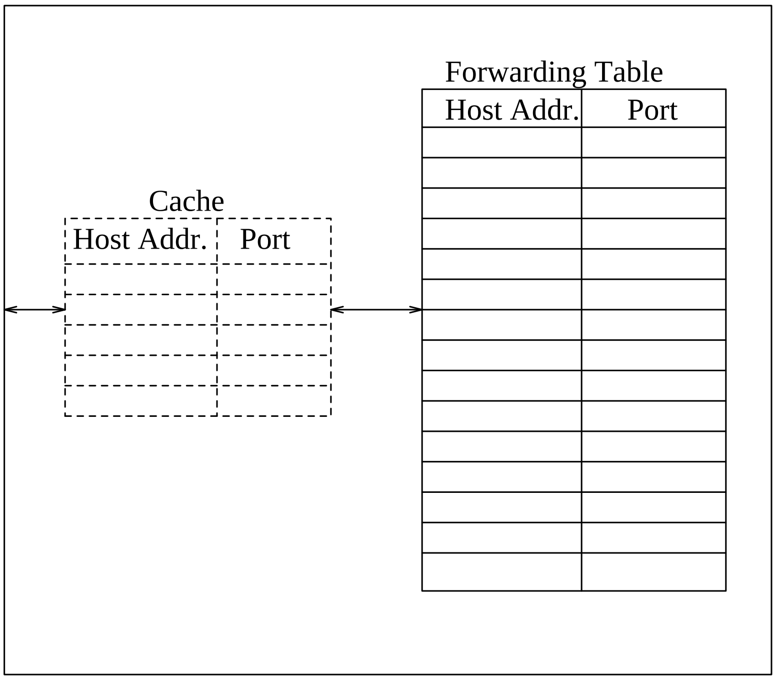

Figure 1.3: Location of the Proposed Forwarding-Table Cache

response is of the order of 30 to 100 milliseconds on average [24]. Transmission time is of the order of a few milliseconds.

When a frame is received by the interconnection device it determines whether this frame has to be forwarded to any of the ports other than the one on which it was received. If the frame is not to be forwarded, i.e., it was addressed to a host on the same network as it was received from, it is discarded, otherwise it is forwarded to the appropriate port of the device. If the device has no information about the location of the host a particular frame is destined to, it simply broadcasts this frame on all ports except the one on which this frame was received. Using the source addresses of incoming frames, the device maintains information about the location of hosts.

The device listens to every frame on every network port to which it is attached. Since it is a store and forward device, it buffers every frame seen on each of the networks it connects. The destination address of the frame (which was received on port, say p_1) is then examined. The forwarding table is searched for the destination

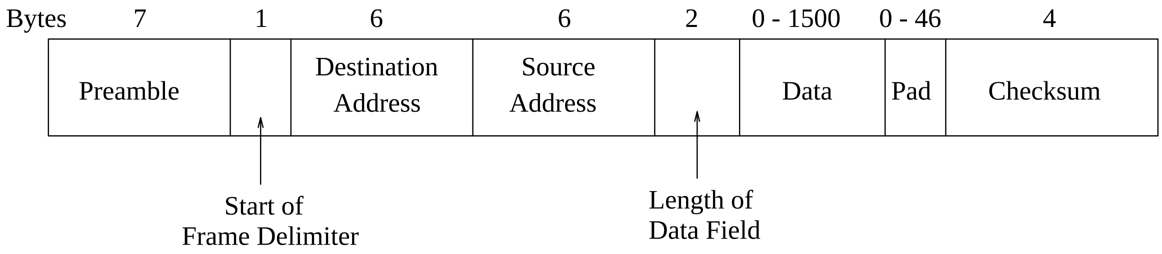

Figure 1.4: IEEE 802.3 Frame Format

address. If there exists an entry for this address, the port address, p_2, is obtained for this address from the forwarding table. There are two possibilities at this stage. Either p_1 and p_2 are the same or they are different. If they are different, the frame is forwarded to port p_2. Otherwise, the frame is discarded (filtered). If no entry is found in the forwarding table, the frame is broadcast to all the ports, except the one it was received on (i.e., p_1). Now the forwarding table is searched for the source address. If an entry exists, the corresponding port address is updated with the address of the port this frame was received on (p_1), otherwise a new entry is created for the source address and the corresponding port address (p_1) is stored along with this entry (this way the device "learns" about the location of hosts).

These devices perform their forwarding function transparently so they must listen, store and process all frames being transmitted on each LAN they connect in order to determine if any of the frames require forwarding. This is referred to as *promiscuous mode* operation and is an important performance consideration in the design of these devices; they must be able to process all frames on all ports, regardless of whether or not the frames require forwarding. This consideration assumes more importance as the networks get larger and faster. The forwarding table contains not only entries for hosts on networks directly connected to it but also entries for hosts on other networks connected to these networks, and so on. So an increase in the size of the internetwork could result in more entries having to be stored in the forwarding table. As the speed of networks increase, the time available to the interconnection device to process each incoming frame decreases. Because of its promiscuous method of operation, the number of potential entries that its forwarding table might have, its inability to perform any congestion or flow control, and its need to cope with the higher bandwidth of networks being developed, a data link level interconnection device must forward frames at the fastest possible rate to maintain its transparent operation. Optimizing the operation of such devices is hence an important issue [20, 21, 23].

Exploiting the traffic characteristics may be one way to optimize the performance of a data link level device. Previous studies of characteristics of network traffic show that packet arrivals on a LAN display a property similar to that shown by memory and file references [20]. It has been found that if a packet is seen going from host A to host B, then there is a high probability that the next packet will be going from host A to host B or from host B to host A. This behaviour is termed "address locality". A reference stream exhibiting address locality favours the usage of a small subset of all operational hosts as their destinations for significant periods of time. If this favoured set of addresses could be made accessible with lower delay in an interconnection device, the lookup time, and hence the forwarding delay, of the device could be reduced. One way of doing this would be to use a small fast memory, a forwarding-table cache (as shown in Figure 1.3), to hold these entries. For deciding the design parameters for the forwarding table cache, i.e., which entries to hold, the number of entries etc., an understanding of the characteristics of the traffic locality is essential.

The goals of this thesis are twofold:

1. To characterize various aspects of traffic locality in an operating LAN environment.

2. To estimate the performance benefits that can be achieved by designing data link level interconnection devices that exploit the locality characteristics of the traffic.

A number of aspects of traffic locality are identified and quantified using a set of traces captured from an operational LAN. For the purpose of determining the performance of a forwarding-table cache in an interconnection device, a trace-driven discrete-event simulation model has been developed. The performance values obtained using this simulation, with a variety of parameters chosen on the basis of the insights provided by the locality characterization, are then used to determine the overall benefit in lookup time that can be achieved by using a forwarding-table cache.

Chapter 2 describes the concepts of locality and caching. Chapter 3 discusses some previous work in the area of network traffic locality. The characterization of LAN traffic locality is discussed in Chapter 4. Cache performance results are discussed in Chapter 5. The major conclusions, contributions and directions for future research are covered in Chapter 6.

Chapter 2

Locality and Caching

Studies of memory and file referencing behaviour have found that such references typically display non-random characteristics. References are distributed non-uniformly in both time and space with small subsets of information favoured over relatively long time intervals. The term "locality of reference" has been used to describe this phenomenon. This property of references has been exploited in a variety of situations. For example, small, high-speed storage devices (called caches) have been used to store favoured subsets so as to make information retrieval more efficient. By applying this property to arrivals on computer networks it has been found that the performance of gateways can be made more efficient [11]. This chapter deals with the basic concepts of locality and caching.

2.1 Locality

The principle of locality was first applied in the study of memory referencing behaviour [9], but was subsequently observed in the usage patterns of other resources, such as file systems [28]. The general principle of locality [10] has been described as:

1. During any interval of time the resource demands of the system are distributed non-uniformly over the total set of available resources.
2. The resource demand pattern for a given resource tends to change slowly with time.
3. The correlation between the immediate past and immediate future demands tends to be high; the correlation between disjoint resource demand patterns tends to zero as the elapsed time between them tends to infinity.

Locality has two aspects, referred to as *temporal locality* and *spatial locality*. Temporal locality means that the information currently in use is likely to be used again in the near future. In the case of page referencing behaviour, for example, references created by the execution of program loops in which both data and instructions are re-used create temporal locality. Spatial locality implies a high probability of references to neighbouring addresses. A sequence of repeated references to the same page exhibits high temporal locality since a page currently in use is re-used frequently in the next few references. A sequence of references to adjacent addresses (as in sequential execution) exhibits high spatial locality since there exists a high probability of use of an address adjacent to the one in use currently. While the definition of neighbouring addresses is somewhat clear for page and file addresses, it is not so clear for networks.

The terms *persistence* and *concentration* have also been used to characterize locality behaviour [6]. Persistence refers to the tendency to repeat a particular reference (or group of references) in succession. This is clearly related to temporal locality. Concentration, on the other hand, refers to the tendency of the references to be limited (concentrated) to a small subset of the total reference space.

Locality characteristics are what make virtual memory systems successful. The basic idea behind virtual memory is that even though the memory requirements of an individual program may exceed the amount of physical memory available for it, the operating system can keep those parts of the program currently in use in main memory, and the rest on the disk. Most processes exhibit locality of reference, meaning that during any phase of execution, the process references only a relatively small fraction of its pages. This behaviour of the references makes the use of virtual memory very efficient.

Processor caches exploit locality by storing information that was used recently by the processor. Information (data or instructions) is fetched into a processor cache, if its use in the future is anticipated based on the locality characteristics of the references. Processor caches reduce the average time spent to obtain the information needed for processing.

File systems also rely on the locality of reference property for efficient operation. Since access to a disk is much slower than access to memory, file systems are designed to reduce the number of disk accesses needed. Studies of file referencing behaviour have shown that there exists considerable locality in such references [28]. Block caches are used to reduce the number of accesses to the disk by keeping in memory a collection of blocks that logically belong on the disk.

The same principles of locality and caching apply to distributed system environments. When a file server and its disk are remotely located, i.e., across a network, the cost of accessing it is still higher. Providing disk caches with the remotely located disk and with the local disks at the workstations helps to reduce the access time.

Locality of reference has been observed in a number of environments and using this behaviour has resulted in considerable improvements in performance. Frames travelling on a network can be treated as a series of references to and from the objects they address. The research presented in this thesis is focussed on the locality of such references in a local area network (LAN) environment, and the exploitation of this behaviour in the design of interconnection devices.

2.2 Caching

Caching is a technique used to reduce mean access time to stored information by putting items that are more likely to be used in a faster (but more expensive) medium. The technique has been applied effectively in a number of areas, including memory caches, disk caches and file caches, and works well when a large frequency of the references are to a small fraction of the information items. In other words, locality in the reference patterns is important to the success of any cache.

Optimizing the design of a cache generally has three aspects [35]:

1. Maximizing the probability of finding a reference's target in the cache (the *hit ratio*).
2. Minimizing the delay due to cache misses.
3. Minimizing the overheads of updating the primary information source, maintaining multicache consistency, etc.

Among the issues involved in cache design are the selection of appropriate fetch policies (what information to put in the cache and when), placement policies (where to put information in the cache), replacement policies (what information to remove from the cache to make room for newly fetched items), primary storage update policies (how to update primary storage when cached items are changed), and cache size.

One of the aims of cache design is to maximize the hit ratio. The simplest and most common cache fetch policy is demand fetching, by which a piece of information is fetched into the cache at the time when it is requested. This obviously fetches the right information, but there is a delay introduced while the information is fetched from primary storage. Demand fetches cannot be avoided entirely, but they can be reduced if it is possible to predict successfully which pieces of information will be needed soon and fetch them in advance. A cache fetch policy which fetches information before it is needed is called a *prefetch policy*.

Information can be stored in a cache in a number of possible ways. Fully associative caches allow incoming information to be stored in any cache location. Because large associative memories are usually very expensive and somewhat slow, caches are generally organized as groups of smaller associative memories so that only one of the associative memories has to be searched to determine whether the desired information is located in the cache. Each such (small) associative memory is called a *set* and the number of elements over which the associative search is conducted is called the *set size*. Such memories are called *set-associative caches*. A *direct-mapped cache* maps incoming information (based on its value) to a certain cache location, and this information has to be stored at that location. The *placement policy* is used to determine where in the cache a piece of information will be placed.

In the steady state, the cache is full and a cache miss implies not only a fetch but also a replacement. Replacement policies are broadly classified into usage-based or non-usage-based or fixed- or variable-space. Usage-based policies take the record of use of an entry in the cache into account when doing replacement, e.g., LRU (least recently used), LFU (least frequently used) and Working Set. Non-usage-based policies make the replacement decision based on some factor other than and not related to usage; FIFO (first in, first out) and Rand (random) are in this latter class. Rand randomly selects an entry from the cache to be replaced. Given a reference trace and a fixed size cache, it has been proven that the MIN policy [3] would cause fewer cache misses than any other replacement policy. When a new entry has to be brought into the cache, MIN chooses for replacement the item that will be referenced farthest in the future than any other item in the cache. Since MIN looks ahead in the reference trace, i.e., requires knowledge about the references that would occur in the future, it cannot be implemented in a real system. It does, however serve as a useful standard for comparison.

When new information arrives, the main storage can be updated in a number of ways. For instance, when a write (store) is performed to main storage, that operation can actually be reflected in the cache and main storage in a number of ways. The cache can receive the write and the main storage can be updated when that line is replaced in the cache. This strategy is known as *copy-back*. Copy-back may also require that a piece of information be fetched if it is absent from the cache, i.e., *fetch-on-write*. Another strategy, known as *write-through*, updates main memory immediately when a write occurs. Write-through may specify that if the information is in the cache, it will be updated in main memory. If the data is not in the cache, it may or may not be fetched. The choice between copy-back and write-through strategies is also influenced by the need to keep the cache memories consistent, in certain environments, e.g., in a shared memory multiprocessor system.

The final parameter is cache size. It is obvious that the larger the cache, the higher should be the probability of finding the needed information in it and thus the higher the hit ratio. Cache sizes cannot be expanded without limit, however, because of cost, access time (the larger the cache, the slower it may become) and, in some cases, physical size limitations.

2.3 Summary

A set of references is said to exhibit locality when the individual references are distributed non-uniformly over time and space and only a small subset of the complete set of information is referenced over significant intervals. The locality property is crucial to the success of virtual memory, disk caches and processor caches. Cache memories are designed to capitalize on the locality behaviour of the references. A number of issues involved in the design of cache memories have been discussed. They are the cache fetch policy, the placement policy, the replacement policy, the primary storage update policy and the cache size.

Chapter 3

Studies of Locality in Computer Networks

Packets on a local area network can be viewed as a series of references to and from the objects they address. The non-uniform distribution of these references over time and space is called *network traffic locality.* The amount of locality in this reference stream may be critical to the efficiency of network implementations [11, 15, 17, 20, 21, 26, 29], if the locality can be exploited through caching or scheduling mechanisms.

This chapter deals with previous work done in the area of computer network locality analysis. These studies have treated network locality with an addressing granularity of networks, individual hosts and individual processes. Four studies are reviewed in this chapter.

3.1 Packet Trains Model

In studies of computer networks it is common to assume that packet arrivals are independent and unpredictable. If it were possible to predict something about future arrivals, better design of packet handling strategies would be possible, or at least implementation of current strategies could be made more effective. The problem of predicting packet arrivals in a computer communication system is analogous to that of predicting memory page references in a paged memory computer. Page references have been found to be correlated such that the probability of a page being referenced decreases as the time from its previous reference increases [10]. This has led directly to the popularity of the LRU (least recently used) policy for page replacement. Similar strategies could be developed to cache recently referenced addresses at packet forwarding points in a network, if the probabilities of packets going to different destinations in a computer network are not the same.

Research by Jain and Routhier [20] involved analysis and modelling of the packet arrival patterns on traffic traces obtained from a token-ring local area network at the Massachusetts Institute of Technology. This research was aimed at providing a comprehensive model for computer network traffic. This section reviews the analysis of some of the traffic models considered in this work.

Their initial analysis considered just the timing characteristics of the packet arrival process. In fact, the source and destination addresses of the packets were simply ignored. Time intervals between successive packets form a time series, whose mean, standard deviation, coefficient of variation and autocorrelation function (ACF) can be computed. The autocorrelation function shows the relationship of an element of a time series, say t_i, with a previous element, say, t_{i-k}. The covariance between t_i and t_{i-k} normalized by the variance of t gives the ACF at lag k. If the arrival of packets on the network is a Poisson process, the interarrival times would have a coefficient of variation of 1 and the ACF would be 0 at all lags.

Results obtained from their measurements showed that the ACF was small (0.04), which indicates that successive time intervals are independent. However, the coefficient of variation (43.1) was very high compared to unity, leading to the conclusion that interarrival times are not exponentially distributed and thus that packet arrivals are not a Poisson process.

Another analysis involved analyzing the traffic by dividing the stream into node-pair processes. For each pair of nodes on the network, say (A, B), all traffic traveling between A and B form a time series that can be analyzed separately. Mean, standard deviation, coefficient of variation and ACF at lags 1, 2, and 3 were computed for the ten most active node pairs. The ACF was still small (in the 0-0.1 range), indicating negligible correlation. The overall coefficient of variation for node-pair processes (20.6) was smaller than that for the packet stream as a whole, but was still high compared to that for a Poisson process. From the distribution of interarrival times it was concluded that most packets arrive within a short interval of the previous arrival.

In order to better model the traffic, a "train" model was proposed. A *train* of packets is defined as a series of packets travelling between a given pair of nodes, such that intercar (or interpacket) times were smaller than a specified maximum referred to as the maximum allowed intercar gap (MAIG). If no packets are seen travelling between these two nodes for a period of time equal to the MAIG, the previous train is declared to have ended and the next packet between these nodes is declared as the locomotive of the next train. There might be several trains travelling simultaneously on the network.

Experiments were conducted using various values for MAIG. The coefficient of variation was close to unity, which implies that interpacket time in a train is exponentially distributed. However, the ACF is nonzero, indicating that intercar periods are dependent. During busy periods, it takes a long time to send/route a packet because more hosts are contending for the network. As a result intercar times for a given host pair are long. During periods of light load intercar times are short.

Difficulties making accurate predictions with this model led to the proposal of a more limited model called the "tandem-trailer" model. A sequence of successive packets going from one source to the same destination is called a tandem trailer.

This model aimed to provide lower variance and more correlation so as to increase the predictability of resource requirements, but the results obtained show a non-zero variance, and the predictability was basically the same as with the train model.

Jain and Routhier's work also showed that the probability of the next packet coming from any of the possible sources is not the same. There had been a notion that there exists a uniform probability of a packet coming from all the sources on the network. Under this assumption, given a network with m nodes,

The probability of the next packet coming from a given source $i = 1/m$

The probability of the next two packets coming from a given source $i = 1/m^2$

and

The probability of the next two packets coming from the same source $= 1/m$

Twenty one nodes were monitored. Theoretically the probability that two consecutive packets will come from the same source would equal 1/21 or approximately 5 percent. However, the results in [20] showed that the probability of a packet going from A to B being followed by another packet going from A to B is 29 percent, and that of an A-to-B packet being followed by a B-to-A packet is 31 percent. This showed the existence of considerable "source locality" (the tendency of successive packets to belong to the same train).

Two major results of the packets trains research are:

1. Packet arrival process on a network is not a poisson process.
2. The packets have been found to have high source locality.

These results showed that packet arrivals on a network are not independent. Recent research by Ling [24] has shown similar traffic characteristics in an Ethernet LAN environment. In order to model the workload of Ethernet, Ling extended the packet trains model. A packet tandem model was developed using traffic measurements from the Computational Science network at the University of Saskatchewan. The traffic traces used for the analysis presented in this thesis were captured from the same environment.

There are a number of applications that would benefit from the locality property of the traffic. These results obtained by Jain form the basis for future research in this direction, which includes the work presented in this thesis.

3.2 Characterizing Destination Address Locality

As the speed of networks increase, the time available for packet forwarding at network interconnection points decreases (interconnection devices which previously handled a few hundred frames per second, are now expected to handle thousands or hundreds of thousands of frames per second). One part of the forwarding operation consists of searching through a large address database to determine the direction the currently received packet must be forwarded. This problem is encountered in the design of bridges, routers, gateways and name servers. Jain [21] used a destination address reference trace measured on an extended local area network to see if destination references have significant locality. If some locality existed in the destination address reference pattern, these addresses could be cached in an attempt to reduce the lookup time. The applicability of three familiar locality models, IRM (independent reference model), WS (working set), and LRU (least recently used) to the address reference trace was studied.

The independent reference model assumes that references are independent, i.e., knowing that the last reference was to address k does not give any information about the next address to be referenced. This model assumes that there is no temporal or spatial locality. The cumulative frequency of reference as a function of fraction of distinct addresses, obtained by appropriately parameterizing the IRM model, shows that the destination reference pattern is nonuniform. In the reference trace studied, 50% of the frames were destined to 4% of the destinations and 90% of the frames were destined to 17% of the destinations. In other words, destination addresses exhibited strong concentration of reference in this particular trace.

The working set model assumes that the addresses referenced in the last W references are highly likely to be rereferenced. The interval W is called the *working set window size*, and the number of distinct references in the interval is called the *working set size*. If temporal locality is present, the working set size would be much smaller than the working set window size. The degree of locality seems to depend somewhat on the window size observed. For a window size of 500 the working set size was found to be 65, indicating high temporal locality. For window sizes below 50, temporal locality was low (working set size is almost equal to the working set window size).

The LRU stack model assumes that the probability of reference to an address is a decreasing function of the time since it was last referenced. This analysis showed that the top 100 stack positions account for 98% of the frames.

A study of MIN, LRU, FIFO and Rand cache replacement policies was also conducted. The metrics used were miss probability, interfault distance, and normalized search time. Since MIN requires lookahead in the reference string, it cannot be implemented in a real system, but the results obtained for MIN can be used as the theoretical optimum and as such as a benchmark for comparison. The miss ratio of MIN was found to be better than the rest by a factor of two, so there is room for improvement by designing another replacement algorithm. The interfault distance, defined as the number of references between successive cache misses, was also found to be better for LRU than for FIFO and Rand.

The last metric considered was that of normalized search time. Assuming that a full set of n entries would generally require a search time proportional to $1 + log_2(n)$ (assuming binary search), so:

Time to search without cache $= 1 + log_2(n)$

Time to search the cache $= 1 + log_2(c)$; c being the size of the cache.

Normalized search time (t) = Search time with cache / Search time without cache

Let p be the cache miss ratio

Time to search in case of a cache hit $= 1 + log_2(c)$

Time to search in case of a cache miss $= 1 + log_2(c) + 1 + log_2(n)$

So search time with cache $= (1-p)(1 + log_2(c)) + p(1 + log_2(c) + 1 + log_2(n))$

So the Normalized search time is,

$$t = \frac{(1-p)(1 + log_2(c)) + p(1 + log_2(c) + 1 + log_2(n))}{1 + log_2(n)}$$

An LRU cache achieved 30% lower search time than that without caching. This can be a major improvement in performance of the device.

This research by Jain was an attempt to determine if the destination addresses of references on a network exhibit the locality behaviour and whether this behaviour can be used to make destination address recognition in a bridge more efficient by using a cache to store some of these addresses. Four cache replacement policies were compared.

In the preliminary research conducted for the work presented in this thesis, it was observed that the source addresses of the last few references can help predict the destination addresses of some future frames. The research by Jain [21], presented above does not take this factor into account. Since one of the goals of this thesis, namely, determining the performance of a data link level interconnection device, is the same as that of the work discussed above (albeit with certain extensions), it would help in comparing results.

3.3 Gateway Routing-Table Cache Performance

A gateway is a device that connects two or more heterogeneous networks and is responsible for transporting packets between them, assuring that packets arriving from one network are transmitted toward their ultimate destination. The gateway examines the destination address of a received packet and determines the next-hop address by using the destination address as an index to a *routing table* [11, 12]. A routing table is a table of destination/next-hop address pairs maintained by the gateway. High-throughput gateways are necessary in an internetwork environment to provide high speed communication that is transparent to the user. A fully utilized gateway connecting two 100 Mbps networks transporting 500 byte packets would have only 40 microseconds to handle each packet. So the routing table lookup time must be reduced to achieve high throughput gateways.

In order to reduce routing-table lookup time per packet, a routing-table cache can be used, since previous work has shown the existence of considerable locality in computer network traffic. Measurements by Jain and Routhier [20] show that if a packet is sent from A to B, there is a 29% chance that the next packet is from A to B and a 31% chance that the next packet is from B to A. These characteristics can be attributed to two types of traffic prevalent on computer networks, namely, interactive (e.g., remote login) and bulk data transfer (file transfers).

The purpose of the research done by Feldmeier [11] was to determine whether internet address caching in gateways is effective and to calculate the expected cache performance. The methodology used for this research was trace-driven simulation. A trace was gathered for an operating gateway by recording every routing table reference during normal gateway operation. This trace was then used to drive a cache simulation model. Two replacement strategies, LRU and FIFO, were simulated. In addition, two situations were analyzed. The first involved no prefetching for the source address, i.e., only the packet destination is used to update the cache. The second allowed prefetching any time the packet source was not in the cache. For the model, the cache was fully associative. This avoids the problem of measurement bias because of specific network addresses in the measurements and makes the results general for any set of network addresses with similar characteristics. LRU and FIFO caches were modeled.

Measurements for this study were made at five gateways and analysis was performed on them. Cache performance depends on the relative popularity of various addresses used in packets that pass through the gateway. It was found that relatively few addresses (20-25) account for the majority of packets that pass through the gateway, a sure sign that a cache could be effective.

Packet trains also contribute significantly to the effectiveness of a routing-table cache. The first packet of the train initializes the cache and as long as the cache is not too small, all following packets are cache hits. Cache hits also occur when the first packet of a train is heading to a destination already in the cache. The destination addresses of packets in an active train are most likely to be near the first slot of an LRU cache. Addresses of inactive but popular train destinations are likely to be between slot i, where i is the number of simultaneously active trains, and the last slot of the cache.

Feldmeier also introduced the notion of source prefetching for a routing-table cache. Since a packet from host A to host B is often followed by a packet from host B to host A, a strategy for prefetching would produce better results. If the cache prefetches the next-hop for the packet source, then a packet traveling through the same gateway in the reverse direction would no longer cause a cache miss. Even if prefetching is inexpensive, there is the possibility of forcing an active entry out of the cache. This implies that in order for prefetching to be effective, the probability of a reverse-direction packet must be large or the cache must hold up to twice as many entries as the number of simultaneously active host-pairs.

The important question that this analysis tried to answer is what performance improvement is gained for the gateway by using a routing-table cache. A worst-case analysis was done for a fully associative cache with an LRU replacement policy.

The cache was implemented completely in software as a doubly-linked linear list. The list was searched linearly from most-recently-used to least-recently-used. Only destination addresses were cached.

Feldmeier concluded that the two cases (with and without prefetching) represent the upper and lower performance bounds for fully associative LRU caches where prefetching occurs. It was further observed that in the case of a small cache, prefetching may be detrimental to performance since it may cause a destination address which has a higher possibility of being referenced next, to be thrown out. A cost analysis was performed, and it was shown that lookup time can be reduced to 23% of its current time, and a better cache design could reduce this even further.

This research by Feldmeier attempted to estimate the performance of a network layer interconnection device that has a routing-table cache to exploit the traffic locality. This study also illustrated the existence of considerable locality in the traffic. The results obtained showed that the performance of such a device shows significant improvement when locality of traffic is taken into account. It would therefore be interesting to see as to how much improvement can be brought into the performance of a data link level device designed to suit the locality property of the traffic.

3.4 Network Locality at the Scale of Processes

Mogul [29] attempted to trace locality in network traffic at a finer grain, looking at references to individual processes. With few exceptions, processes (or threads) are the entities actually doing the communication. Host addresses are a consequence of sharing hardware resources between several processes. Often, the locality properties visible at the host address level actually arise because of per-process network locality. Similarly, host-pair locality is often a consequence of locality in the communication between pairs of processes.

Per-process network locality is important for several reasons. Security and qualityof-service policies of network infrastructure (such as routers) might be better expressed in terms of communicating processes rather than communicating hosts. Once a packet reaches its destination host, the speed of delivery to the receiving process is strongly determined by decisions such as scheduling choices that might benefit from consideration of locality. The measurements for the study in [29] were made by using a passive monitor on a broadcast LAN.

Mogul's experiments showed that per-process locality is an important characteristic of LAN packet traffic. About two thirds of the packets arriving at a host came from the same port as their predecessors. Nearly half were destined for the port that last sent a packet from that host. The results indicated a high degree of temporal locality at the process level.

Temporal locality could be a major factor in determining the scheduling policy with regards to what happens to a process waiting for input from the network. There are two options, namely:

1. "Dally" the process until the response arrives.
2. Switch immediately to another runnable process.

The factor that would determine the choice between these options is how the cost of context switching compares to the average time wasted dallying. If the kernel could estimate which packet reception attempts are likely to succeed quickly, either by static analysis or by keeping historical information, then it could dally only when it had a high chance of paying off. This would complicate the kernel, but might eliminate many context switches without wasting many CPU cycles.

Six scheduling algorithms based on an estimate of rapid network response were simulated. The results showed that the algorithm that allows the process to "dally" if the previous packet arrived within a certain time period performs best. It was observed that more than 50% of the previous packets have arrived within this time limit. However, there seems to be enough variation in communication patterns of individual processes and protocols to make performance differences between algorithms minimal. For example, while some replies arrived within a very short interval, most did not.

It is possible that a more successful scheduling algorithm could be devised, especially if it classified processes according to the protocols they were using. A better approach might be to provide a mechanism for user processes (which may know much more about their communications patterns than can be inferred by the kernel) to indicate whether to dally or not when waiting for a packet. For example, an RPC (remote procedure call) to release a file lock should return promptly, and would be a candidate for dallying; an RPC call to write data to the server disk would not be.

There has been considerable interest in evaluating the benefits in communication processing and implementations by using the traffic locality. Previous sections have shown studies conducted at various address granularities. Work discussed in this section brings forward another interesting application of network traffic locality, namely its application in scheduling decisions. Such optimizations would assume more importance as the bandwidth of networks increase further.

3.5 Summary

In this chapter some of the previous work done in the area of computer network traffic locality has been discussed. The results obtained by Jain and Routhier [20] show that the packet arrivals on a LAN are not independent but are correlated. It was observed that every host on the network does not have the same probability of sending the next packet. This behaviour is referred to as the "source locality" property of the traffic. In the study by Jain [21] it was observed that the packets received on a LAN show considerable destination address locality (every host on the network does not have the same probability of receiving the next packet). This behaviour can be exploited in the design of bridges, routers, gateways and name servers. Work by Feldmeier [11] shows that exploiting the locality characteristics of traffic can result in improvements in the performance of gateways. Mogul [29] focussed on locality at the per-process level and found that scheduling decisions made based on the locality property of the traffic can result in more efficient delivery of packets to the destination processes.

All these studies establish that the traffic on LANs exhibits locality properties and bring out some interesting applications of this property, but a detailed characterization of the various aspects of locality has not yet been attempted. Such a characterization is necessary if the locality property is to be exploited to its fullest potential. The next chapter deals with the characterization of aspects of LAN traffic locality.

Chapter 4

Characterization of LAN Traffic Locality

Previous research has shown that considerable locality of reference exists in computer network traffic. This implies that there is correlation between arrivals and hence there exists some predictability as far as future arrivals are concerned. This predictability could be used to make processing at the network interconnections more efficient. By storing relevant recently referenced addresses in a high speed buffer in the interconnection device, higher throughput and lower delays can be achieved. One of the goals of this research is to estimate the performance benefit that can be achieved by using a forwarding-table cache in a transparent data link layer interconnection device. The first step in this direction is to determine the characteristics of the network traffic locality and quantify them. This chapter presents a detailed analysis of some salient aspects of computer network traffic locality.

Section 4.1 describes the methodology used to determine the characteristics of network traffic. The characteristics studied are described under three categories in sections 4.2 to 4.4, namely, source address characteristics, destination address characteristics, and host-pair characteristics. Section 4.5 gives some of the conclusions reached about the characteristics of local area network traffic locality.

4.1 Methodology

For the purpose of determining the characteristics of network traffic in an actual operating environment, traces of the activity on the Computational Science Ethernet local area network at the University of Saskatchewan. These traces were captured using an HP 4972A LAN protocol analyzer [18]. This device has a one megabyte buffer memory for incoming data and a 20 megabyte Winchester disk. It timestamps each incoming frame, with timestamps accurate to 32 microseconds. This device was used in promiscuous mode, attached to one of the LAN segments. After capturing each trace, the data transferred to the file server for future use.

Ten different traces were obtained and formed the basis for the analysis that follows. Details of these traces are given in Table 4.1. A trace consists of a timestamp, a source address, and a destination address entry for each frame. So that the set of traces used are representative of the general activity on a local area network, the traces were captured on different days of the week and at times when different levels of activity were expected. The size of each trace was limited by the amount of disk space available on the LAN protocol analyser when the trace was collected. Thus the duration of each trace was dependent on the level of network activity during the trace period. Each trace contains a time-stamped reference string varying from 39,000 to 80,000 frames with a duration varying from 267 seconds to 620 seconds. From 503 to 665 distinct Ethernet addresses were seen during the periods traced. Since the Computational Science network is connected to the campus network, some traffic observed in the traces originated on other parts of the campus internetwork.

Ten traces represent a large body of experimental work. To conserve space, only

Table 4.1: Description of the Traces

No.	Date	Start Time	Duration	Size(frames)	Load(frames/sec)	Hosts Active
1	Feb 10	14:46:43	267 secs	50068	188	612
2	Feb 11	15:20:47	287 secs	50076	175	630
3	Feb 12	18:24:46	344 secs	80134	233	515
4	Feb 14	21:29:16	620 secs	80326	130	633
5	Feb 16	15:06:09	332 secs	80116	241	511
6	Feb 18	13:10:11	322 secs	60065	187	665
7	Feb 20	00:32:01	495 secs	60922	123	503
8	Mar 11	09:06:28	300 secs	39403	131	534
9	Mar 12	11:16:06	267 secs	51573	193	641
10	Mar 14	03:25:43	284 secs	50970	180	535
Averages:			351 secs	60365	178	578

three of the ten traces are selected for detailed analysis[1]. The chosen traces are Trace5, Trace6 and Trace8. Trace5 is chosen because it has the highest average load in terms of frames per second (241 frames/sec). It has a comparatively low number of active hosts (511). Trace6 has the highest number of active hosts (665) but the load (187 frames/sec) is about average. Trace8 has low load (131 frames/sec) and a number of active hosts (534) that is close to the average.

The locality analysis focusses on the addresses used in each packet in the trace. The addresses are further classified into sources and destinations for the purpose of analysis. Trace5 contains 469 distinct source addresses, and 117 distinct destination addresses. Out of the 511 hosts active during this trace, 394 hosts appear as senders only, and 42 as receivers only. There are 75 hosts that appear as both sender and receiver. Of the 665 host addresses in Trace6, there are 602 distinct source addresses, and 167 destination addresses. Among these 498 hosts appear as senders only, and 42 as receivers only. 104 hosts appear as both sender and receiver. Trace8 has 488 distinct source addresses, and 117 destination addresses. 417 hosts appear as senders

[1]The complete results for all ten traces are given in Appendix A.

only, 46 as receivers only, and 71 hosts appear as both sender and receiver. Every trace has almost four times the number of senders as receivers. There are a substantial number of hosts that act only as receivers or senders for the entire duration of the trace.

The detailed analysis of the traces focussed on the following characteristics:

1. *Persistence*: This measure shows how frequently successive packets travel to the same destination or come from the same source. Traffic with high persistence would frequently have successive frames heading for the same destination. This property could be used at an interconnection device to avoid repeated searches into the large forwarding table for the same address, thus resulting in faster frame forwarding.

2. *Address Re-use*: This measure quantifies the frequency of occurrence of cases when an address used in the currently received frame appears again as the destination address in at least one of the next few frames. This was examined with respect to both source addresses and destination addresses. The higher the frequency of address re-use the more beneficial it would be to store the address used in the past few references to predict the destination of the next few frames. Such predictability has the potential of making the processing of each incoming frame at a bridge, router, or gateway much more efficient.

3. *Concentration*: It is expected that any given trace interval will contain addresses of only a subset of the total set of active hosts on the network. The smaller the subset that is addressed in a particular duration of time, the higher is the concentration. Higher concentration means fewer slots are required in a forwarding-table cache to achieve a particular level of performance. Again, this aspect was quantified for both the source addresses and the destination addresses.

4. *Reference Density*: All hosts are unlikely to be equally responsible for the

traffic in a particular trace. Is there a subset of hosts contributing the majority of the references, and if so, what is the size of this subset? Reference density reflects the extent to which a few hosts on the network account for a large proportion of the traffic in the entire trace. This was analyzed with respect to both the source addresses and destination addresses. A trace can have very high concentration due to the activity being limited to a small number of hosts, but all these hosts might be responsible for almost an equal fraction of the traffic in the entire trace. The reference density measure focuses on this important point.

5. *Host-pair Locality*: All the locality characteristics proposed so far deal with source and destination addresses separately. Host-pair locality deals with the frequency of occurrence of cases when at least one of the next few frames is seen travelling between the same host pair as the currently received frame. A high level of host-pair locality would imply a high level of bidirectional or unidirectional communication between a pair of hosts. Coupled with the persistence values this characteristic would show which of the activities, unidirectional or bidirectional communication, is more pronounced. For example, if persistence values are low and the traffic has high host-pair locality, it would be clear that bidirectional (interactive) activity between host-pairs dominates. Such information would be useful when methods for exploiting the locality property of the traffic are explored.

4.2 Destination Address Characteristics

The presentation of results begins with a consideration of destination addresses. The consideration of source addresses follows in Section 4.3.

4.2.1 Persistence

A burst of consecutive frames heading for the same destination would be said to exhibit high destination address persistence. If a cache were used to store destination addresses from a reference stream with high destination address persistence, good performance could be expected from just a single element cache or a small cache (with LRU as the replacement policy). This is because only the first reference of a persistent reference stream can cause a cache miss and all the repeated references would be resolved by the entry on the top of the cache. Workload generated by bulk data transfers, for example, might be expected to yield reference streams with high destination address persistence. Traffic consisting of frames travelling from different sources to the same destination would also exhibit high destination address persistence.

The first effort to characterize network traffic locality is with a measure of persistence. The aim is to find the frequency of occurrence of substreams of size w, each frame of which has the same destination address as that of the frame preceding the substream, where w is called the *persistence window size.* The calculation is as follows:

$$P_w = \frac{N_w}{T_w} \times 100\%$$

where P_w is the persistence of a trace for a window size w, N_w is the total occurrence of substreams having each of the w frames with the same destination address and T_w is the total number of possible substreams of size w in the trace. For determining the persistence of a trace for a window size of w, sets of frames considered w at a time starting from the beginning of the trace and moving down the trace one frame at a time. So the persistence value for a window size of w gives the likelihood that the next w frames from any random point in the trace will have the same destination address.

The persistence data obtained for the three selected traces is plotted in Figure 4.1 for w varying from 1 to 10. The persistence values for each of the traces is low. The persistence values fall steeply when the window size being observed changes from 1 to 2. For Trace8, 17% of the time two successive frames (w=1) have the same destination address. This figure is 14% and 9% for the other two traces. Five successive frames (w=4) have the same destination between 1% and 3% of the time, and nine consecutive frames (w=8) have the same destination in less than 2% of the cases. For Trace5 some persistence is observed 18% of the time, and no persistence is seen 82% of the time, i.e., the next frame is destined to a different address than the current frame.

From the analysis of persistence it is concluded that there aren't many instances when successive frames are observed on the network destined for the same host. There are two possible reasons for this behaviour. First, low persistence might be due to bidirectional traffic being dominant. That is, packets from A to B are followed by packets from B to A and vice versa. The results for address re-use and host-pair locality characteristics will show if this is the case. Second, low persistence may be due to the statistical multiplexing of the network among many hosts. That is, since the traffic observed is the aggregate traffic between various hosts on the network, many individual traffic streams may be interleaved, masking their persistence.

The load on the network appears to have a bearing on the overall persistence observed, providing some basis for the latter hypothesis. Trace5, which has a much higher load than Trace6 and Trace8, shows lower destination address persistence. The persistent traffic is less frequently masked when the other hosts on the network are not very active (low load). This explains the difference in destination address persistence values obtained for the three traces.

High values of P_w for large values of w would have been useful in making desti-

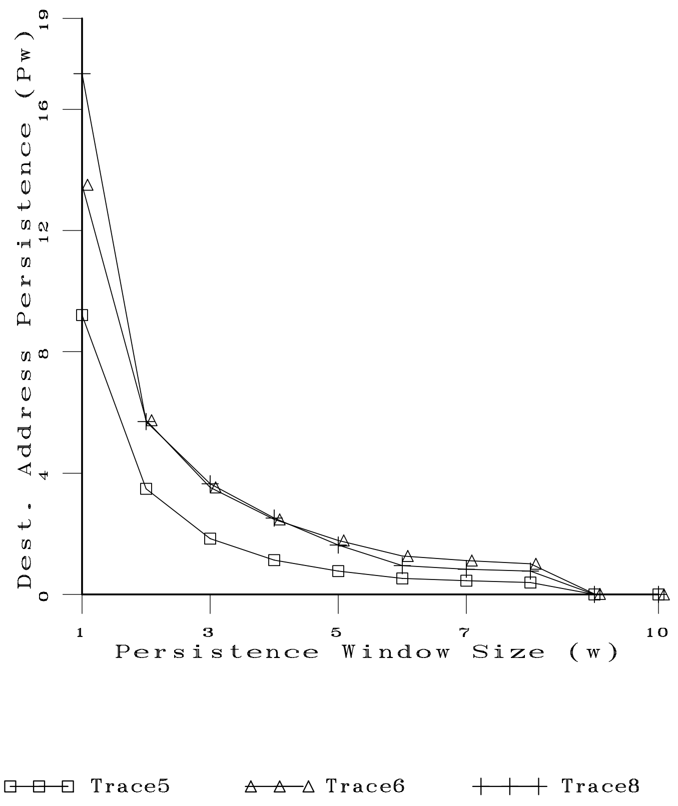

Figure 4.1: Destination Address Persistence Graph

nation address recognition more efficient and a single element cache or a small cache with the LRU replacement policy would have been extremely effective for caches holding such references. Instead, low values of destination address persistence have been observed, which leaves little opportunity to capitalize on this aspect of locality.

4.2.2 Destination Address Re-use

One important question is whether the destination address of the currently received frame is going to reappear as the destination address of at least one of the next n

frames, for small values of n. If so, what is the frequency of such cases? Destination address re-use is defined as the tendency for the destination address of a particular frame to be used as the destination address in future references. This definition, which is a measure of the frequency of occurrence of frames heading for the same destination but not necessarily immediately following one another, is a relaxed form of the definition of persistence which gives the frequency of frames heading for the same host and appearing on the network back to back. To study this aspect of locality, the complete reference trace is divided into substreams. Let w be the size of each substream. The frequency of reuse of the destination address of a frame as the destination address of at least one frame in the next w frames, D_w, is computed as follows:

$$D_w = \frac{N_w}{T_w} \times 100\%$$

where D_w is defined as the destination address re-use of the trace for substream size w, N_w is the total number of substreams exhibiting destination address re-use and T_w is the total number of substreams of size w in the trace.

The destination address re-use graph for the three selected traces is shown in Fig 4.2. The larger the window size w, the higher the probability that a destination is used again within that window. The rise in the re-use values is steep for small window sizes, indicating temporal locality. For larger window sizes, the address becomes "stale", and the probability of it being used after a few more frames have arrived diminishes, resulting in a smaller increase in the address re-use values.

The analysis shows significant destination address reuse. For $w = 1$, 9%-17% of the cases have two consecutive frames heading for the same destination, which is the same as destination address persistence for a persistence window size of 1. In 36%-53% and 58%-65% of the cases, the destination address of the current frame appears as the destination address of at least one of the 5 or 10 next frames, respectively.

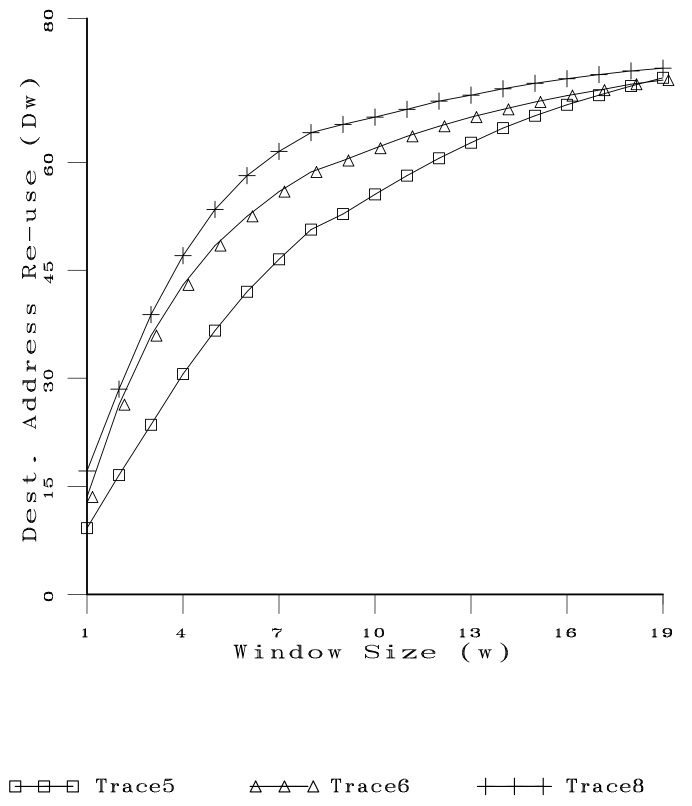

Figure 4.2: Destination Address Re-use Graph

The load on the network also has an effect on the destination address reuse results. The trace with highest load (Trace5) has the lowest destination address re-use values. The number of destination addresses in the trace also has an effect on the destination address re-use values. The larger the number of different possible destination addresses, the higher is the chance that a subtrace would contain references with different destination addresses. Since Trace6 has a larger number of different destination addresses than Trace5, Trace5, despite having higher load, catches up with the re-use values for Trace6 for window sizes above 15. So for a trace with higher load and more destination addresses, a lower address re-use value would be observed than for a trace with lower load and fewer destinations.

Overall, the destination address re-use measure suggests a caching approach for the design of an interconnection device. By using the destination address of a few recently received frames, a prediction can be made to some extent about the destination addresses of the next few arrivals. The extent to which this can be exploited was investigated by simulation and is reported in Chapter 5. Many network system design problems can benefit from this type of predictability. Another issue found to be of interest in the preliminary studies of network traffic locality is the contribution of the source address of the currently received frame in predicting destination addresses of future arrivals. This issue will be dealt with in the analysis of source address re-use in Section 4.3.2.

4.2.3 Concentration

For applications such as cache management, it is important to know how much of the total address space that is referenced during the entire duration of a trace is referenced during a smaller duration. Concentration is an aspect of locality which deals with this issue. The tendency of the destination addresses to be limited to a small subset of the address space is defined as the destination address concentration of the trace.

The working set model, used extensively to characterize program behaviour [9, 13], seems to be a good framework to express the notion of concentration in a reference stream. Let w be the working set window size and W_w be the corresponding average working set size. The working set size for a particular trace is the number of different hosts that receive at least one of the w frames. So when sets of w frames at a time are considered, W_w shows the number of different destination addresses which appear in such sets. The smaller the working set size, the higher is the degree of concentration in the trace.

Table 4.2: Destination Address Working Set Size for Different Window Sizes

w	10	30	60	100	150	200	300	400	500
Trace5	6	12	16	21	24	28	33	37	40
Trace6	5	11	16	21	25	29	35	39	42
Trace8	5	10	15	20	24	29	35	39	43

Table 4.2 shows the working set sizes for various working set window sizes for the selected set of traces. These results show that significant temporal locality exists in the references. For any trace interval of length w, the number of hosts active during that period (W_w) is significantly less than w and the increase with increasing w is clearly sub-linear.

The traffic analysed shows higher concentration than that observed by Jain [21]. For example, for a window size of 100, the working set size was 40 in the traffic used by Jain. The reason for the difference in results is the higher number of hosts attached to the network used in Jain's study as compared to the network used for this research.

it is desirable to have a measure of concentration that permits comparison across different window sizes. One of the drawbacks of the working set model is that the W_w values are not comparable for different values of w. For example, if $W_{100} = 20$ and $W_{200} = 28$ for the same trace, it does not necessarily mean that higher concentration is being observed for the window size of 100. In fact, one could argue that the window size of 200 shows higher concentration since there could be 200 different destination addresses that appear but only 28 (14%) are observed, while for a window size of 100, 20 (20%) different destination addresses appear instead of the 100 that are possible. So the concentration for different window sizes cannot be compared simply by comparing the corresponding values of the working set size. Instead, concentration should be expressed as a function of the working set size and the window size in order for the results for different window sizes to be comparable.

Defining concentration as the ratio of the working set window size and the corresponding working set size is an alternative but it is not suitable. Consider a window size of 100. If there is one destination address appearing in all the 100 frames sampled, i.e., $W_{100} = 1$, concentration would be maximum, 100; for $W_{100} = 2$ the concentration would be 50; and for $W_{100} = 3$ the concentration would be 33. For a particular window size, the values fall non-linearly as the working set size increases. Furthermore, the results for this measure are still not comparable across working set window sizes.

To overcome these deficiencies, a new measure for concentration is proposed. The new concentration measure is an extension to the working set model, and is normalized to values between 0 and 1. If the destination addresses of all the w frames are the same, i.e., $W_w = 1$, the concentration is said to be maximum, i.e., $C_w = 1$. If all the w frames have different destination addresses, i.e., $W_w = w$, the concentration is said to be minimum, i.e., $C_w = 0$. The concentration C_w is given by the following linear relationship with respect to the working set size (W_w):

$$C_w = \frac{w - W_w}{w - 1}, w > 1$$

The new concentration measure intuitively captures the notion of concentration. The smaller the number of different destination addresses in a given subtrace, the higher is its concentration.

The concentration measure for a variety of window sizes is plotted in Fig 4.3 for Trace5. Concentration increases with window size initially, but stabilizes for larger window sizes. The three traces selected show almost the same behaviour with regard to the destination address concentration. For window size of 30, references are concentrated to 10-12 hosts (from the W_w values), i.e., $C_w = 0.65$; for a window size of 200, they are concentrated to 28-29 hosts ($C_w = 0.86$); and for a window size of 500, references are localized to just 40-43 addresses ($C_w = 0.92$). The ragged nature of the curve is due to the conversion of the average working set sizes, which are

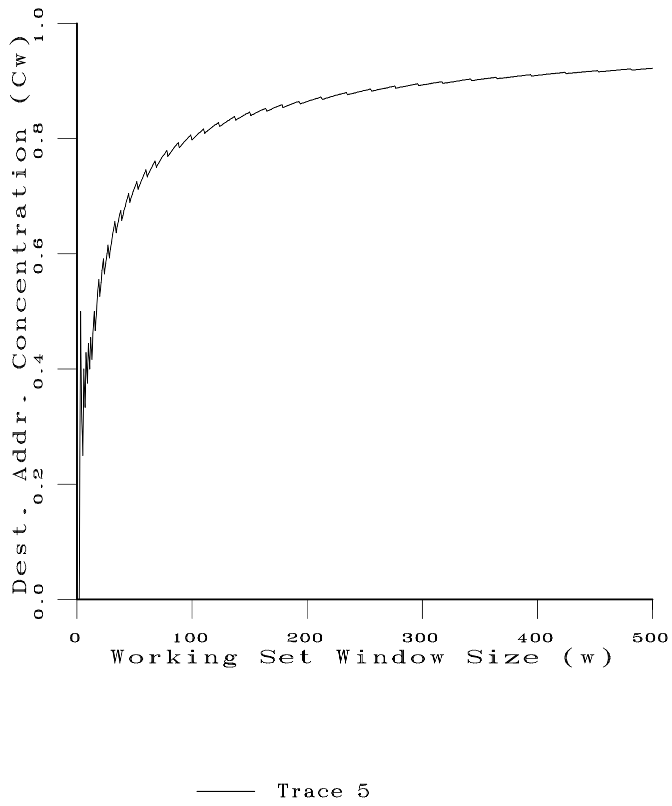

Figure 4.3: Destination Address Concentration for Trace5

real numbers, into integers for computing the concentration. These values show that references on a local area network are concentrated to small subsets of the address space over significant periods.

All traces show significant concentration and address re-use. The higher the destination address reuse and concentration, the lower would be the number of destination addresses that would have to be stored in a cache to predict the destinations of future arrivals.

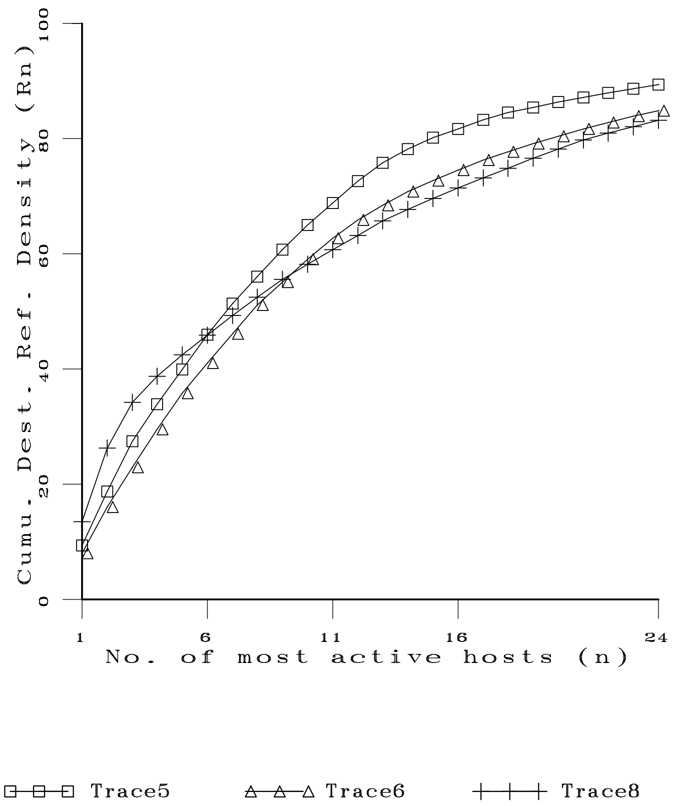

Figure 4.4: Cumulative Destination Address Reference Density Graph

4.2.4 Reference Density

Given h, the number of hosts active during a trace, and k, the number of addresses referenced in the trace, do all the hosts receive an equal proportion (k/h) of these references or are there certain hosts that receive more than their share of references? Destination address reference density, R_n, is defined as the percentage of references accounted for (as recipients) by the n most active hosts (in order of activity).

Figure 4.4 shows the graph of cumulative reference density of the 24 most active hosts for the selected set of traces. All the traces show similar behaviour. The curves show a non-linear increase which is rapid for the first 15 hosts and then levels off. This is because the reference densities of the most active hosts are considered first, so $R_{(n+2)} - R_{(n+1)}$ cannot be higher than $R_{(n+1)} - R_n$. As more hosts are considered, their contribution to the cumulative reference density diminishes causing the gradient of the curve to decrease. In the case of Trace5, the 10 most active hosts (2% of the active hosts) receive 65% of the references and the 24 most active hosts (5% of the active hosts) receive 90% of the references. In other words, the references are distributed nonuniformly across the address space. Results obtained by Jain [21] also showed high destination reference densities (5% of the possible destinations accounted for 55% of the traffic). The traffic analysed in this research shows higher reference density than those used by Jain. This is perhaps due to the presence of fewer hosts and file servers on the network considered here. The destination reference density values depend on the number of hosts on the network. The high values here reflect that there are certain hosts on the network that are receiving a large fraction of the traffic. These hosts are the file servers and a few hosts that are used by a large number of users.

4.3 Source Address Characteristics

This section presents results of the analysis of the characteristics discussed in Section 4.2 with respect to source addresses.

4.3.1 Persistence

A reference stream with repeated references from a single address in succession would be said to exhibit high source address persistence. The methodology used to determine the source address persistence is similar to that used for destination address persistence. Results obtained are shown for the three selected traces in Figure 4.5. The persistence values fall sharply from the window size of 1 to the window size of

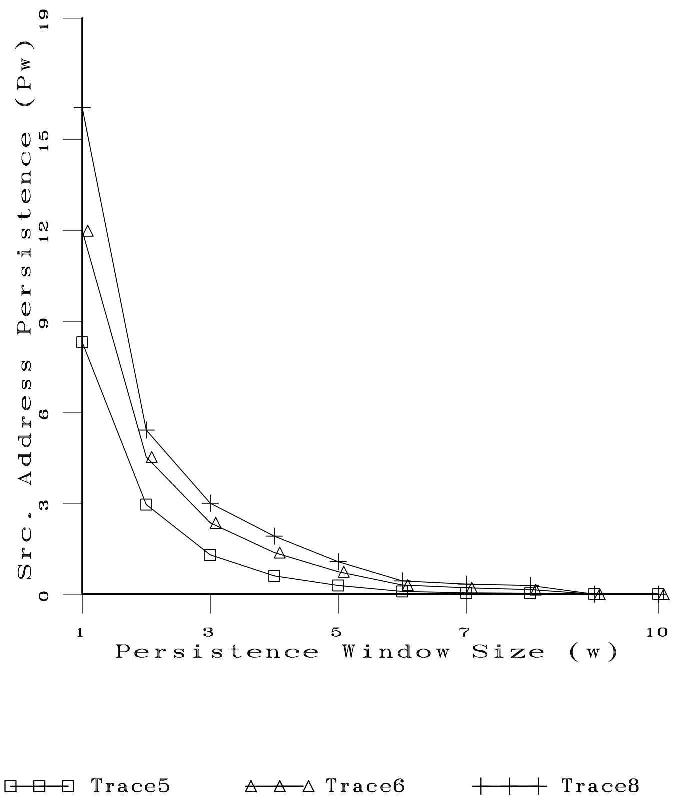

Figure 4.5: Source Address Persistence Graph

2 and almost 85% of the times no persistence is observed (not even two successive packets observed to be from the same source). Here again for the selected traces, the persistence values for the trace with higher load are found to be lower than for the traces with lower load.

Values obtained for source address persistence are lower than those for the destination address persistence (shown in Figure 4.1). This is because the number of different source addresses that appear in each trace is almost four times the number of different destination addresses in that trace. This results in a higher likelihood of frames from different sources getting interleaved and hence results in lower source address persistence.

4.3.2 Source Address Re-use

Source address re-use is defined as the tendency for the source address to be re-used as the destination address in future references. If there is a sufficiently high frequency of source address reuse then it could be possible to use the source address of the currently received frame to predict the destination addresses of future arrivals.

To study this aspect of locality, the complete reference trace is again divided into substreams of size w. The frequency of reuse of a source address as the destination address (S_w) of at least one frame in the next w frames is defined as follows:

$$S_w = \frac{N_w}{T_w} \times 100\%$$

where N_w is the total number of such substreams and T_w is the total number of substreams of size w in the trace.

Figure 4.6 shows a plot of S_w as a function of w for the selected traces. The source address re-use values rise quite sharply from window size of 1 to 6. Beyond this the curves have a very gradual rise. In 25% of the cases, the very next frame observed has a destination address the same as the source address of the previous frame. In 60% of the cases, at least one of the next six frames uses the source address of the currently observed frame as the destination address. Here again Trace5 which has the highest load, shows lower re-use, but since it has a lower number of different source addresses (469) than Trace6 (602) and Trace8 (488), its re-use values exceed those of the other traces for higher window sizes.

These figures show that there is a high degree of source address reuse. This implies that knowing the source address of the currently received frame can, to some extent, enable the prediction of the destination of the next few frames. This knowledge

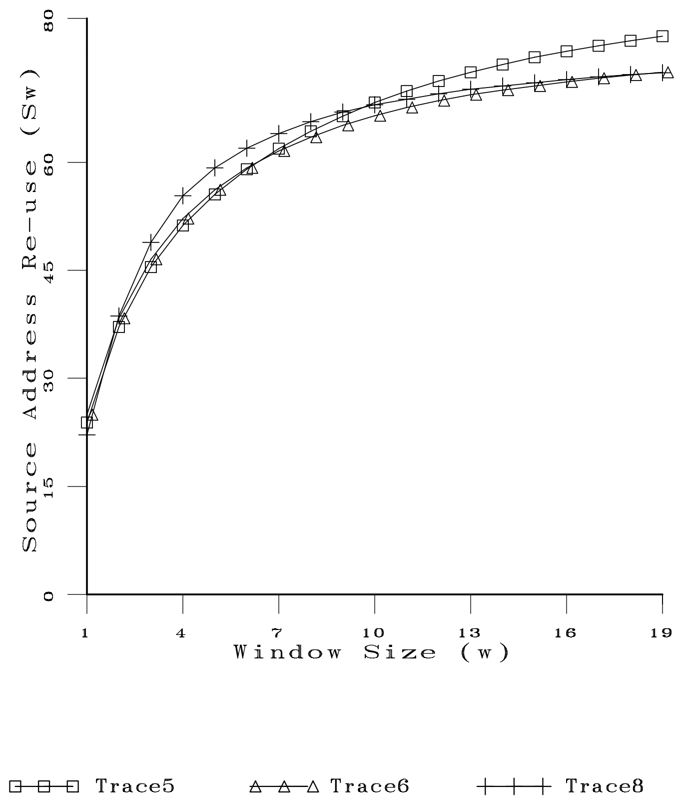

Figure 4.6: Source Address Re-use Graph

could be very useful from the point of view of communication processing. A bridge or a name server could fetch this "to be" destination address and hence reduce the delay for future processing. Overall system throughput could hence be improved with minimal changes to existing systems and protocols.

The source address re-use values obtained are higher than the corresponding destination address re-use values (shown in Figure 4.2) for window sizes less than 7. For example for Trace5, $S_1 = 24\%$ and $D_1 = 9\%$. This implies that when a frame is seen going from host A to host B there is a much better chance that the next frame would be destined to host A rather than host B. Fetching an entry into the forwarding-table cache for host A would therefore be more beneficial than fetching an entry for host B. Cache fetch policies, described in the next chapter, take this observation into account. In the case of higher window sizes the destination address re-use values are almost the same as the source address re-use values. This means that the increase in destination address re-use values beyond window size of 7 is greater than the increase in source address re-use values. Cache fetch policies can take these results into account too.

High values of source address re-use show that a lot of activity on the network might involve bidirectional traffic. These results along with the host-pair locality will give a good estimate of the level of bidirectional traffic.

4.3.3 Concentration

If a set of consecutive frames, say 100, are captured from a network, how many different hosts should these be expected to have originated from? The lower the number of different source addresses in these frames, the higher would be the source address concentration. For cache fetch policies that involve fetching the source address, it would be desirable that the source address concentration values be high so that a smaller sized cache would attain a particular level of performance. The method used for determining source address concentration is similar to that used for destination address concentration. For the selected set of traces, Table 4.3 shows the working set size (W_w) for various working set window sizes (w). Source address concentration (C_w) results obtained for Trace5 are shown in Figure 4.7. Trace6 and Trace8 show a similar pattern. Concentration rises quickly at first, then stabilizes for higher window sizes.

The selected traces show lower source address concentration than destination address concentration (shown in Figure 4.3). The working set sizes for source addresses are consistently larger than those for the destination addresses. For window sizes above 100, they are almost double those for the destination address case. These

Table 4.3: Source Address Working Set Sizes for Different Window Sizes

w	10	30	60	100	150	200	300	400	500
Trace5	6	13	18	25	32	38	50	61	71
Trace6	6	12	20	28	38	46	62	75	88
Trace8	6	14	22	31	46	58	77	93	106

results show that the destination addresses are much more concentrated than the source addresses. This is because the number of hosts that send at least one frame are almost four times the number of hosts that receive at least one frame. So the chance of different source addresses appearing in a particular subtrace is higher than the chance of different destination addresses appearing.

4.3.4 Reference Density

Source reference density of an address trace is a measure of the number of hosts that are responsible for the majority of references. This measure reflects the extent to which the references on a computer network are localized to a small set of hosts. The cumulative source address reference density graph is shown for the selected traces in Figure 4.8.

In Trace5, over 84 percent of the traffic is generated by 25 hosts (4%) of the 612 hosts active during the entire trace. This figure is 70 to 75 percent for Trace6 and Trace8 which have 665 and 534 hosts, respectively, active during the entire duration of the trace. This shows that all the active hosts do not contribute equally to the number of frames generated.

4.4 Host-pair Locality

Up to this point characteristics of source and destination address locality have been considered in isolation. In this section an aspect of locality that involves both the

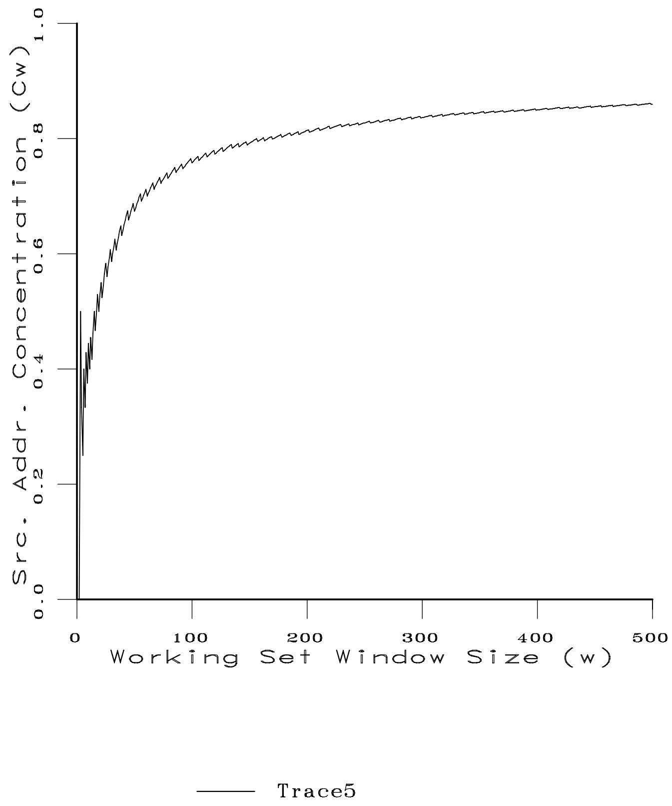

Figure 4.7: Source Address Concentration for Trace5

source and the destination address is studied. Host-pair locality is defined as the frequency of occurrence of at least one frame in the next w frames that is traveling between the same host pairs as the currently received one. This measure is similar to the address re-use measures discussed previously. Exchanges of frames between two hosts in both directions would be considered in this case. This would include unidirectional as well as bidirectional traffic between this pair of hosts.

Results for the host-pair locality measure are shown in Figure 4.9. For small w, values for host-pair locality are much higher than source and destination address locality. These results, coupled with the destination address locality and the source address

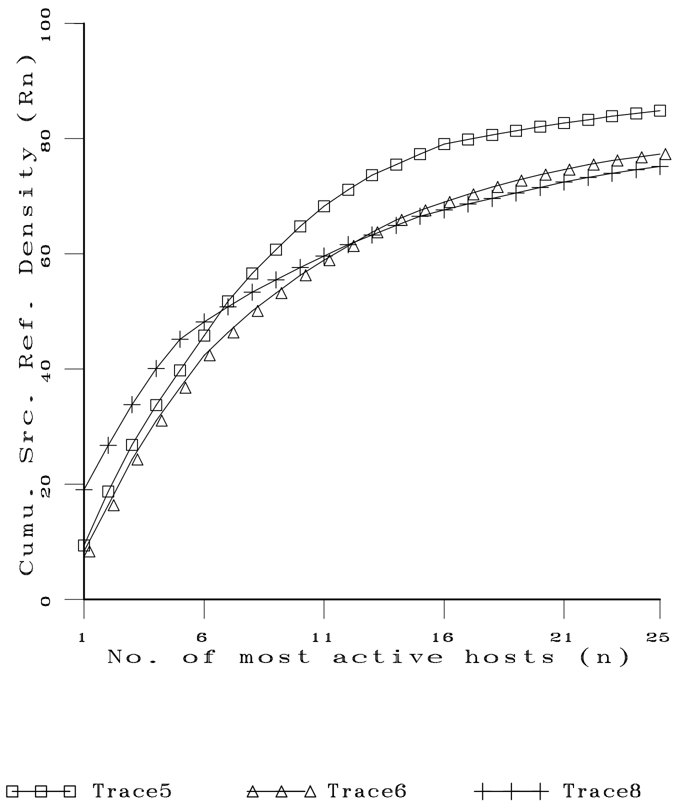

Figure 4.8: Cumulative Source Address Reference Density Graph

re-use figures, show that most of the traffic seen on the network is bidirectional. For example, $D_1 = 9\%$ for Trace5. This means that in 9% of the cases successive frames were destined to the same host. At the same time $S_1 = 24\%$ meaning that in 24% of the cases the next frame will be destined for the host that was the source of the current frame. So in 33% of the cases the next frame is destined for either the source or the destination of the current frame. The host-pair locality for window size of 1 is 31%. So in 2% of the cases the next frame was destined to the source or destination of the previous frame but has originated from a host other than these two. Even if all these cases occur when the frame is destined to the source address of the previous

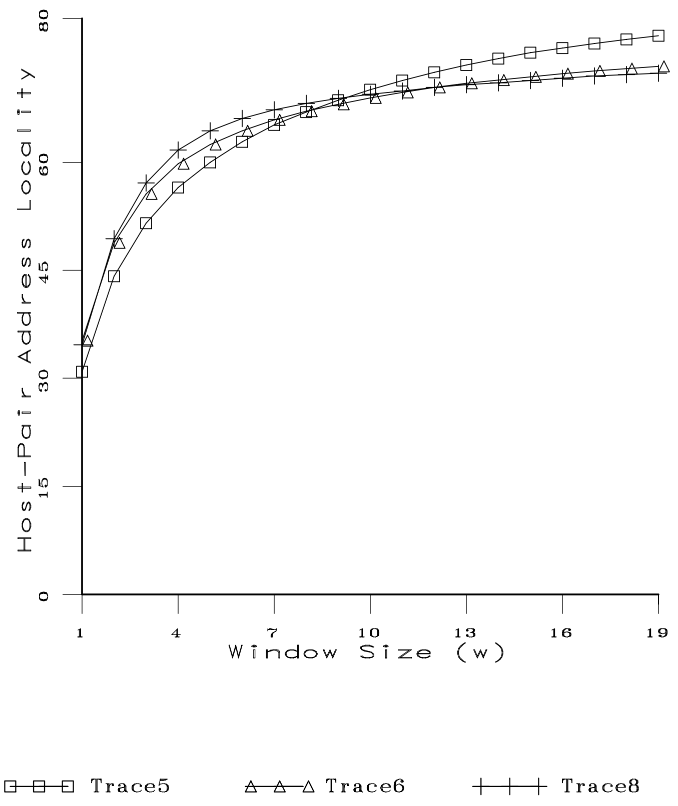

Figure 4.9: Host-Pair Locality Graph

frame, in $S_1 - 2\%$ (i.e., 22%) of the cases observed, the frame originates from the destination of the previous frame, and heads for the source of the previous frame. In 9% of the cases the next frame travels from the source to the destination of the previous frame. So the bidirectional traffic is twice the amount of unidirectional traffic. This explains the low source and destination address persistence values observed in the traffic and highlights the extent to which the source address of a frame can predict the destination address of the next one.

4.5 Summary

An analysis of traffic on a local area network was performed and various aspects of network locality were studied. Four characteristics of network traffic locality were presented: persistence, address re-use, concentration and reference density. These characteristics were studied separately for source and destination addresses. Host-pair locality, which takes into account the traffic between a set of hosts, was also analysed.

Low persistence was observed in the references but there did seem to exist substantial frequency of address reuse. The frequency of reuse diminishes with time elapsed since first reference. References were found to be highly concentrated and for each of the traffic traces considered, a small number of active hosts on the network account for a majority of the references. The frequency of instances when at least one out of two or three following frames was observed traveling between the same host-pair was very high. When traffic between a particular host-pair is considered, bidirectional traffic is almost twice the level of unidirectional traffic.

As far as predicting the addresses to which future references would be destined is concerned, examination of source address re-use shows that the source addresses of the current and the last few references can be reasonably good predictors. The host-pair locality and the address re-use results together show that request-response traffic is more pronounced than persistent unidirectional traffic, since the typical activity in the environment under consideration involves exchange of frames between clients and servers. Most of the persistent unidirectional activity between a particular host pair is obscured when the traffic on the network is considered as a whole since many chains of activity overlap giving a traffic mix much different than what might be expected in the case of an isolated pair of hosts interacting.

Having quantified some aspects of LAN traffic locality, the second objective of the thesis, namely determining the performance of a data link layer interconnection device having a forwarding-table cache, is dealt with in the next chapter.

Chapter 5

Performance Analysis of a Forwarding-Table Cache

This chapter deals with evaluating the benefit that can be achieved by capitalizing on the locality properties of network traffic in the design of data link layer interconnection devices. A cache could be used with the forwarding table in these devices to store addresses that have a high probability of being used in the future. Since the measurements reported in Chapter 4 show a high degree of source and destination address re-use, concentration, and reference density, caches much smaller in size than the forwarding-table itself should be able to resolve a sufficient fraction of the references to make this worthwhile, resulting in a decrease in the mean lookup time of these devices. This chapter reports on experiments carried out to study the effectiveness of such an approach.

Section 5.1 describes the methodology used to evaluate the performance of a forwarding-table cache. Issues related to the design of a forwarding-table cache are discussed in Section 5.2. Section 5.3 describes the simulation model. Section 5.4 discusses the results obtained by simulating a variety of cache replacement policies and cache fetch policies. An analysis of performance benefits, comparing the lookup time of the forwarding-table when a cache is not used to that obtained using a cache along with the forwarding-table, is also presented in this section. Section 5.5 summarizes the results obtained in this study.

5.1 Methodology

There are three widely used cache performance evaluation methodologies [1]. They are :

1. Hardware Measurements
2. Analytical Models
3. Trace-Driven Simulation

Hardware measurement would involve instrumenting an existing system and observing the performance of the cache. This approach is problematic for the analysis at hand because the cache parameters cannot be easily varied. Analytical models and trace-driven simulation provide greater experimental flexibility.

Analytical models estimate performance quickly, albeit at the cost of accuracy. In the case of caches, mathematical models have been used to gain insight into the behaviour of a cache. Trace-driven simulation is perhaps the most popular method for cache performance evaluation [1]. Trace-driven simulation evaluates a model of a proposed system using streams of addresses generated during the operation of existing systems as input. Trace-driven simulation offers the advantages of flexibility, accuracy, and ease of use. The experiments are fully repeatable and the same data can be used to compare different cache strategies. For the last several years trace-driven simulation has been the mainstay of cache performance estimation [5, 11, 12, 17, 31, 35, 36].

The usefulness of trace-driven simulation depends on the representativeness of the traces used to drive the simulations [31]. It is important to obtain traces that provide accurate cache performance predictions.

Given the general advantages of using trace-driven simulation and the availability of a tracing device, the HP 4972A LAN Protocol Analyzer, trace-driven simulation was chosen as the methodology for studying the performance of a forwarding-table cache.

5.2 Forwarding-table Cache Design Issues

General cache design issues were discussed in Chapter 2. These issues are explored with respect to the forwarding table cache design as follows:

1. Cache Fetch Policy: Three policies are studied:
   - DF (Destination Fetch): The destination address from the current frame is fetched into the cache.
   - SF (Source Fetch): The source address from the current frame is fetched into the cache.
   - SDF (Source and Destination Fetch): Both the source and destination address from the current frame are fetched into the cache.
2. Placement Policy: A fully associative cache is considered because the performance of any set-associative cache depends on the value of those items to be stored in the cache and the cache association mechanism. In fully associative caches information can be placed anywhere in the cache and placement is not an issue.
3. Replacement Policy: Five policies are considered: least-recently-used (LRU), least-frequently-used (LFU), first-in-first-out (FIFO), random replacement (Rand) and MIN. MIN, an optimal replacement policy, is considered for comparison purposes only. It is not a practically implementable policy.
4. Cache Size: A wide range of cache sizes are considered.

5.3 Simulation Model

One of the aims of this research is to study the effect of a forwarding-table cache on the forwarding delay of an interconnection device, specifically, to explore various options available for the design of such a cache and to analyze the benefits that can be achieved by using a forwarding-table cache. The forwarding table is a component of the interconnection device that contains a list of host addresses along with the port number from which a frame from each of these hosts was last seen. Destination addresses of incoming frames are looked up in the forwarding table to determine the port that the frame should be forwarded on. In the case where a forwarding-table cache is used, an attempt will be made first to resolve the references in the cache; if this fails, a forwarding table lookup would be performed. The objective of this simulation is to determine if using such a cache is beneficial for the system and, if it is, to determine the magnitude of the benefit that can be achieved.

The main objects involved in this system are the forwarding table and the forwarding table cache. For the sake of simplicity, the device is assumed not to have a processor cache. The event most strongly affecting the state of the model is the arrival of a frame at the device. Since the simulation is driven by previously captured address traces, the input to the model is a list of time-stamped references. Since references arrive at discrete points of time, the system state would change at discrete moments, so a discrete-event simulation [27] was used. The performance parameter of interest is the mean lookup time of a forwarding table which has a cache as compared to one without a cache. Two performance measures form the basis of this comparison: the frequency of references that cannot be resolved in the cache (cache miss ratio) and the mean depth in the cache at which addresses are resolved (mean address resolution depth).

The model was implemented using the C programming language. A record of activities is maintained, e.g., the number of misses and the depth in the LRU stack reached before a particular reference is resolved. The program provides the flexibility of experimenting with a variety of cache sizes and policies.

For validation purposes, sample data files were created to include various possible conditions. The results obtained using these data files were then compared with results determined manually. The results obtained from the simulation were found to match expectations.

5.4 Simulation Experiments and Analysis

The simulation program was executed for each of the traffic traces under different cache sizes and fetch policies. The simulation was first run for five replacement policies (LRU, LFU, Rand, FIFO and MIN), using the DF (destination fetch) policy for the cache and the three selected traces as input. The results are shown in Figure 5.1 for Trace5. Since the other traces show very similar results they are not presented. In Figure 5.1, the miss ratios drop sharply at first with increasing cache sizes as active entries are added to the cache. Beyond cache sizes of about 15 the additional entries retained have a much lower chance of being re-used and so the drop in miss ratios becomes more gradual. LRU consistently outperforms LFU, FIFO and Rand, and is within 20% of the optimal strategy (MIN). Beyond cache size of 50, because of the large number of entries in the cache, the difference in performance of all policies becomes marginal.

A cache operating with LFU as the replacement policy does not take into account the fact that a newly fetched entry has a high probability of being used in the next few references. Instead of being retained, this entry is most likely the one to be discarded from the cache when a replacement has to be made. Although LFU retains the entries that have had a high frequency of reference, the loss of some recently used entries results in its performance being poorer than the other replacement policies. A FIFO cache, on the other hand, retains these recently fetched entries until at least

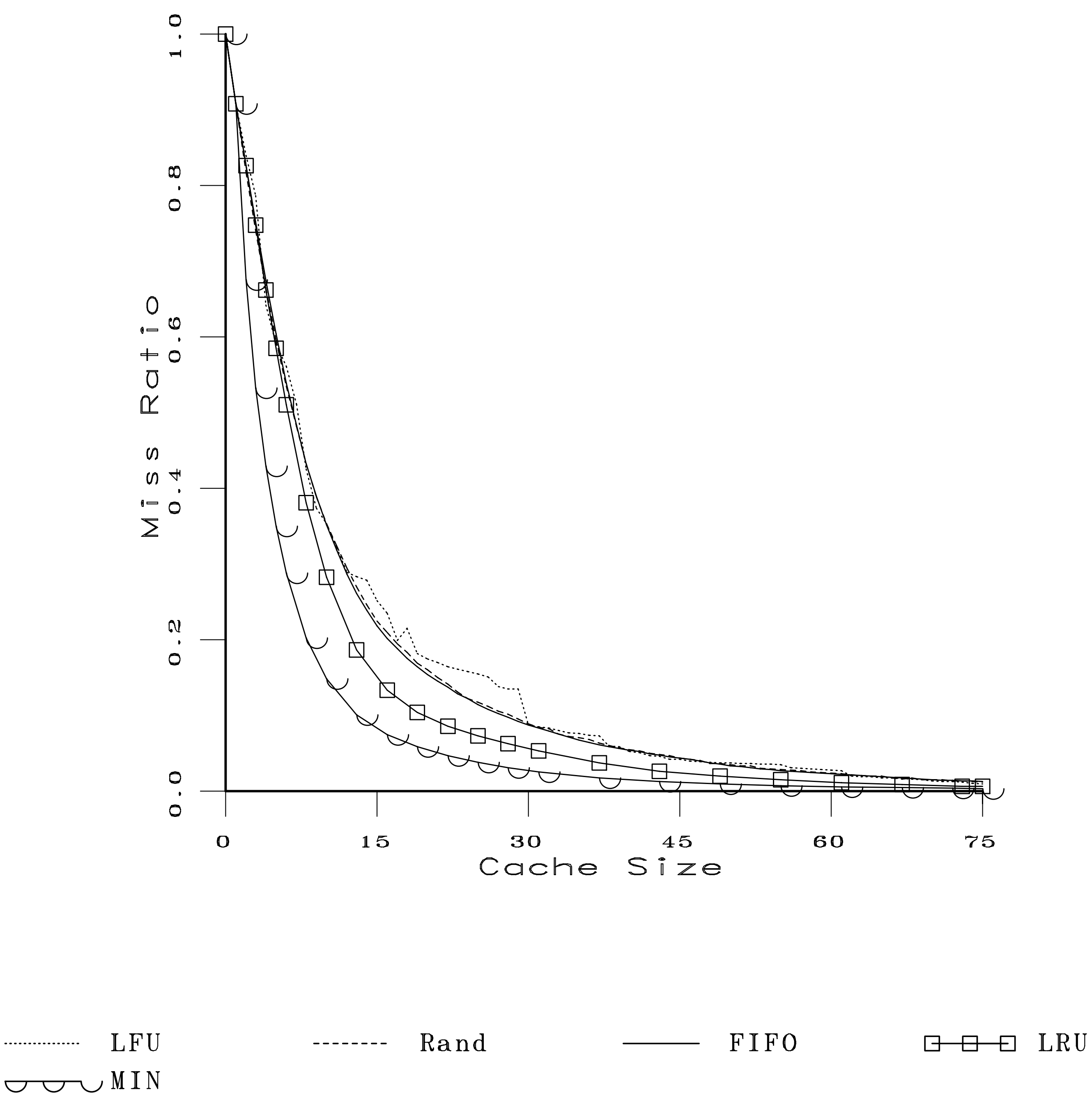

Figure 5.1: Cache Miss Ratios for LFU, FIFO, RAND, LRU and MIN (Trace5)

$n - 1$ more addresses are received, where n is the size of the cache, thus working to some extent like LRU except that the referencing of an entry already in the cache does not affect the duration of its stay in the cache, since replacements are not done on the basis of usage. Rand, which indiscriminately picks an entry to be discarded when a replacement has to be done, performs almost as well as FIFO.

As expected, MIN outperforms all other strategies for all cache sizes, but since it looks ahead in the reference string, it cannot be implemented in a real system. Results for MIN nevertheless show that there is room for improvement in implementable policies. Although LRU is the best among the implementable policies considered, it is sub-optimal. The suitability of LRU when used with the various fetch policies can be confirmed by an LRU stack reference frequency analysis. The results of this analysis are discussed in the next section.

5.4.1 LRU Reference Frequency

The replacement policy of the cache is an important design component that has a major impact on the performance of a cache. An analysis of the LRU stack reference frequency was performed to determine if the LRU replacement policy would be suitable for the selected traces and the three fetch policies investigated. The traces are fed into an LRU stack model and the frequency of reference to each of the positions on the stack is determined. If the frequency of reference decreases from the top of the stack (where new entries are added) to the bottom of the stack, it would mean that whichever entry is present at the bottom of the stack (which happens to be the least-recently-used entry in the stack) has the lowest probability of being used. So when a replacement has to be performed, removing this least-recently-used entry would be the judicious thing to do. If the bottom of the stack has a reference frequency higher than some of the other positions on the stack, removing the least-recently-used item, i.e., the contents of the bottom, would be inappropriate since this item has a higher chance of being used in the future than some of the other entries in the stack. So LRU is the appropriate choice (for a cache of size equal to or less than the size of the stack) only if the stack reference frequencies decrease with the depth of the stack. In order to perform an initial assessment of the effect of the various fetch policies discussed above, all three of these fetch policies were applied to this model. The proportion of references resolved at each position on the list was determined, for each trace. The results obtained are as follows:

Reference Frequency for LRU with the DF Policy

DF is a fetch policy that involves fetching the destination address of each incoming frame into the cache. Using the DF policy, the results for different stack sizes (represented by n) for Trace5 are shown in Figure 5.2. The other traces show similar behaviour. For most of the traces, the reference frequency was found to be a monotonically decreasing function of position on the stack, with the top 10 to 12 positions accounting for more than 70% of the references. The reference frequency results for DF depend on the destination address re-use results discussed in the previous chapter, e.g., the reference frequency for stack position 1 (top of the stack) is equivalent to the destination address re-use value for $w = 1$. LRU is suitable for the references with DF as the fetch policy.

Reference Frequency for LRU with the SF Policy

SF is a fetch policy in which only the source address of the current reference is fetched if it is not found in the cache. Interest in this policy stems from the figures for source address re-use presented in Chapter 4. A high degree of source address re-use was observed in the traffic traces, and the reference frequency using this fetch policy would give an idea as to how many references can be resolved by caching the source address of previous references. Results of reference frequency for Trace5 are shown in Figure

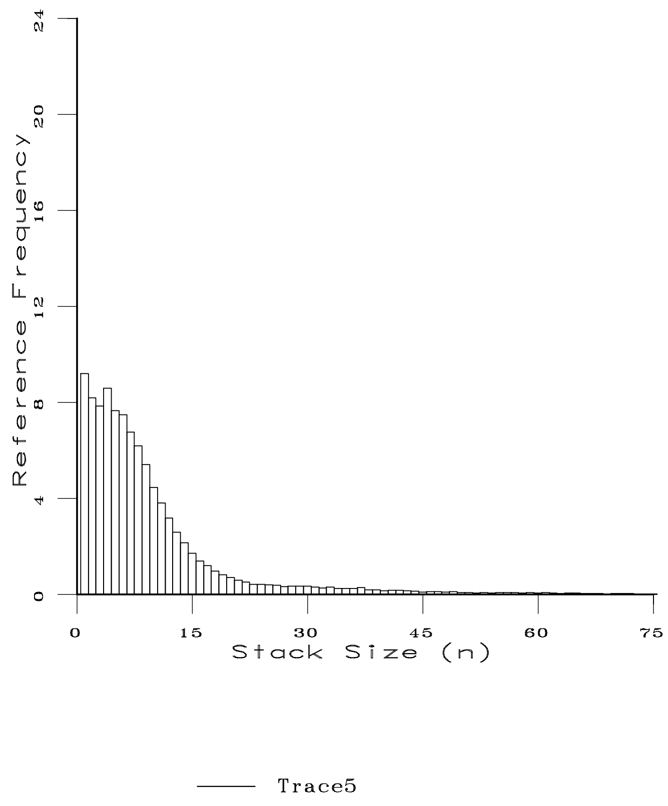

Figure 5.2: Reference Frequency for DF using Trace5

5.3. The results for the other two traces are similar to these.

SF again results in the reference frequency decreasing almost monotonically with distance from the top of the stack. An important result obtained for SF is that the references are largely concentrated in the top 13-15 positions whereas they were widely spread out with DF. This is because the source address re-use values show a negligible increase for window sizes greater than 15. This clustering of high reference frequencies near the top of the stack is desirable since it implies that a smaller cache would be able to resolve a large number of references and hence result in lower mean lookup time, provided that the cumulative reference frequency (equivalent to the hit

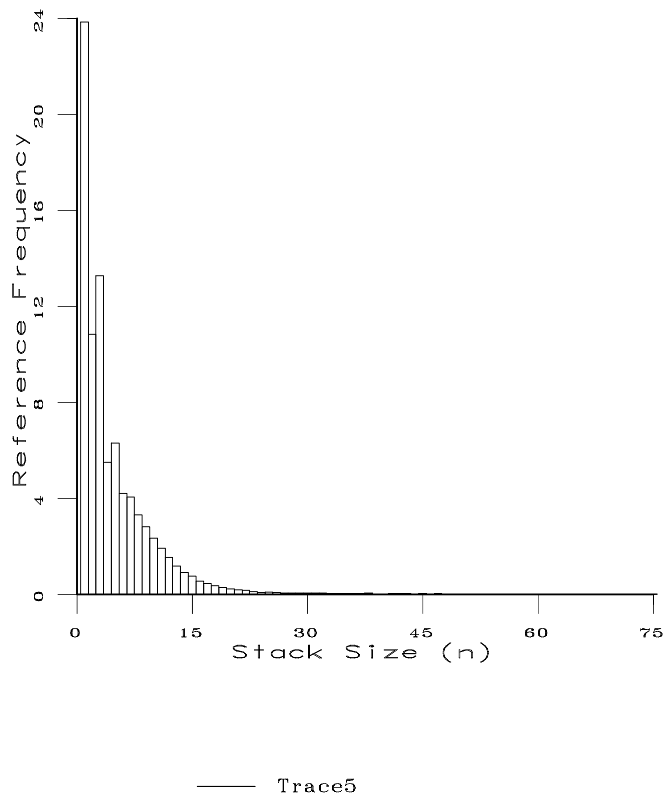

Figure 5.3: Reference Frequency for SF using Trace5

ratio) is also high. So an SF cache of size above 15 would perform marginally better than the cache of size 15. LRU seems to be suitable when SF is used as the fetch policy.

Reference Frequency for LRU with the SDF Policy

SDF is a fetch policy in which the destination address of the current reference is fetched as well as the source address. Since both the source address and the destination address of the current frame were found to have high frequency of reuse as the destination address of a future reference, the SDF policy would capitalize on both

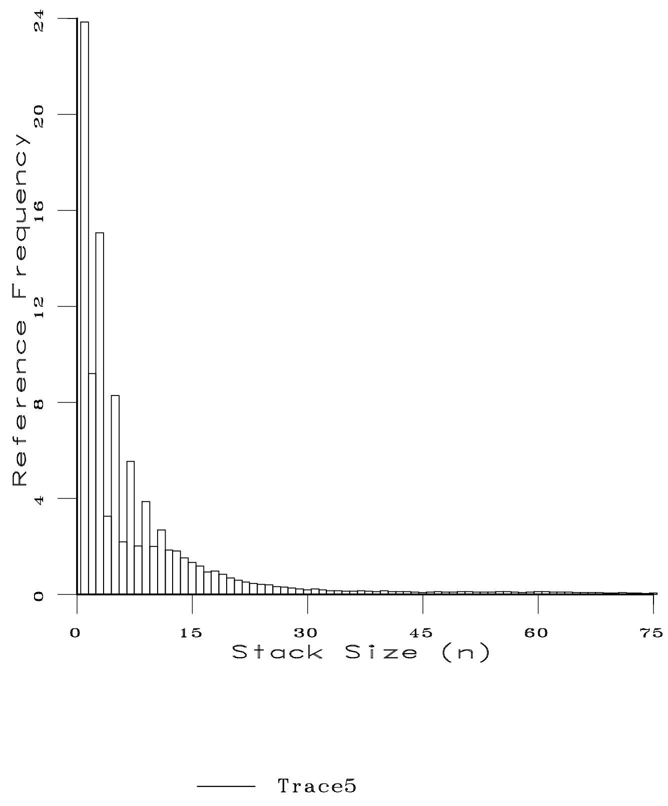

Figure 5.4: Reference Frequency for SDF using Trace5

these properties. Results of the analysis of reference frequency for Trace5 are shown in Figure 5.4. Trace6 and Trace8 show similar results.

SDF results in frequencies of reference that are not as widely dispersed as they are for DF, and not as highly concentrated as they are for SF. The cumulative frequency of the first 10 to 15 positions for SDF is around 70 to 80%, which is similar to SF (i.e., much higher than DF). Using SDF seems to create an undesirable effect, however. The reference frequency is no longer a monotonically decreasing function of position in the stack. There are a number of instances in which low and high frequency cases are interleaved, especially for the first 10 stack positions. The effect is explained as follows. Every frame arrival brings two new entries to the top of the stack. Position 1 holds the entry corresponding to the source address of the most recently received frame. Position 2 holds the entry corresponding to the destination address of this frame. Entries that have been brought in due to a source fetch remain at odd numbered positions until they are reused as the destination address of a future arrival or are replaced. Similarly entries brought in as destination fetch remain at even numbered positions. Since the source address re-use values are much higher than the corresponding destination address re-use values, the first few odd numbered positions of the stack have a higher reference frequency than some of the even numbered entries. This anomaly makes LRU less suitable as a cache replacement strategy if SDF were used as the fetch policy for a cache of size less than 13.

With each of the three fetch policies, high reference frequencies are observed for the first 13-15 stack positions. Beyond these positions, there are almost negligible reference frequencies when the source addresses are fetched, while they are higher when destination addresses are fetched. A sharp improvement in performance, would therefore be expected up to cache sizes of 13-15. Beyond this size the improvements would be much more gradual.

This analysis of LRU stack reference frequencies shows that LRU is well suited for the DF and SF fetch policies, but when SDF is used, LRU may not be appropriate for small cache sizes. Miss ratios for SDF are nevertheless discussed along with those for the other fetch policies in the next section.

5.4.2 Cache Miss Ratio

Having determined that LRU provides the best miss ratios and is appropriate for the references in consideration, the three fetch policies (SF, DF and SDF) were then simulated with LRU as the replacement policy. The results are shown for Trace5 in Figure 5.5. Trace6 and Trace8 show similar results. For all three fetch policies, the

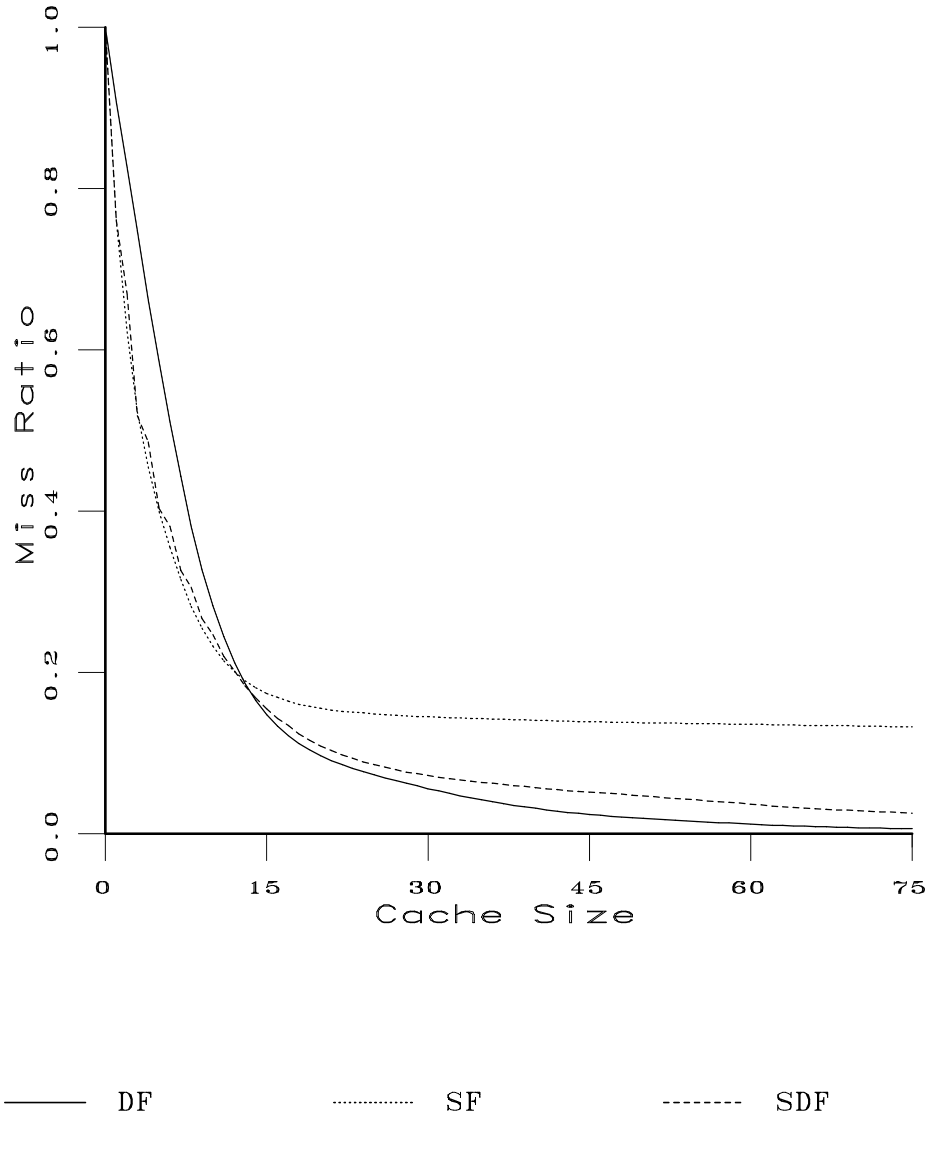

Figure 5.5: Cache Miss Ratios for Trace5

miss ratios fall sharply up to a cache size of about 13, and then decrease slowly beyond this point. For SF, the miss ratios become almost constant beyond cache size of 13, while the other two policies show a gradual decrease. Their relative performance for a particular cache size depends on the cumulative reference density for all the positions on a stack of this size, which in turn depends on the address re-use values of these traces. For cache sizes below 13, for most of the traces, the three fetch policies perform similarly, although SF performs marginally better than SDF which in turn performs better than DF. The better performance of SF and SDF over DF for these cache sizes can be attributed to the fact that the figures for source address re-use for window sizes less that 13, as discussed in Section 4.3.3, are higher than the corresponding figures for destination address re-use described in Section 4.2.3. Hence fetching the source address of each frame received would produce fewer misses than fetching the destination addresses into a cache of size less than 13.

For larger caches (more than 13 entries), Figure 5.5 shows that it is necessary to cache destination addresses to achieve improved performance. In other words, very little additional benefit results from caching further source addresses. Caching only source addresses (the SF policy) results in virtually no improvement. While caching both source and destination addresses (the SDF policy) improves performance over SF, it does not achieve the performance improvement attained by caching only the destination addresses (DF policy). In other words, caching source addresses as well as destination addresses hurts performance for larger caches.

Another reason for the lack of improvement in the performance of SF beyond cache size 13 is that there are some hosts that appear as destinations but never appear as the source address of a frame for the entire duration of the trace. For example, out of the 117 different hosts that receive frames during the time Trace5 was captured, 47 hosts never send a frame. Since SF does not fetch destination addresses into the cache, no entry for these hosts is ever fetched into the SF cache. Therefore, each frame going to any of these 47 hosts results in a cache miss. DF and SDF do not face this problem. SDF, which fetches the source address as well as the destination address, performs worse than DF beyond cache size of 13. This is due to the futility of retaining entries that were brought in due to source prefetch and not used since the last 13 or more arrivals.

5.4.3 Mean Address Resolution Depth

Another measure by which to judge the relative performance of the three fetch policies is the mean depth in the LRU stack at which the references are resolved. Results for this measure are shown in Figure 5.6 for Trace5. Assuming that a cache is structured as a list with the entries ordered by the time of recent use, mean address resolution depth of the cache for a particular trace is the average number of nodes searched in the list to resolve the references (when searched sequentially). In the case of a cache miss, the entire cache is searched. Trace6 and Trace8 show a similar pattern of behaviour. As expected, the mean address resolution depth increases with cache size. For DF and SDF the curves level off rather sharply after cache size of 13-15. In fact, the two curves meet at a cache size of 75. For SF, the mean address resolution depth continues to increase steadily with increase in cache size.

The mean address resolution depth is lower for SF and SDF for cache sizes under 13. This can be explained using the stack reference frequency results discussed in Section 5.4.1. Because of the lower values of reference frequency for most of the first 13 positions of the stack for DF, and the lower cumulative reference frequency (hit rate), the addresses tend to be resolved deeper in the DF stack.

For larger caches (more than 13 entries) the graph is explained in the following way. The stack reference frequency for SF for stack size beyond 13 is lower than that of DF and SDF. Increasing the size of the SF cache beyond 13 causes a slight reduction in the miss rate, but also causes a constant increase in the depth that is

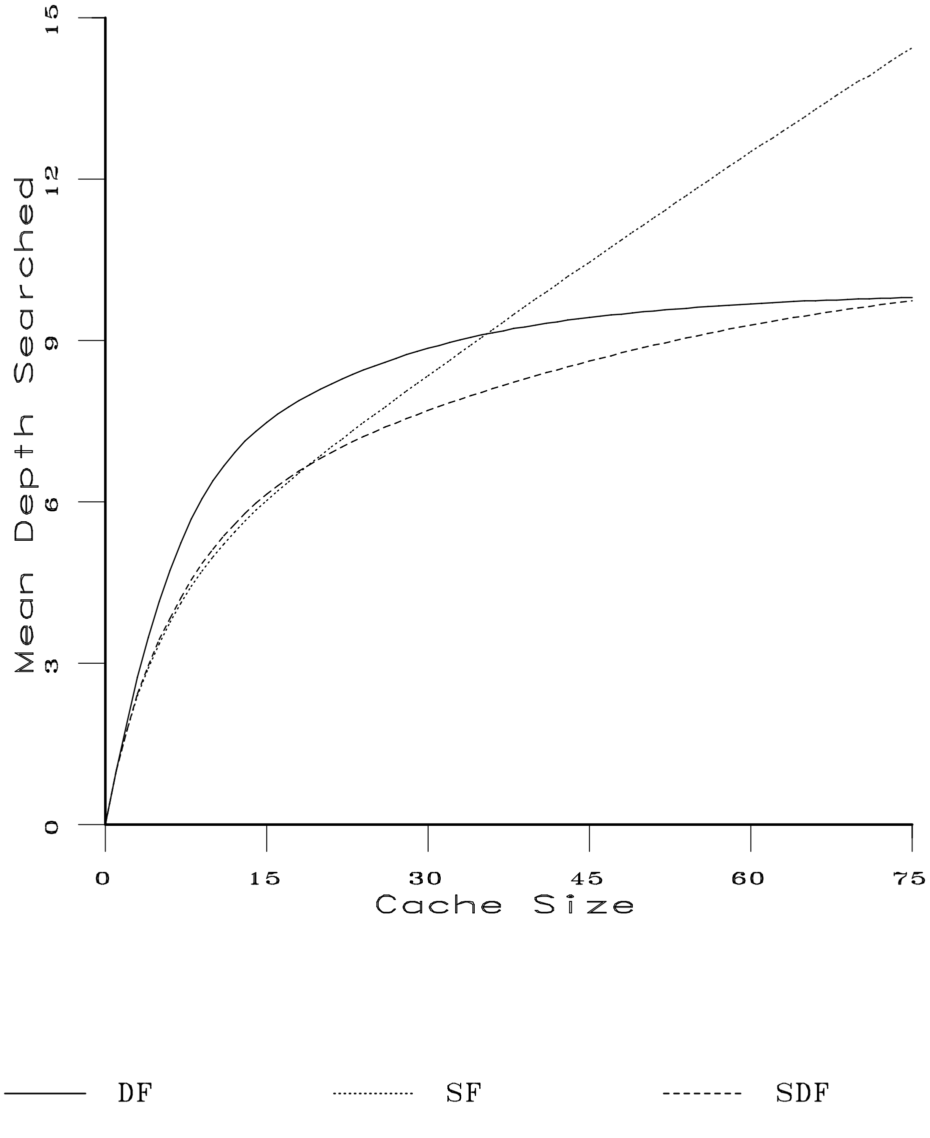

Figure 5.6: Mean Address Resolution Depth for Trace5

searched in case of a cache miss (the complete cache). The second effect dominates resulting in a steady increase. After stack positions 30-40, DF and SDF have levelled off. DF has a slightly higher total reference frequency than SDF, while SDF has higher reference frequency for the first few slots, causing the mean address resolution depth for DF and SDF to be almost the same.

These two measures, cache miss ratio and mean address resolution depth, show that for caches of size below 13, it is preferable to cache source addresses (as in the SF policy), while beyond this cache size, it is preferable to cache destination addresses (as in the DF policy). But the performance of a cache cannot be judged only by the miss ratio or the mean address resolution depth. As the cache becomes larger, the cost of performing a search and the cost of maintaining the LRU cache also increase. The desirable size of the cache therefore is one for which the cost of using the cache, coupled with the cost of searching the forwarding table in case of misses, is less than the cost of searching only the forwarding table every time. The next section addresses this issue.

5.4.4 Forwarding-Table Cache Performance Analysis

Having obtained miss ratio and depth of search values for each fetch policy, the next and final step is to compare the mean access time of a forwarding table with a cache to that of forwarding table without a cache. Given the smaller size and faster access method of the cache, a cache would have lower mean access time than the forwarding table. Even if the cache and the forwarding table use the same access method, say binary search, if the effect of locality is high enough, a reduction in overall access time should be observed with a cache simply because of the reduced search time. The lower the cost of the access, the lower is the mean frame forwarding delay. Hence performance is better. In order to compute the access time ratio of the forwarding table with and without a cache the following method was used.

Let n be the size of the forwarding table. The value of n for a particular trace is assumed to be the number of active hosts in the trace.

Let c be the size of the cache.

Then $N_f = 1 + log(n)$ is an estimate of the mean number of accesses of the forwarding table, (assuming binary search),

and $N_c = 1 + log(c)$ is an estimate of the mean number of accesses of the cache, (again, assuming binary search),

Let m be the cache miss ratio, for cache size c and a particular fetch policy.

Let S_c be the speed of the cache (in accesses/second) and S_f be the speed of the forwarding table.

Then $T_f = \frac{(1+log(n))}{S_f}$ is an estimate of the mean access time of the forwarding table per search, and $T_c = \frac{(1+log(c))}{S_c}$ is an estimate of the mean access time of the cache per search.

If there is a cache hit, the probability of which is $(1 - m)$, the lookup time is the mean access time of the cache. If there is a cache miss, the probability of which is m, the lookup time is the sum of the mean access time of the cache and of the forwarding table, since the lookup is performed in the cache first and when the entry is not found, the forwarding table is searched. So, the mean access time of the forwarding table with a cache, or the cost of searching the forwarding table with a cache, is given by:

$$T_{fc} = (1 - m)T_c + m(T_f + T_c)$$

This simplifies to $T_{fc} = T_c + m(T_f)$

The ratio of the costs associated with searching the forwarding table with and without a cache is

$$CR = \frac{T_{fc}}{T_f}$$

This is equivalent to

$$CR = \frac{(1-m)(1+log(c))S_f + m((1+log(c))S_f + (1+log(n))S_c)}{(1+log(n))S_c}$$

This simplifies to

$$CR = \frac{(1+log(c))}{(1+log(n))r} + m$$

where

$$r = \frac{S_c}{S_f}$$

is the speed ratio

Results obtained using the above equation with speed ratio of 1, i.e., with the speed of access of the cache being the same as the speed of access of the forwarding table ($S_c = S_f$), are shown using the three fetch policies, in Figure 5.7 through Figure 5.9 for the selected traces. In these figures, the cost ratio is plotted against cache size. Initially the cost ratios fall sharply for small increases in cache size, but then begin to increase. This is due to the steep fall in the cache miss ratios for small increases in the cache size initially, but after a certain point the decrease in miss ratios is small as the cache size increases. As the cache size increases, the mean number of accesses required to search the cache increases, while the mean number of accesses required to search the forwarding table is still the same. Although the miss ratios also decrease, the net effect is an increase in search time for the system as a whole. The point of maximum benefit is the minimum for each curve. The best results for all fetch policies are observed when cache sizes are in the range of 10-20.

For most of the traces, DF performs better than the other two fetch policies for cache sizes above 13. With DF, a cost ratio (CR) of 0.65 is achieved for cache sizes

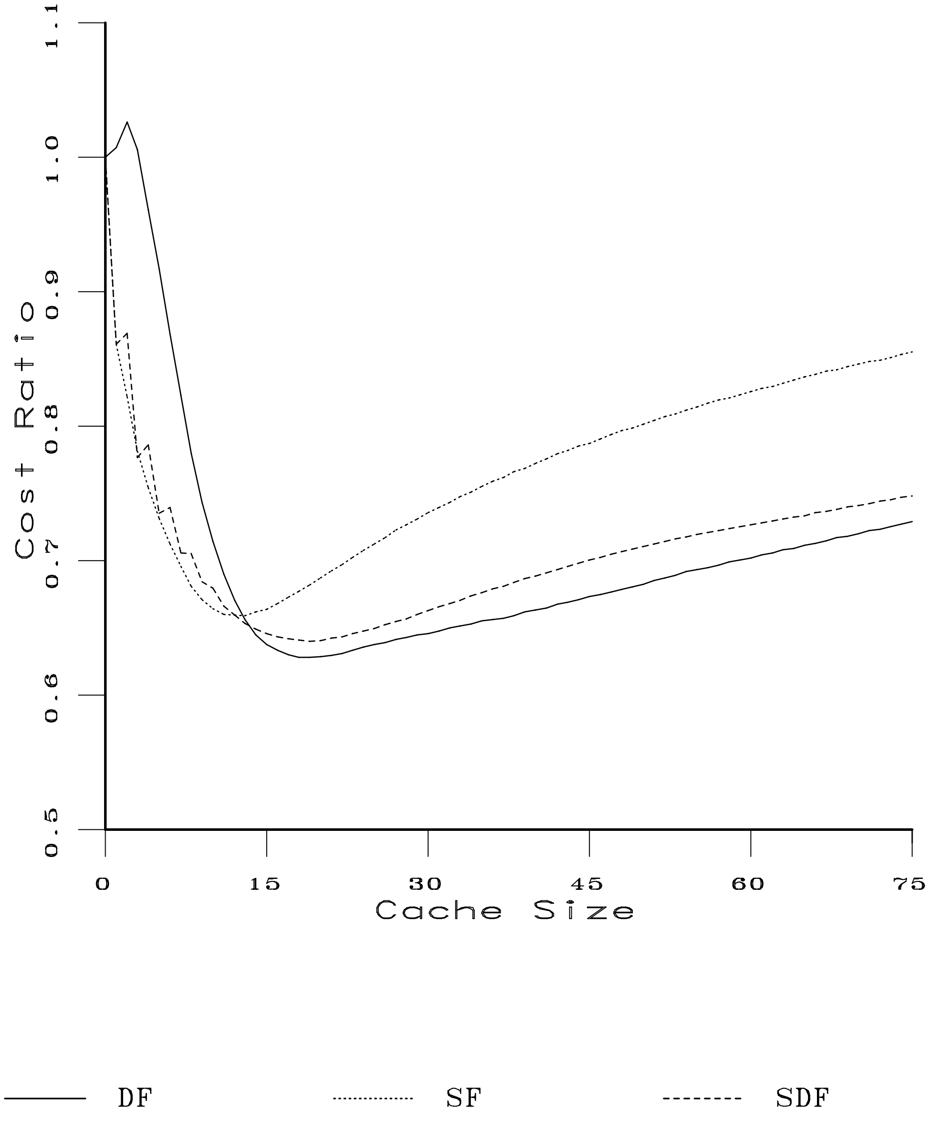

Figure 5.7: Cost Ratio as a Function of Cache Size for Trace5

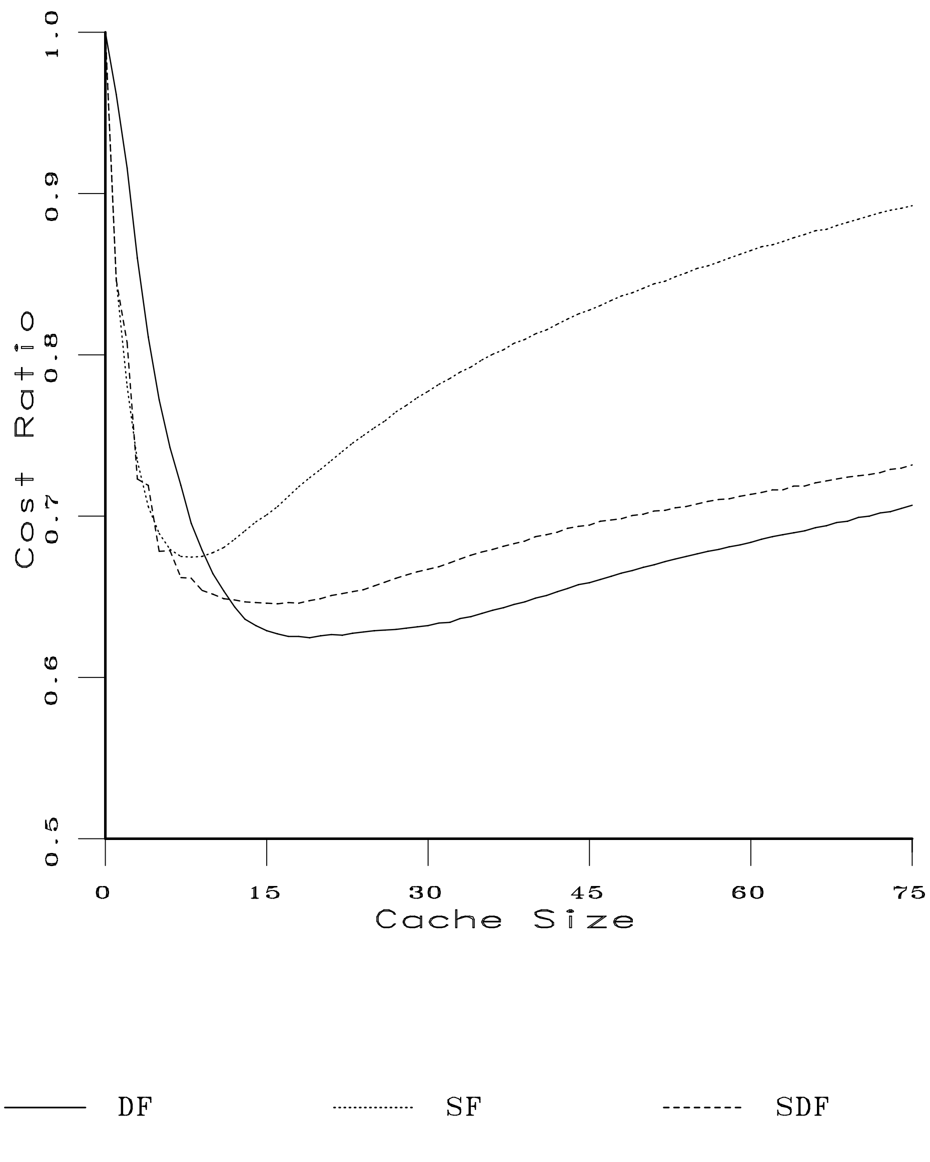

Figure 5.8: Cost Ratio as a Function of Cache Size for Trace6

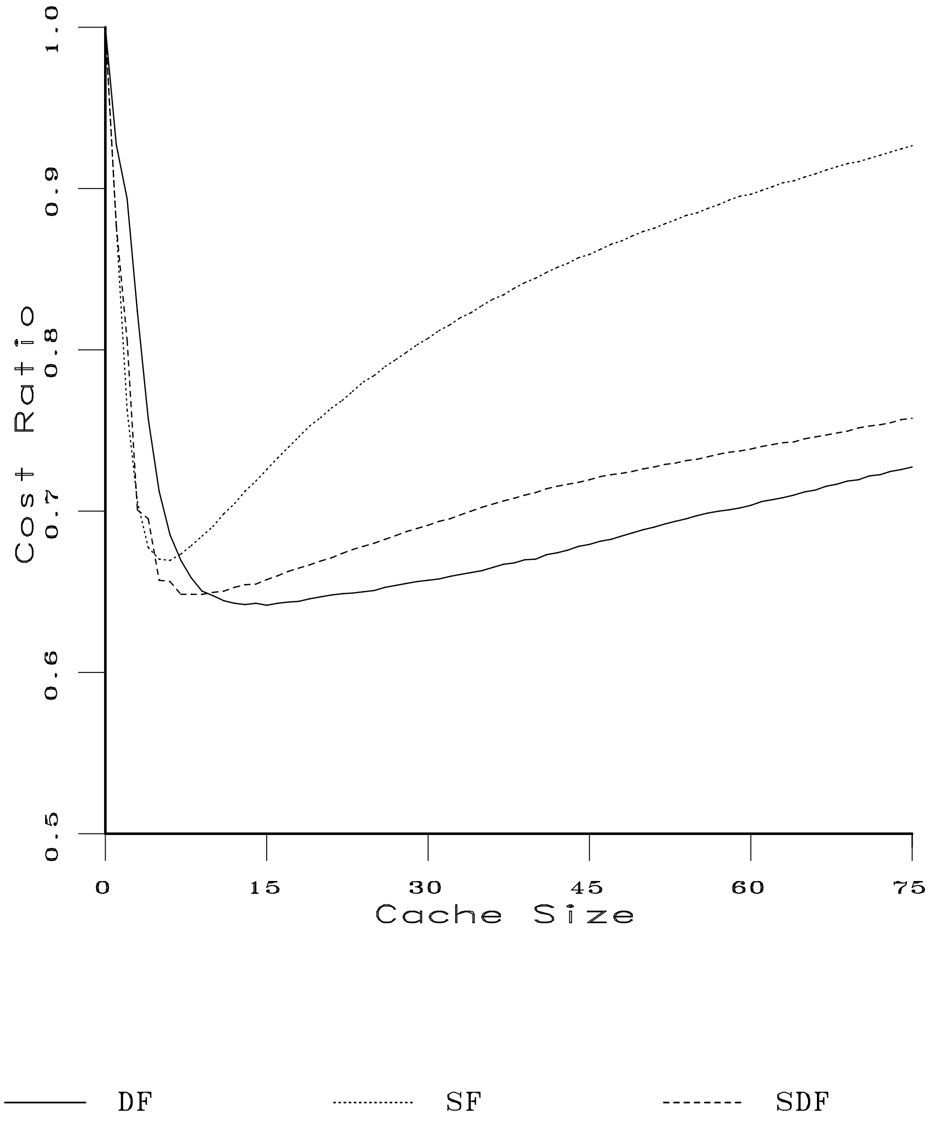

Figure 5.9: Cost Ratio as a Function of Cache Size for Trace8

in the range of 15-20. This means that a forwarding table with a cache of size 15-20, using the LRU replacement policy and the DF fetch policy, has a mean access time which is 65% of the mean access time of the forwarding table if it were not using a cache. This 35% reduction in mean access time is a considerable improvement in performance, and is obtained under the very conservative assumption that the cache has the same speed of access as the forwarding table. SF and SDF also result in improvement, but to a lesser degree. In fact, the performance of SF degrades quite rapidly beyond a cache size of 20. The cost ratio depends on the cache miss ratio (m) and the ratio of mean access times (r). The performance of SDF is slightly worse than that of DF.

It is not uncommon for a cache to provide higher speed of access than memory. Figure 5.10 shows the effect of different speed ratios on the cost ratios for Trace5 with the DF fetch policy. The results show that increasing the speed ratio reduces the cost ratio and shifts the "knee" of the curve to the right (towards higher cache sizes). Increasing the speed ratio from 1 to 3 causes almost 50% reduction in the cost ratio (from a cost improvement of 35% to a cost improvement of 70%). Changing the speed ratio from 3 to 5 or from 5 to 10, however, does not result in such a large performance gain because the cost of searching the forwarding table now becomes the dominating factor. Clearly a large gain can be achieved at relatively low cost by implementing a cache that is three times faster than the forwarding table. The additional benefit of an even faster cache is less significant.

In general the conclusion is that using a forwarding-table cache can result in substantial reduction in mean lookup time at the interconnection device. Under quite modest assumptions, gains of up to 70% can be achieved.

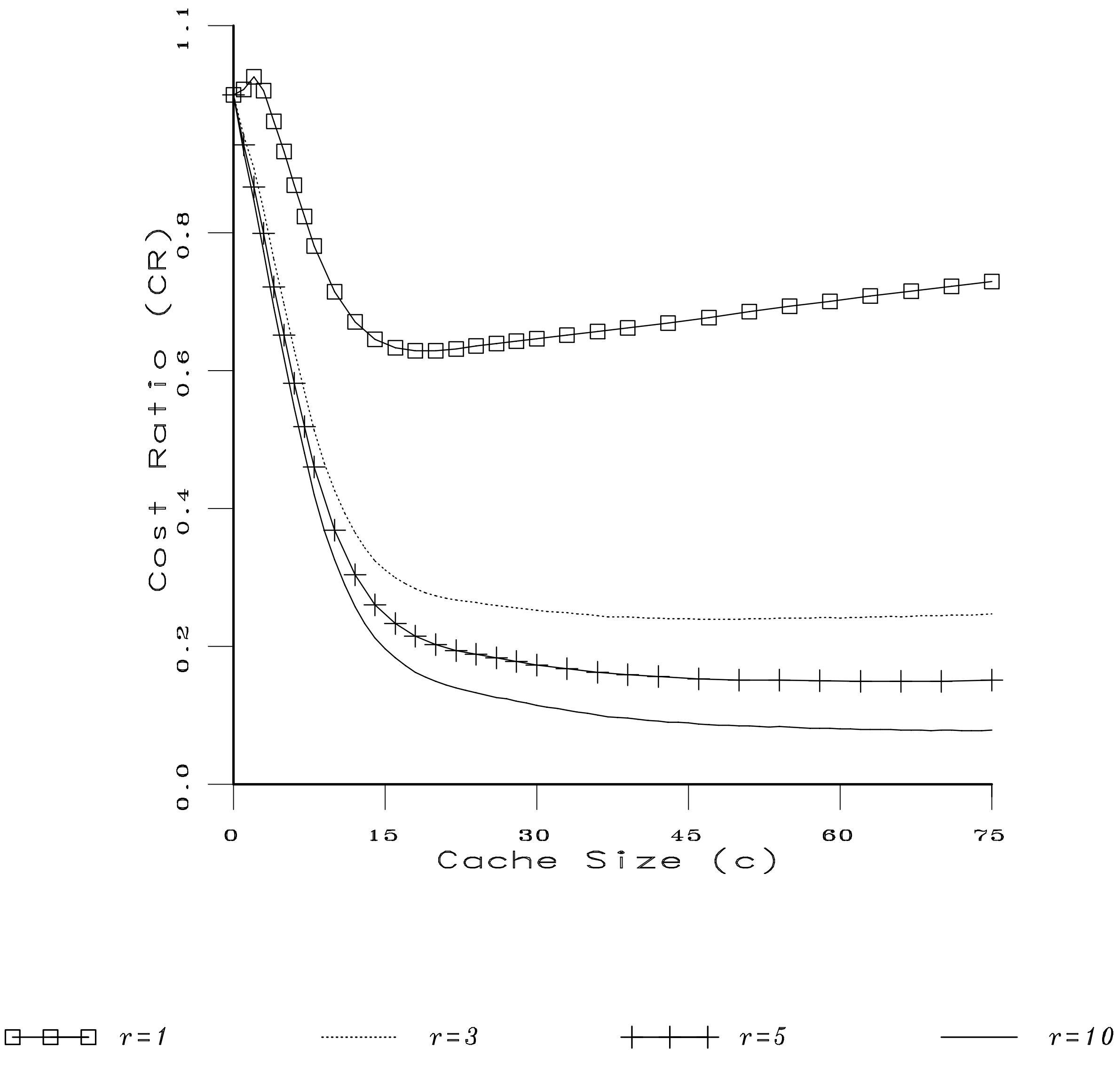

Figure 5.10: Cost Ratios for different Speed Ratios and Cache Sizes for Trace5

5.5 Summary

In this chapter an analysis of the performance of a forwarding-table cache was presented.

In order to determine the benefit in performance with respect to the lookup time of the forwarding table with and without a cache, some cache design parameters had to be selected. These included the cache replacement policy and the cache fetch policy. Five cache replacement policies were simulated to evaluate their relative performance, in terms of the cache miss ratios. These were LRU, LFU, FIFO, Rand and MIN. LRU performs better than all other implementable policies and within 20% of the optimal MIN policy. Although LRU is the best among the implementable policies considered, there is still room for improvement.

In the analysis of characteristics of local area network traffic locality in Chapter 4, it was concluded that both the source address and the destination address of the recently received frames are useful in predicting the destination of future arrivals. Based on this conclusion, three fetch policies were selected for a forwarding-table cache. The first policy (DF) caches the destination address, the second (SF) caches the source address, and the third (SDF) caches both the source address and the destination address of the current frame.

An LRU stack reference frequency analysis confirmed the suitability of LRU as the replacement policy for the cache with each of the three policies. The reference frequencies obtained showed an almost monotonic fall with increasing stack depth along with high cumulative reference frequency. This showed that the last slot on the LRU stack is almost always the least frequently referenced position. Therefore when a entry has to be removed from the cache, it should be removed from this last slot. These results illustrated that LRU is indeed the appropriate replacement policy for the references under consideration. Hence LRU was selected as the replacement policy. Another important insight obtained from the LRU stack reference frequency analysis was that by using SF, the top 13 positions on the stack have a much higher reference frequencies than in the case of DF or SDF. This is clearly an environment-specific figure that comes from the high degree of source address re-use observed for window sizes up to 13. For stack positions beyond 13, the reference frequency for SF falls sharply, because the older entries have a much lower degree of re-use.

Using trace-driven simulation, cache performance measures, namely cache miss ratio and mean address resolution depth, were obtained for a range of cache and the three fetch policies. For cache sizes less than 13 it was shown to be best to cache source addresses (as in the SF policy). Beyond this size it was shown to be the best to cache destination addresses (as in the DF policy). This is because of the low degree of re-use of source addresses beyond the 13 popular hosts. For larger cache sizes destination address re-use dominates.

A final analysis was performed to judge the reduction in the mean lookup time achievable by adding a cache to the forwarding table. For the DF fetch policy, a 70% reduction in mean lookup time can be achieved by using a cache that is only three times faster than the forwarding table. This is a substantial gain at a very modest cost.

Chapter 6

Conclusions

The advent of computer networks has revolutionized the way computing environments are organized. The old model of a single large computer serving all the users is quickly being replaced by one which has autonomous workstations connected by a local area network. Such an organization offers sharing of resources, higher reliability and lower cost for the same level of performance. It also provides an excellent communication medium for the users.

There has been considerable interest in connecting LANs into internetworks for a number of reasons. Connecting together isolated LANs installed in different sections within the same site enhances the availability of resources and extends the communication medium. It also provides better performance (because of isolation of local traffic on each LAN) and better security than a single large network.

The design of devices connecting LANs is important. When connecting broadcast networks (e.g., Ethernet), data link level interconnection devices must search their forwarding tables for each frame travelling on each LAN they are connected to. Since these devices operate at a low level, they cannot participate in congestion or flow control, so they have to perform their responsibility fast enough to maintain their transparent operation.

As internetworks get larger, the number of potential entries that might exist in the forwarding table of a data link level interconnection device also increases. The increase in bandwidth would further reduce the time these devices have to process each frame. In order to keep pace with these advances, the design of such devices has to be improved [11, 20, 21, 23].

6.1 LAN Traffic Locality: Characterization

The locality behaviour of traffic on computer networks has been actively investigated in recent years. Researchers have observed that this behaviour holds considerable promise with respect to making network implementations and communication processing at various levels more efficient. An initial analysis showed that considerable locality does exist in the traffic on an operational LAN. The work presented in this thesis focussed on characterizing the salient aspects of LAN traffic locality, which is essential if this behaviour is to be exploited to its fullest potential. Based on these characteristics, the improvement in performance of a data link layer interconnection device that takes the locality of traffic into account is estimated.

The aspects of locality identified for characterization were considered in three categories: destination address characteristics, source address characteristics, and host-pair characteristics. Traffic traces from an operational LAN were used for the analysis. The traffic was found to have low persistence but to be high in both source and destination address re-use. High concentration and reference density were found for both the source and the destination addresses. The traces were found to have a high level of host-pair locality too. Besides quantifying each aspect, the analysis brings forward an important aspect namely that in order to predict the destinations of future references, the source addresses of the current and last few frames can be very helpful, and in fact more helpful than the destination addresses.

6.2 LAN Traffic Locality: Application

In order to reduce the mean lookup time of an interconnection device, given the locality behaviour of the traffic, a forwarding-table cache could be used to perform quicker lookup of the addresses. This issue, discussed in Chapter 5, has various facets. Different replacement policies and fetch policies can be used, besides varying sizes for the forwarding table cache. Using a discrete-event trace-driven simulation model, five different replacement policies were studied, namely LRU, LFU, FIFO, Rand and MIN. Of the implementable policies, LRU resulted in the lowest miss ratios. LRU results in miss ratios 20% higher than those caused by MIN, which is the optimal replacement policy. This shows that there is room for improvement as far as the replacement policies are concerned. Using LRU as the replacement policy, three different fetch policies and cache size varying from 1 to 75, the performance with respect to cache miss ratio and the mean address resolution depth was determined. For cache sizes below 13, the source fetch policy (SF) results in the lowest miss ratios and mean address lookup time, but beyond this cache size the destination fetch policy (DF) performs the best.

An analysis was then performed to determine the benefit with respect to mean lookup time that can be achieved by using a forwarding-table cache. Results obtained with the LRU replacement policy and the DF fetch policy show that a 70% reduction in the lookup time can be attained by using a cache that is only three times faster than the forwarding table.

This research provides a detailed analysis of various aspects of local area network traffic locality. This insight should be of value for future research in this area.

6.3 Directions for Future Research

One possible application of network traffic locality has been studied in this thesis and it has been found that exploiting locality can result in improved performance of interconnection devices. With networks getting faster and larger, such optimizations are becoming increasingly important. As discussed in Chapter 3, locality of traffic can be exploited at various levels. The application considered here dealt with the data link level. Gateways that operate at the network level can also take advantage of the locality characteristics of the traffic. At the per-process level, the use of this property can result in improvements in communication processing.

Cut-through bridging is a technique that can also be used to improve the efficiency of interconnection devices [23]. A technique that predicts future arrivals based on the locality aspects and uses the cut-through approach for frame forwarding would be expected to perform well. A comparative study of the locality based approach discussed in this thesis and the cut-through method is another area for future work.

Using the same framework, characterization of locality using traffic from other environments would give more insight into the dependence of the various aspects of locality on such parameters as the total number of hosts and the number of file servers on the network.

The locality of the traffic on a network is strongly related to the protocols being used. Implementation of protocols can be made more efficient by taking the locality of traffic into account, as shown by the optimistic implementation of a bulk data transfer protocol by Carter and Zwaenepoel [7]. Further design and implementation of protocols must take the locality factor into account since enhanced locality can result in a number of useful optimizations, one of which has been demonstrated in this thesis.

References

[1] Agarwal A., *Analysis of Cache Performance for Operating Systems and Multiprogramming*, Kluwer Academic Publishers, Norwell, MA, 1989.

[2] Amer P. D., "A Measurement Center for the NBS Local Area Computer Network", *IEEE Transactions on Computers*, Vol. C-31, August 1982, pp. 723-729.

[3] Belady L. A., "A Study of Replacement Algorithms for a Virtual-Storage Computer", *IBM Systems Journal*, Vol. 5, No. 2, 1966, pp. 78-101.

[4] Black U., *Computer Networks – Protocols, Standards and Interfaces*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[5] Bunt R. B., Harbus R. S. and Plumb S. J., "The Effective Management of Paging Storage Hierarchies", *Proceedings of 1982 SIGMETRICS Conference*, Seattle, August 1982, pp. 30-35.

[6] Bunt R. B. and Murphy J. M., "The Measurement of Locality and the Behaviour of Programs", *Computer Journal*, Vol. 27, No. 3, 1984, pp. 238-245.

[7] Carter J. B. and Zwaenepoel W., "Optimistic Implementation of Bulk Data Transfer Protocols", *Proceedings of 1989 SIGMETRICS Conference*, Berkeley, California, May 1989, pp. 61-69.

[8] Cheriton D. R. and Williamson C. L., "Network Measurement of the VMTP Request-Response Protocol in the V Distributed System", *Proceedings of 1987 SIGMETRICS Conference*, Banff, Alberta, May, 1987, pp. 128-140.

[9] Denning P. J., "Working Set Model for Program Behavior", *Communications of the ACM*, Vol. 11, No. 5, May 1968, pp. 323-333.

[10] Denning P. J., "Virtual Memories", *Computing Surveys*, Vol. 2, No. 3, September 1970, pp. 153-189.

[11] Feldmeier D. C., "Estimated Performance of a Gateway Routing-Table Cache", MIT/LCS/TM-350, March 1988.

[12] Feldmeier D. C., "Improving Gateway Performance with a Routing-Table Cache", *Proceedings of IEEE Infocom '88*, New Orleans, March 1988, pp. 298-307.

[13] Ferrari D., "Improving Locality by Critical Working Sets", *Communications of the ACM*, Vol. 17, No. 2, November 1974, pp. 614-620.

[14] Floyd B., "Transparent Bridges for Interconnection of IEEE 802 LANs", *IEEE Network Magazine*, January 1988, Vol. 2, pp. 5-9.

[15] Garg V. K. and Ramamoorthy C. V., "Effect of Locality in Large Networks", *Proceedings of the 7th International Conference on Distributed Computing Systems*, Berlin, West Germany, September 1987, pp. 544-550.

[16] Gulati N., "Interconnection of Local Area Networks Using Bridges", *Proceedings of the Third Annual Graduate Symposium on Computational Science*, Department of Computational Science, Saskatoon, Canada, April 1991, pp. 107-121.

[17] Gulati N., "Network Locality Analysis and Design of a Locality-Based Bridge", *Proceedings of the 7th Australian Computer Science Conference*, Hobart, Australia, January 1992.

[18] Hewlett Packard Inc., "HP4972A LAN Performance Analyser-User's Guide", Colorado Telecommunications Division, September 1989.

[19] Hussien T. M. and Rene P. S., "Distributed Discrete Event Simulation for Communication Networks", *IEEE Journal on Selected Areas in Communication*,Vol. SAC-8, No. 9, December 1990, pp. 1723-1733.

[20] Jain R. and Routhier S. A., "Packet Trains – Measurement and a New Model for Computer Network Traffic", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, No. 6, September 1986, pp. 986-995.

[21] Jain R., "Characteristics of Destination Address Locality in Computer Networks: A Comparison of Caching Schemes", DEC-TR-592, February 1989.

[22] Klienrock G. M., "Congestion Control in Interconnected LANs", *IEEE Network Magazine*, January 1988, Vol. 2, pp. 15-21.

[23] Kwok C. C. and Mukherjee B., "Cut-through Bridging for CSMA/CD LANs", *IEEE Transactions on Communications*, Vol. 38, No. 7, July 1990, pp. 938-942.

[24] Ling S. H., "Characterizing and Modelling the Workload of Ethernet Local Area Networks", MSc. Thesis, Department of Computational Science, University of Saskatchewan, 1992.

[25] Lisa F. P., "A Survey of Discrete Event Simulation and State-of-the-art Discrete Event Languages", *Proceeding of SIGSIM Conference 1989*, Boulder, CO, May 1989, pp. 8-24.

[26] Lorence M. J. and Satyanarayanan M., "IPwatch: A Tool For Monitoring Network Locality", *Operating Systems Review*, Vol. 24, No. 1, January 1990, pp. 58-80.

[27] Maisel H. and Gnugnoli G., *Simulation of Discrete Stochastic Systems*, Scientific Research Associates, Inc., Palo Alto, CA, 1972.

[28] Majumdar S., "Locality and File Referencing Behaviour: Principles and Applications", MSc. Thesis, Department of Computational Science, University of Saskatchewan, 1984.

[29] Mogul C. J., "Network Locality at the Scale of Processes", *Proceedings of ACM SIGCOMM '91*, Zurich, Switzerland, September 1991, pp. 273-284.

[30] Pei T. and Zukowski C., "Putting Routing Tables in Silicon", *IEEE Network Magazine*, January 1992, pp. 42-50.

[31] Przybylski S. A., *Cache and Memory Hierarchy Design : A Performance-Directed Approach*, Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[32] Schwartz M., *Telecommunications Networks - Protocols, Modelling and Analysis*, Addison-Wesley, Reading, MA, 1988.

[33] Seifert W. M., "Bridges and Routers", *IEEE Network Magazine*, January 1988, Vol. 2, pp. 31-38.

[34] Smith A. J., "Sequentiality and Prefetching in Database Systems", *ACM Transactions on Database Systems*, September 1978, pp. 223-247.

[35] Smith A. J., "Cache Memories", *Computing Surveys*, Vol. 14, No. 3, September 1982, pp. 473-530.

[36] Smith A. J., "Miss Ratio Analysis and Design Consideration", *ACM Transactions on Computer Systems*, Vol. 3, No. 3, August 1985, pp. 161-203.

[37] Tanenbaum A. S., *Computer Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[38] Vitalink Communications Corporation, "Building and Managing Multivendor Networks using Bridge and Router Technologies", June 1990.

Appendix A

This appendix contains the complete set of results, for each trace, for the characteristics of locality discussed in Chapter 4.

Table A.1: Destination Address Persistence

w	1	2	3	4	5	6	7	8	9	10
Trace1	14.71	7.79	5.24	3.61	2.39	1.41	1.14	0.96	0.01	0.00
Trace2	14.45	6.17	3.20	2.02	1.28	0.74	0.61	0.53	0.01	0.00
Trace3	10.29	3.37	1.91	1.21	0.68	0.31	0.25	0.20	0.01	0.00
Trace4	6.54	2.20	1.33	0.91	0.63	0.43	0.38	0.34	0.01	0.00
Trace5	9.20	3.47	1.84	1.13	0.76	0.52	0.44	0.39	0.01	0.00
Trace6	13.51	5.75	3.52	2.47	1.77	1.26	1.11	1.01	0.01	0.00
Trace7	10.97	5.72	3.98	2.61	1.46	0.46	0.37	0.30	0.00	0.00
Trace8	17.16	5.70	3.64	2.52	1.63	0.95	0.83	0.77	0.00	0.00
Trace9	13.26	5.84	3.33	1.94	1.10	0.50	0.39	0.34	0.00	0.00
Trace10	24.06	7.95	4.72	2.85	1.41	0.35	0.22	0.14	0.00	0.00

Table A.2: Destination Address Re-use

w	1	2	3	4	5	6	7	11	15	19
Trace1	14.71	24.25	32.24	38.61	43.56	47.74	51.06	59.35	64.96	68.05
Trace2	14.45	24.15	31.42	37.20	41.82	45.70	48.92	57.45	63.04	66.14
Trace3	10.29	26.57	38.03	46.13	51.70	55.92	59.09	66.74	71.57	74.06
Trace4	6.54	19.14	28.62	35.50	41.03	45.59	49.36	57.66	62.78	65.13
Trace5	9.20	16.57	23.52	30.56	36.61	41.96	46.48	58.17	66.48	70.52
Trace6	13.51	26.34	35.94	42.97	48.47	52.52	55.92	63.59	68.36	71.44
Trace7	10.97	20.64	27.77	33.30	37.99	42.14	45.82	54.82	61.14	65.00
Trace8	17.16	28.50	38.84	47.07	53.43	58.15	61.50	67.36	70.95	73.09
Trace9	13.26	20.52	26.82	32.50	37.45	41.44	44.73	53.05	58.67	62.83
Trace10	24.06	41.85	56.66	64.97	69.34	72.03	73.82	77.58	79.29	80.46

Table A.3: Destination Address Working Set Sizes for Different Window Sizes

w	10	30	60	100	150	200	300	400	500
Trace1	6	12	17	22	26	30	36	41	46
Trace2	6	12	18	23	29	33	40	44	48
Trace3	5	10	15	19	23	26	32	36	40
Trace4	6	12	18	24	29	33	39	45	49
Trace5	6	12	16	21	24	28	33	37	40
Trace6	5	11	16	21	25	29	35	39	42
Trace7	6	13	18	23	27	32	38	43	47
Trace8	5	10	15	20	24	29	35	39	43
Trace9	6	13	20	25	29	33	39	43	47
Trace10	4	8	11	15	18	21	26	30	34

Table A.4: Destination Address Concentration

w	10	30	60	100	150	200	300	400	500
Trace1	0.44	0.66	0.75	0.79	0.83	0.85	0.89	0.90	0.91
Trace2	0.44	0.59	0.71	0.78	0.81	0.84	0.87	0.89	0.91
Trace3	0.56	0.69	0.76	0.82	0.85	0.87	0.89	0.91	0.92
Trace4	0.44	0.62	0.71	0.77	0.81	0.84	0.87	0.89	0.90
Trace5	0.44	0.62	0.74	0.79	0.84	0.86	0.89	0.91	0.92
Trace6	0.56	0.66	0.75	0.79	0.84	0.86	0.89	0.90	0.92
Trace7	0.44	0.59	0.71	0.78	0.83	0.84	0.88	0.89	0.91
Trace8	0.56	0.69	0.76	0.81	0.85	0.86	0.89	0.90	0.91
Trace9	0.44	0.59	0.68	0.76	0.81	0.84	0.87	0.89	0.90
Trace10	0.67	0.76	0.83	0.86	0.89	0.89	0.92	0.93	0.93

Table A.5: Destination Address Reference Density

No.	R_1	R_3	R_5	R_{10}	R_{15}	R_{20}	R_{25}
Trace1	14.33	30.93	46.49	62.47	71.84	78.29	83.18
Trace2	8.69	24.57	35.01	53.99	67.90	76.16	81.77
Trace3	20.84	43.46	56.61	71.15	78.85	84.60	88.31
Trace4	13.10	32.69	43.72	63.57	73.85	80.77	86.09
Trace5	9.38	27.45	39.96	64.98	80.17	86.33	90.00
Trace6	8.02	22.97	35.83	59.08	72.68	80.41	85.65
Trace7	13.89	33.24	43.46	64.36	77.15	84.58	89.30
Trace8	13.46	34.21	42.49	58.14	69.64	78.20	84.13
Trace9	14.33	30.29	40.45	57.69	70.35	79.08	84.88
Trace10	29.99	64.94	72.54	84.22	89.25	92.55	94.52

Table A.6: Source Address Persistence

w	1	2	3	4	5	6	7	8	9
Trace1	12.77	6.68	4.27	2.76	1.61	0.72	0.51	0.38	0.00
Trace2	12.95	5.53	2.81	1.65	0.92	0.40	0.26	0.19	0.00
Trace3	9.52	3.03	1.63	0.96	0.45	0.10	0.06	0.03	0.00
Trace4	5.58	1.69	0.88	0.49	0.23	0.07	0.05	0.04	0.00
Trace5	8.31	2.95	1.29	0.60	0.28	0.08	0.05	0.03	0.00
Trace6	11.98	4.52	2.35	1.36	0.73	0.30	0.21	0.14	0.00
Trace7	10.01	5.21	3.55	2.25	1.12	0.17	0.09	0.06	0.00
Trace8	16.05	5.41	2.99	1.91	1.06	0.43	0.33	0.27	0.00
Trace9	12.95	5.66	3.10	1.73	0.90	0.30	0.18	0.12	0.00
Trace10	23.36	8.55	5.41	3.54	2.09	1.04	0.62	0.40	0.00

Table A.7: Source Address Re-use

w	1	2	3	4	5	6	7	11	15	19
Trace1	19.20	31.90	39.88	45.19	49.31	52.60	55.31	62.85	67.29	69.95
Trace2	18.71	30.24	37.86	43.29	47.47	50.89	53.69	61.32	65.59	67.72
Trace3	28.64	43.11	51.63	57.04	61.19	64.32	66.77	72.95	76.33	77.87
Trace4	26.37	38.38	44.97	49.33	52.49	54.96	56.97	61.84	64.38	65.54
Trace5	23.85	37.11	45.38	51.18	55.47	58.95	61.80	69.90	74.61	76.85
Trace6	24.97	38.32	46.56	52.16	56.16	59.20	61.57	67.60	70.62	72.50
Trace7	21.96	32.11	37.96	42.27	45.62	48.22	50.41	57.11	61.66	64.40
Trace8	22.18	38.64	48.88	55.34	59.20	61.96	64.00	68.75	71.05	72.46
Trace9	14.17	26.02	34.46	40.44	45.02	48.81	51.86	60.02	65.16	68.69
Trace10	31.99	51.37	60.57	65.15	68.06	70.13	71.96	76.41	78.97	80.46

Table A.8: Source Address Working Set Sizes for Different Window Sizes

w	10	30	60	100	150	200	300	400	500
Trace1	6	13	21	30	40	50	68	84	98
Trace2	6	14	22	31	41	51	67	82	95
Trace3	6	11	17	24	31	37	50	62	73
Trace4	6	14	23	34	46	58	78	95	108
Trace5	6	13	18	25	32	38	50	61	71
Trace6	6	12	20	28	38	46	62	75	88
Trace7	7	14	23	34	41	51	67	82	95
Trace8	6	14	22	31	46	58	77	93	106
Trace9	6	14	21	28	35	41	52	61	69
Trace10	4	9	14	20	28	35	47	60	71

Table A.9: Source Address Concentration

w	10	30	60	100	150	200	300	400	500
Trace1	0.44	0.59	0.66	0.71	0.74	0.75	0.77	0.79	0.81
Trace2	0.44	0.55	0.64	0.69	0.73	0.75	0.78	0.79	0.81
Trace3	0.44	0.66	0.73	0.77	0.79	0.82	0.84	0.85	0.86
Trace4	0.44	0.55	0.63	0.67	0.69	0.71	0.74	0.76	0.79
Trace5	0.44	0.59	0.71	0.76	0.79	0.81	0.84	0.85	0.86
Trace6	0.44	0.62	0.68	0.73	0.75	0.77	0.79	0.81	0.83
Trace7	0.33	0.55	0.63	0.67	0.69	0.71	0.75	0.77	0.79
Trace8	0.44	0.62	0.69	0.74	0.77	0.78	0.79	0.82	0.83
Trace9	0.44	0.62	0.66	0.73	0.78	0.80	0.83	0.85	0.88
Trace10	0.67	0.72	0.78	0.81	0.82	0.83	0.85	0.85	0.86

Table A.10: Source Address Reference Density

No.	R_1	R_3	R_5	R_{10}	R_{15}	R_{20}	R_{25}
Trace1	15.08	31.73	46.02	57.40	65.45	71.48	75.03
Trace2	9.38	25.62	38.05	51.90	61.99	69.38	74.53
Trace3	20.51	42.14	55.29	69.95	76.66	80.55	82.83
Trace4	13.08	34.07	42.58	55.40	63.73	69.22	71.86
Trace5	9.38	26.80	39.76	64.78	77.32	82.03	84.84
Trace6	8.31	24.31	36.72	56.25	67.56	73.70	77.27
Trace7	13.86	37.39	48.68	59.81	65.57	68.51	70.51
Trace8	19.03	33.84	45.14	57.59	66.50	71.53	75.16
Trace9	14.93	30.03	39.88	58.16	69.30	76.54	82.08
Trace10	32.26	67.72	74.28	80.21	83.60	84.99	86.06

Table A.11: Host-Pair Locality

w	1	2	3	4	5	6	7	11	15	19
Trace1	28.81	42.22	49.54	53.88	56.97	59.21	61.02	65.58	68.51	70.57
Trace2	28.66	41.14	47.98	52.38	55.38	57.62	59.38	63.67	66.57	68.50
Trace3	36.23	51.48	58.92	63.29	66.26	68.41	69.99	73.75	76.29	77.88
Trace4	29.84	41.50	47.30	51.14	53.91	55.97	57.78	61.33	63.42	64.75
Trace5	30.89	44.16	51.56	56.49	59.97	62.84	65.21	71.38	75.24	77.59
Trace6	35.21	48.82	55.61	59.80	62.47	64.40	65.90	69.73	71.88	73.32
Trace7	29.99	40.34	45.89	49.85	52.83	55.08	56.94	61.37	64.15	65.85
Trace8	34.65	49.38	57.13	61.70	64.36	66.10	67.28	69.88	71.38	72.39
Trace9	23.01	36.33	44.71	50.15	53.91	56.82	59.16	63.78	66.84	69.19
Trace10	52.06	66.96	72.39	74.85	76.32	77.28	78.18	79.80	80.69	81.29

Appendix B

This appendix contains the complete set of results, for each trace, for the analyses discussed in Chapter 5.

Table B.1: Reference Frequency for the DF Policy

n	1	2	3	4	5	10	20	40	75
Trace1	14.72	10.16	9.42	8.29	6.64	3.26	0.79	0.14	0.08
Trace2	14.45	10.55	8.69	7.55	6.40	3.33	0.80	0.27	0.09
Trace3	10.29	17.86	13.56	10.12	7.28	2.81	0.51	0.25	0.06
Trace4	6.54	13.13	10.75	8.77	7.13	3.40	0.69	0.25	0.06
Trace5	9.20	8.19	7.84	8.59	7.66	4.46	0.70	0.16	0.03
Trace6	13.49	14.14	11.26	8.81	7.02	2.91	0.65	0.16	0.04
Trace7	10.97	9.98	8.12	7.18	6.39	3.89	0.79	0.19	0.02
Trace8	17.13	13.33	12.77	10.86	7.63	1.88	0.57	0.25	0.03
Trace9	13.25	7.72	7.25	7.26	6.31	3.02	1.92	0.22	0.06
Trace10	23.18	25.14	15.26	8.98	5.02	1.28	0.41	0.10	0.02

Table B.2: Reference Frequency for the SF Policy

n	1	2	3	4	5	10	20	40	75
Trace1	19.20	16.59	11.92	6.42	5.64	1.65	0.15	0.04	0.01
Trace2	18.72	16.39	10.30	6.57	5.09	1.66	0.19	0.04	0.01
Trace3	28.64	12.71	13.93	7.33	6.18	1.30	0.11	0.02	0.01
Trace4	26.38	10.06	9.84	4.94	3.87	1.18	0.16	0.05	0.01
Trace5	23.86	10.84	13.28	5.50	6.31	2.34	0.22	0.02	0.01
Trace6	24.95	15.39	11.89	6.50	5.06	1.18	0.14	0.04	0.02
Trace7	21.95	14.24	8.24	5.09	4.07	1.24	0.14	0.03	0.01
Trace8	22.14	20.81	12.76	6.62	4.65	0.75	0.12	0.03	0.01
Trace9	14.16	14.40	12.08	7.02	7.01	2.24	0.75	0.027	0.01
Trace10	32.70	28.25	11.04	5.69	2.53	0.34	0.05	0.02	0.00

Table B.3: Reference Frequency for the SDF Policy

n	1	2	3	4	5	10	20	40	75
Trace1	19.20	14.72	14.09	4.39	7.55	1.98	0.68	0.16	0.04
Trace2	18.72	14.45	12.24	5.25	6.64	1.97	0.75	0.18	0.11
Trace3	28.64	10.29	15.33	5.99	7.24	1.67	0.43	0.13	0.04
Trace4	26.37	6.54	13.07	3.00	6.00	2.35	0.80	0.20	0.10
Trace5	23.86	9.21	15.07	3.26	8.28	1.99	0.68	0.14	0.04
Trace6	24.95	13.49	14.13	4.37	7.14	1.69	0.59	0.19	0.06
Trace7	21.95	10.97	11.32	3.02	6.15	2.25	0.92	0.23	0.07
Trace8	22.13	17.12	16.28	4.70	7.02	1.43	0.50	0.19	0.07
Trace9	14.15	13.25	13.11	5.39	8.41	2.03	1.00	0.19	0.05
Trace10	32.70	23.18	13.72	5.58	4.24	0.98	0.41	0.13	0.02

Table B.4: Cache Miss Ratios for the DF Policy

n	1	2	3	4	5	10	20	40	75
Trace1	0.85	0.75	0.65	0.57	0.50	0.28	0.12	0.05	0.01
Trace2	0.85	0.75	0.66	0.58	0.52	0.31	0.14	0.05	0.01
Trace3	0.89	0.71	0.58	0.48	0.40	0.21	0.08	0.03	0.01
Trace4	0.93	0.80	0.69	0.60	0.53	0.30	0.15	0.05	0.01
Trace5	0.90	0.82	0.74	0.66	0.58	0.28	0.09	0.03	0.01
Trace6	0.86	0.72	0.61	0.52	0.45	0.24	0.11	0.04	0.01
Trace7	0.89	0.79	0.70	0.63	0.57	0.31	0.13	0.05	0.01
Trace8	0.82	0.69	0.56	0.45	0.38	0.21	0.11	0.04	0.01
Trace9	0.86	0.79	0.71	0.64	0.58	0.38	0.16	0.04	0.01
Trace10	0.76	0.51	0.36	0.27	0.22	0.12	0.06	0.02	0.01

Table B.5: Cache Miss Ratios for the SF Policy

n	1	2	3	4	5	10	20	40	75
Trace1	0.80	0.66	0.55	0.48	0.42	0.29	0.22	0.20	0.20
Trace2	0.81	0.67	0.58	0.50	0.45	0.31	0.24	0.22	0.21
Trace3	0.71	0.55	0.44	0.36	0.31	0.19	0.15	0.14	0.13
Trace4	0.73	0.61	0.53	0.48	0.44	0.35	0.31	0.29	0.28
Trace5	0.76	0.62	0.52	0.45	0.39	0.23	0.15	0.14	0.13
Trace6	0.75	0.58	0.48	0.41	0.36	0.26	0.21	0.20	0.19
Trace7	0.78	0.64	0.56	0.50	0.46	0.35	0.31	0.30	0.29
Trace8	0.77	0.56	0.44	0.37	0.34	0.26	0.22	0.21	0.20
Trace9	0.85	0.73	0.63	0.55	0.49	0.31	0.17	0.13	0.12
Trace10	0.67	0.42	0.31	0.26	0.23	0.17	0.15	0.14	0.14

Table B.6: Cache Miss Ratios for the SDF Policy

n	1	2	3	4	5	10	20	40	75
Trace1	0.80	0.66	0.51	0.47	0.40	0.25	0.13	0.07	0.03
Trace2	0.81	0.66	0.54	0.49	0.42	0.28	0.16	0.09	0.04
Trace3	0.71	0.61	0.45	0.39	0.32	0.18	0.09	0.05	0.02
Trace4	0.73	0.67	0.54	0.51	0.44	0.31	0.18	0.11	0.05
Trace5	0.76	0.66	0.51	0.48	0.40	0.24	0.10	0.05	0.02
Trace6	0.75	0.61	0.47	0.43	0.35	0.23	0.13	0.07	0.03
Trace7	0.78	0.67	0.55	0.52	0.46	0.32	0.18	0.09	0.04
Trace8	0.77	0.60	0.44	0.39	0.32	0.22	0.14	0.08	0.03
Trace9	0.85	0.72	0.59	0.54	0.45	0.29	0.16	0.06	0.03
Trace10	0.67	0.44	0.30	0.24	0.20	0.13	0.08	0.04	0.02

Table B.7: Mean Address Resolution Depth for the DF Policy

n	2	3	4	5	10	20	40	75
Trace1	1.85	2.60	3.26	3.83	5.85	7.74	9.33	10.33
Trace2	1.85	2.60	3.26	3.85	6.00	8.17	10.16	11.17
Trace3	1.89	2.61	3.19	3.67	5.26	6.66	7.81	8.45
Trace4	1.93	2.73	3.43	4.04	6.18	8.36	10.41	11.49
Trace5	1.90	2.73	3.48	4.14	6.38	8.09	9.28	9.80
Trace6	1.86	2.58	3.19	3.72	5.51	7.22	8.61	9.37
Trace7	1.89	2.68	3.38	4.02	6.32	8.43	10.13	11.07
Trace8	1.82	2.52	3.09	3.55	5.05	6.69	8.18	8.93
Trace9	1.86	2.65	3.37	4.02	6.47	9.29	11.10	12.02
Trace10	1.76	2.28	2.64	2.92	3.80	4.69	5.52	6.05

Table B.8: Mean Address Resolution Depth for the SF Policy

n	2	3	4	5	10	15	20	40	75
Trace1	1.80	2.46	3.01	3.48	5.21	6.50	7.62	11.79	18.75
Trace2	1.81	2.48	3.04	3.53	5.36	6.78	8.02	12.67	20.31
Trace3	1.71	2.27	2.72	3.08	4.34	5.23	6.00	8.91	13.77
Trace4	1.73	2.35	2.88	3.36	5.38	7.06	8.65	14.67	24.66
Trace5	1.76	2.38	2.90	3.36	4.97	6.02	6.86	9.76	14.43
Trace6	1.75	2.33	2.81	3.22	4.77	6.00	7.11	11.27	18.22
Trace7	1.78	2.41	2.97	3.47	5.52	7.23	8.83	14.97	25.40
Trace8	1.77	2.34	2.78	3.16	4.63	5.86	7.03	11.40	18.73
Trace9	1.85	2.59	3.21	3.75	5.66	6.91	7.92	10.76	14.99
Trace10	1.67	2.09	2.39	2.63	3.51	4.29	5.07	8.05	13.08

Table B.9: Mean Address Resolution Depth for the SDF Policy

n	2	3	4	5	10	15	20	40	75
Trace1	1.80	2.46	2.98	3.46	5.14	6.25	7.05	9.07	11.00
Trace2	1.81	2.48	3.02	3.51	5.34	6.61	7.56	10.03	12.50
Trace3	1.71	2.32	2.78	3.17	4.47	5.27	5.85	7.32	8.74
Trace4	1.73	2.40	2.94	3.45	5.42	6.82	7.88	10.74	13.59
Trace5	1.76	2.43	2.94	3.43	5.11	6.13	6.81	8.34	9.74
Trace6	1.75	2.36	2.84	3.27	4.78	5.82	6.61	8.66	10.52
Trace7	1.78	2.45	3.00	3.53	5.56	7.00	8.08	10.73	13.13
Trace8	1.77	2.38	2.83	3.22	4.61	5.60	6.38	8.55	10.57
Trace9	1.85	2.58	3.17	3.71	5.62	6.90	7.86	9.89	11.48
Trace10	1.67	2.11	2.41	2.66	3.53	4.16	4.64	5.90	7.11

Table B.10: Cost Ratios for the DF Policy ($r = 1$)

n	1	2	3	4	5	10	20	40	75
Trace1	0.94	0.94	0.90	0.86	0.83	0.70	0.63	0.66	0.71
Trace2	0.95	0.94	0.91	0.87	0.84	0.73	0.66	0.67	0.71
Trace3	0.99	0.91	0.84	0.78	0.73	0.64	0.62	0.66	0.72
Trace4	1.03	0.99	0.94	0.89	0.85	0.72	0.67	0.67	0.71
Trace5	1.01	1.02	1.01	0.96	0.91	0.71	0.62	0.66	0.72
Trace6	0.96	0.91	0.86	0.81	0.77	0.66	0.62	0.64	0.70
Trace7	0.99	0.99	0.96	0.93	0.90	0.75	0.66	0.68	0.73
Trace8	0.92	0.89	0.82	0.75	0.71	0.64	0.64	0.67	0.72
Trace9	0.96	0.98	0.96	0.93	0.90	0.80	0.68	0.66	0.71
Trace10	0.86	0.71	0.62	0.57	0.55	0.55	0.58	0.65	0.72

Table B.11: Cost Ratios for the SF Policy ($r = 1$)

n	1	2	3	4	5	10	20	40	75
Trace1	0.90	0.85	0.80	0.77	0.75	0.71	0.74	0.82	0.90
Trace2	0.90	0.86	0.83	0.80	0.78	0.73	0.76	0.83	0.91
Trace3	0.81	0.75	0.70	0.66	0.64	0.62	0.68	0.77	0.85
Trace4	0.83	0.80	0.78	0.77	0.76	0.77	0.82	0.90	0.98
Trace5	0.86	0.82	0.78	0.75	0.73	0.66	0.68	0.77	0.85
Trace6	0.84	0.78	0.73	0.70	0.68	0.67	0.72	0.81	0.89
Trace7	0.88	0.84	0.82	0.80	0.80	0.78	0.84	0.93	1.02
Trace8	0.87	0.76	0.70	0.67	0.67	0.69	0.75	0.84	0.92
Trace9	0.95	0.92	0.88	0.84	0.81	0.73	0.69	0.74	0.82
Trace10	0.77	0.62	0.57	0.56	0.56	0.60	0.68	0.65	0.77

Table B.12: Cost Ratios for the SDF Policy ($r = 1$)

n	1	2	3	4	5	10	20	40	75
Trace1	0.90	0.85	0.77	0.76	0.72	0.67	0.65	0.69	0.74
Trace2	0.90	0.86	0.79	0.78	0.74	0.70	0.67	0.71	0.74
Trace3	0.81	0.80	0.71	0.69	0.65	0.61	0.63	0.68	0.75
Trace4	0.83	0.86	0.79	0.80	0.77	0.73	0.70	0.72	0.75
Trace5	0.86	0.86	0.77	0.73	0.73	0.67	0.64	0.68	0.74
Trace6	0.84	0.80	0.72	0.67	0.67	0.65	0.64	0.68	0.73
Trace7	0.88	0.87	0.81	0.82	0.79	0.75	0.71	0.73	0.77
Trace8	0.87	0.80	0.70	0.69	0.65	0.64	0.66	0.71	0.75
Trace9	0.95	0.91	0.84	0.83	0.77	0.71	0.67	0.67	0.73
Trace10	0.77	0.63	0.55	0.54	0.53	0.56	0.61	0.67	0.74

Table B.13: Cost Ratios for the SF Policy ($r = 3$)

n	1	2	3	4	5	10	20	40	75
Trace1	0.88	0.81	0.74	0.67	0.61	0.42	0.29	0.25	0.24
Trace2	0.88	0.81	0.74	0.68	0.63	0.45	0.31	0.26	0.24
Trace3	0.93	0.78	0.66	0.58	0.51	0.35	0.26	0.24	0.24
Trace4	0.96	0.86	0.77	0.70	0.64	0.44	0.33	0.26	0.24
Trace5	0.94	0.89	0.83	0.76	0.69	0.42	0.27	0.24	0.24
Trace6	0.89	0.78	0.69	0.61	0.55	0.38	0.28	0.24	0.24
Trace7	0.92	0.85	0.79	0.73	0.68	0.46	0.31	0.26	0.25
Trace8	0.86	0.76	0.65	0.55	0.49	0.36	0.29	0.25	0.24
Trace9	0.89	0.85	0.80	0.74	0.68	0.52	0.33	0.25	0.24
Trace10	0.80	0.58	0.44	0.37	0.33	0.26	0.23	0.23	0.24