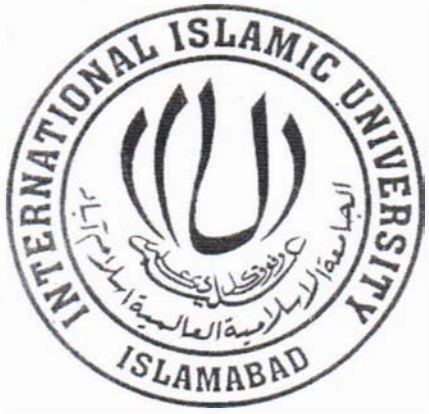
Hierarchal Hash Based Secure Network File System for Wireless Network



Submitted by

Waqas Nasar

563-FBAS/MSCS/F09

Supervised by

Dr. Muhammad Sher

Department of Computer Science and Software Engineering

Faculty of Basic and Applied Sciences

International Islamic University Islamabad

(2011)

International Islamic University
CENTRAL LIBRARY

TH- 8367
Accession No.

MA IMSC
005.4476
WAH

1. Network operating systems

2. Operatings systems (computers)

DATA ENTERED
08/07/13

Department of Computer Sciences and Software Engineering
International Islamic University Islamabad

Date 29-12-2011

Final Approval

It is certified that we have examined the thesis report submitted by Waqas Nasar, Registration number 563-FBAS/MSCS/F09, and it is our judgment that this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the Masters of Science in Computer Science.

Committee:

External Examiner

Dr Raihan ur Rasool

Assistant Professor
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology,
Islamabad.

Internal Examiner

Mr Muhammed Nadeem

Assistant Professor
Department of Computer Science and Software Engineering,
Faculty of Basic and Applied Sciences,
International Islamic University,
Islamabad.

Supervisor

Prof Dr Muhammad Sher
Chairman,
Department of Computer Science and Software Engineering,
Faculty of Basic and Applied Sciences,
International Islamic University,
Islamabad.

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

Declaration

I hereby declare that this thesis, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Waqas Nasar

563/FBAS/MSCS/F09

Acknowledgements

This project and all my efforts are fruitful only due to Allah Almighty, the Most Merciful and Beneficent Who gave me strength to complete this task to the best of abilities and knowledge.

I would like to take this opportunity of thanking my supervisor **Dr. Muhammad Sher** and who gave all his knowledge, guidance and support to boost my confidence and learning. He was always there to solve problems and helped me through this project. I am thankful to him for this.

Last but not the least; I would like to acknowledge Mr. Husnain Abass Naqvi, Mr Shehzad Ashraf Ch, Mr Syed Muhammed Saqlain, Zahid Mehmood, Iftikhar Ali Khan, Anwar Ghani, Naeem, Mubeen, Qasim, Yasir, Zahid, Waqas and all the faculty of computer science, All of them encouraged and provided logistic and technical help to me.

I would like to admit that I owe all my achievements to my truly, sincere and most loving parents, sisters, brothers, Nephew Ali, Niece, cousins and friends who mean the most to me, and whose prayers have always been a source of determination for me.

Waqas Nasar

Every researcher needs to address "What", "Why" and "How" for every research problem.

Abstract

The consideration of high bandwidth consumption over slow wireless network is an important feature for performance and efficiency. The issue got less attention as far as user perspective is considered. It became more important in interactive session on slow networks, where performance is key issue. Over slower networks interactive programs freeze, like during file I/O, batch commands the execution time increases manifold. In case of File I/O either user has to keep local copies of each or they can use remote login and edit the same file where it is placed. Remote login becomes frustrating for long latency networks. Wireless networks require more security and having low bandwidth than traditional wired networks

All wireless networks are resource constrained, the battery power, memory and bandwidth are main constraints along with security of information in wireless networks. Running of ordinary file systems on such wireless networks can result in short battery life and bandwidth dilemma for large file sharing, which results in high latency and unbearable delays which user do not consider while working on wireless networks, also the security is second main issue in wireless networks as any false node can inject his packet or private information can be divulged to some one who is not intended, so there is paramount need of an efficient and secure file system for wireless networks.

The thesis describes the need of secure and bandwidth efficient file system for wireless networks and also proposes a file system which is secure and bandwidth efficient and can become a part and parcel of wireless networks.

Simulation results are evident of the performance achieved for our proposed scheme. Proposed methodology can save maximum bandwidth by taking advantage of cross file commonalities, in best case analysis it will just transmit a packet containing just hash tree.

Table of Contents

Chapter 1

Introduction

1.0. Introduction

The responsibility of a file system is to store, manage, copy, move, modify the files, which is a basic unit in some file system, this unit contains metadata and contents or information stored in it, all above mentioned operation can be performed on both file metadata and file contents. The network file system is responsible for all above mentioned tasks of a file on a network as well as performing three more tasks related to network which are remote procedure call, External data representation and operation of layer seven of OSI reference model.

Remote procedure call is performed at session layer its purpose is to provide access to different procedures residing on remote machine, The representation of data of external devices is to uniform distribution of storage capacities of devices present on a network and to access all these storage devices smoothly by all the nodes of a network. There are two part and parcels of Network File system which are Location transparency and Location independence, the purpose of location independence is to name the files such that the name does not expose the storage location of the file and location independence means if the physical location of a file is changed then its name should not be changed. The purpose of these two characteristics of network file system makes the file user feel comfortable while dealing with files that is he do not worry where the file is going to store or if the location of file is changed then how he will access the file, the file user will perform all the operations as the file is residing on his own machine.

The main problem being faced by user / designers of a network file system is the cache consistency problem which to keep the cache copies consistent with the master file.

Recent advances in wireless networks led to many promising applications, as these applications are increasing rapidly so the secured communication also becomes the most important issue in wireless networks along with low bandwidth. The consideration of high bandwidth consumption over slow wireless network is an important feature for performance and efficiency. The issue got less attention as far as user perspective is considered more important in interactive session on slow networks, where performance is key issue. Over slower networks interactive programs freeze, like during file I/O, batch commands the execution time increases manifold. In case of File I/O either user has to keep local copies of each or they can use remote login and edit the same file where it is placed. Remote login becomes frustrating for long latency networks.

Wireless networks require more security and having low bandwidth than traditional wired networks, so a file system is required which ensures reliable and low bandwidth network operations.

1.1. File System

A file system (file system) means that to arrange data likely to be retained after a program terminates as long as actions to store, get back and renew data, as well as handle the available gap on the device(s) which hold it. A file system organizes records in an capable way and is tuned to the particular uniqueness of the device. There is usually a tense mixture between the operating system and the file system. Some file systems present mechanisms to control entrance to the data and metadata. Ensuring consistency is a major task of a file system. Some file systems present a means for multiple programs to renew data in the same file at almost the same time.

1.2. Network File System

NFS allows a system to share files with others over a network. In Network File System users and programs access files on remote systems as local files.NFS is a network based system for remote file access. Retrieve (work with) files stored on remote servers. The NFS protocol is a collection of remote procedures that allow the client to transparently access files stored on a server.

The main characteristics of Network File System are

1. Local workstations use comparatively less space because commonly used data can be stored on a single machine and other devices can access over the network.
2. Users do not have need to have separate home directories on every network machine. Directories could be set up on the NFS server and set available throughout the network.

1.3. Low bandwidth File System

LBFS is a network file system which conserves message bandwidth between customers and servers. It takes benefit of cross-file similarities. When transferring a file between the customer and server, LBFS identifies chunks of data that the receiver already has in other files and avoids transmitting the unnecessary data over the network.

1.4. AFS (Andrew File System)

The Andrew File System (AFS) is a circulated networked file system which uses a set of trusted servers to present however, location-transparent file name space to all the customer workstations. It was developed by Carnegie Mellon University as element of the Andrew Project. Its major use is in distributed computing.

The main features of Andrew File systems are as follow.

The Important feature of Andrew File System is Volume, a hierarchy of records, sub-directories and AFS mount points (associations with other file systems).

AFS can be presented only to read-only cloned copies. When accessing records in a read-only volume, a client system will recover data from exacting read only copy. If at various points that copy becomes engaged, customers will look for any of the remaining copies. Again users of that data are unconscious of the place of read-only copy; administrators can make and transfer such copies as required. The AFS commands set guarantees that all read-only volumes have correct copies of the original read-write volume at the time the read-only copy was shaped.

1.5 CODA File System

Coda is a circulated file system developed as a research project at Carnegie Mellon University since 1987 under the route of Mahadev Satyanarayanan. It descended openly from an adult edition of AFS (AFS-2) and offers many related features. The Inter Mezzo file system was encouraged by Coda. Coda is still under development, still the focus has shifted from examine to creating a strong result for business use.

Coda has many features that are enviable for network file systems, and some features not found elsewhere.

1. disjointed function for mobile computing
2. Is freely available under a moderate authorization
3. High performance through customer side persistent caching
4. Server copying
5. Security model for verification, encryption and access control
6. persistent process during limited network failures in server network
7. Network bandwidth variation
8. Good scalability
9. Well distinct semantics of division, even in the existence of network failures

1.6 NFS Architecture and Main Components

The procedure of NFS is defined in the structure of three main components that can be viewed as sensibly residing at each of the three OSI model layers related to the TCP/IP application layer. These components are:

1.6.1 Remote Procedure Call (RPC): RPC is a standard session layer service used to apply client/server internetworking functionality.

1.6.2 External Data Representation (XDR): XDR is an expressive language that allows data types to be defined in a reliable manner. XDR theoretically exist in the presentational layer.

1.6.3 NFS Procedures and Operations: The real functionality of NFS is implemented in the form of events and operations that theoretically function at layer seven of the OSI model.

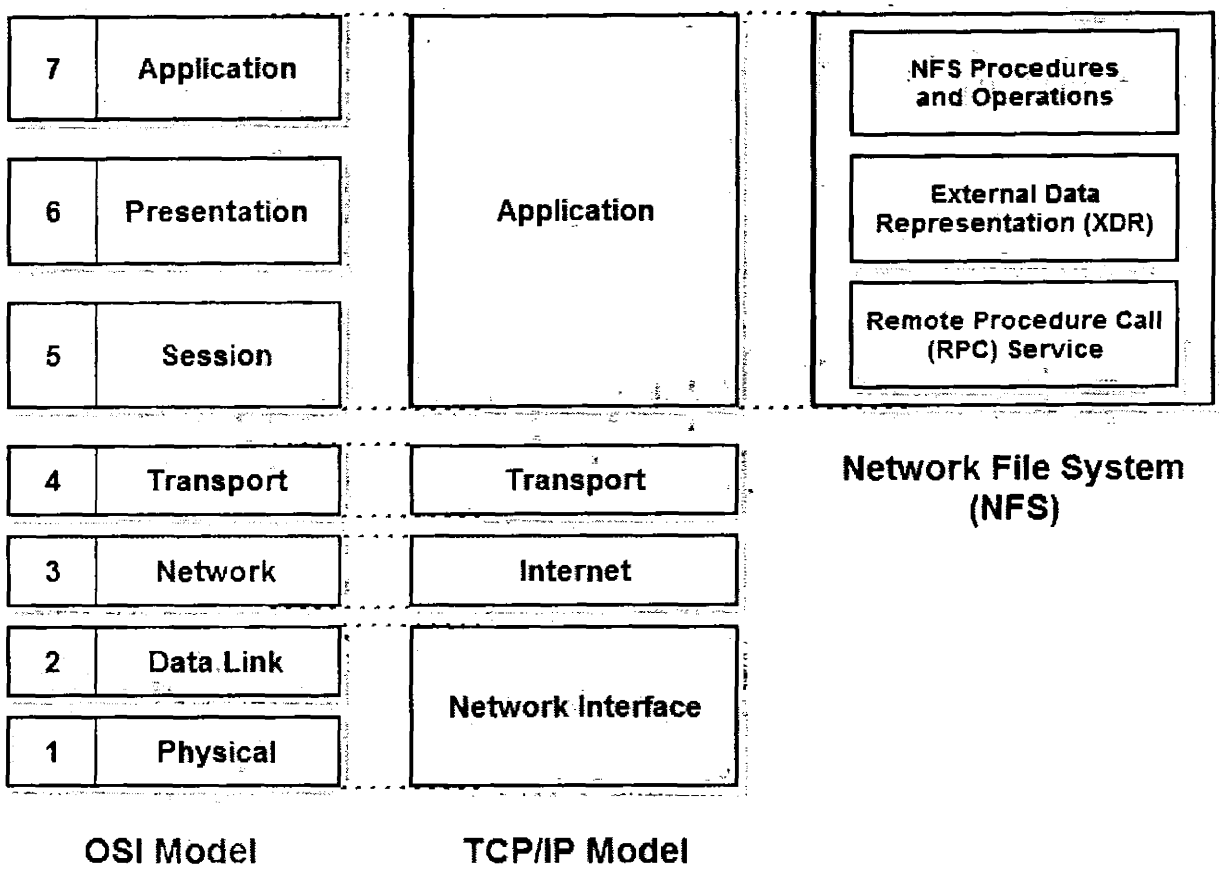


Figure 1.1 NFS Architectural Components

1.7 Research Methodology

A detailed study and literature review has been done before writing about the subject in detail. The topic requires high level of data explore in latest journals, articles, white papers and

technical writings. To understand the topic I explored many books and electronic resources (digital library, e-journals and websites). In order to support the topic tests are conducted in Java which are discussed in next chapters.

1.8 Thesis Organization:

The thesis contains these chapters. Chapter order and description are mentioned below.

Chapter 1 is detail introduction of the file system and different file system like CODA, Andrew File System, Network File System and Low Bandwidth File System and their Characteristics has been discussed in this chapter.

Chapter 2 provides a detailed study (Articles, Research Papers) about the subject technology and also all those articles on which thesis is based.

Chapter 3 Problem Definition provides the problem briefly.

Chapter 4 Proposed Solutions Design provides the solution for the given problem and also the diagrammatically design of the solution.

Chapter 5 Conclusion and Future works provide the results of the proposed solution and comparisons of the solution with the existing system and detail results in tables along with the future works.

Chapter 2

Literature Review

2.0. Literature Review Process

*"Indeed, one of my major complaints about the research field is that whereas Newton could say , **If I have seen a little farther than others, it is because I have stood on the shoulders of giants,** I am forced to say, Today we stand on each other's feet. Perhaps the central problem we face in all of research field is how we are to get to the situation where we build on top of the work of others rather than redoing so much of it in a trivially different way. Science is supposed to be cumulative, not almost endless duplication of the same kind of things".*

Richard Hamming 1968 Turning Award Lecture

2.1 File Systems

2.1.1File Systems for mobile computers

Today's mobile computers are famous due to their portability, functionality and availability. Due to their mobile nature these machines usually have problem in synchronizing files with between them as well as with other servers since this synchronization is mostly carried out manually by the users. This is mainly caused by the poor network connection or even the absence of the connectivity for long time. Because of the manual nature of the synchronization process, it is not only time consuming but also prone to errors.[22]

There are a number of proposed solutions which can be classified under two categories.

2.1.1.1 Disconnected Systems

Disconnected systems need no network connection to the central resource for their function. Disconnected systems are actually automated versions of the manual user process. They automate the process of copying the file between the user's system and the central resource. These systems can be robust but limited in nature as they will need some information at some point for completing their function from the central resource. Therefore the main focus of these systems is how to maximize the availability.

Disconnected systems are further divided into "Coda", "AFS", and "PersonalRAID".

Coda uses replication to localize data and provide fast access to the data on the local machine (node). This machine will serve request in disconnected mode and will report in case of cache misses. For cache consistency the technique of callback is used where each node if modifies a file, will coordinate it to all other nodes through the server in order to keep them up to date. In this way there is no chance that a node will use stale data as the server will coordinate all the updates with all nodes. This technique has issues when a node loses its connection to the server in which case it will miss the callback breaks and will keep using the old outdated file even after returning to the network.

To avoid drawbacks of the in the above mentioned technique Coda uses a relaxed replica control model for consistency to ensure optimistic performance of the disconnected operations. In this model the Coda client record a log of operations performed on the cached data. When reconnected the log file is replayed on the server to source file with the replica. To avoid write-write and other conflicts the server uses versioning technique.

AFS is a distributed file system similar and contemporary to the previously mentioned technique Coda. Like Coda. The development of AFS was influenced by the advantage presented by Coda for disconnected operations.

AFS caches data from the central server to provide local access. A similar callback technique like Coda is used for cache consistency. As mentioned due to the advantages of Coda the demand for similar feature for disconnected operations in AFS grew rapidly. As a result in [20], a solution was presented for providing support for disconnected operations known as "disconnected AFS" in order to cope with the growing demand. The only difference is that disconnected AFS was an addition to an existing system where as disconnected is not.

Apart from the above comparison a closer look to both techniques will show the none of these techniques have functional advantage over the other. Instead the working principle of both the system is same and both of them offer same consistency and usability. The plus point for AFS is that though it is a hack to an existing system with no need of any change to the base system while Coda presents a cleaner design.

PersonalRAID strives to solve problem related with two nodes using same file but not at the same time. The typical example would be a person using both home and office PCs. Because of the slow nature of the wide area network the feasibility of running the traditional network file system does not seem justified. The manual process of synchronization through portable storage is not only time consuming but also prone to errors. Since the portable storage technology is cheap in today's computer world. Therefore PersonalRAID [8] uses this technology such as flash or microdrives also known as VA, to solve the synchronization problem. To synchronize both the nodes, updates to the file need to be moved between the two nodes. VA works as an acting stage for the PersonalRAID for the moving the update between the nodes. The node's function is subject to the attachment of VA to that specific node means the system (node) will function when the VA is attached to it. If any one of the component including the VA is lost the data loss to the system can be recovered. Recovery due to the loss of any of the two nodes is relatively simple however the case due to the loss of VA is complex. The synchronization due to the loss of VA needs two passes of the replacement VA to each node.

Due to the limited storage of VA the user needs to visit each system regularly in order to propagate the modified blocks to all nodes. This system supports only one user. For multi-user system the VA has to be presented at each node in order to maintain consistency which is not feasible.

2.1.1.2. Weak-Connectivity Systems

Weak-Connectivity Systems always have or assume to have a link available to them for communicating with the central resource for their proper function. But the problem with this approach is that the link is of lower quality than the one available to traditional computers running normal file system. Therefore it will produce long bouts of delays and will starve any other processes try to use the network resources. These types of systems strive for conserving the network bandwidth. They are categorized as "content-based caching", "weakly-connected Coda", "weakly-connected AFS", and "Segank".

The content-based caching is best suited for bandwidth-poor environment; it will be better to perform some extra computation on the data before sending it out on the network and identify

similarities in patterns. In this sending of duplicated data can be avoided and as a result the burden on the network will be decreased. This is achieved through application of hashing and indexing techniques on the data.

For weakly-connected Coda some modification to the original design were make it feasible for the weakly connected environment. The changes include modification to the consistency mechanism and update propagation methods. Similarly weakly-connected AFS is also a result of modifications to the original version for disconnected systems. The provide users with the most recent version of a file callback breaks are processed which are received from the server. When a cache miss occur it sends request to the server in order serve the user requests. In this version the same logging feature is still working which is used to record operations and replayed at the server at a later time.

Segank [9] is designed to allow a user to have multiple devices (desktop PCs, laptops, PDAs, media boxes, etc...) with connection to different quality network links. These devices can share data in one namespace transparently. This system supports single user just like a personalRAID system. The portable unit in Segank is known as MOAD. The responsibility of a MOAD unit is to carry mapping information about the data's location. Second responsibility is to provide communication for nearby devices. [22]

2.1.2 A Shared-Disk File System

The locking and recovery technique were extended to scale large clusters. This system is fully based on the distributed locking and also recovery technology.as far as the limitations of the distributed locking , the authers also are very pleased to see that distributed locking scales are quite well and bette.the authers also suggest the number of techniques that make the dsitributed locking quite well.The examples are the byte ranging taken optimiztaion dynomic optimization for meta nodes that is nused for managing the file.[23]

existing GPFS file system is also used for the large scale to the supercomouter in the world also provide the falut taolrent faethers that are used for the large systems.The newly systems is also much better with raespect to the cost feature about the large computers also.

At the start of computing there are still problems for the largest systems/ machines and still these problems also continues today.in order to overcome of these problesm(larger size machines) the Computing clusters is new termonolgy that is used.the computing clusters are used widly dure to unlimited processing power and storing capacity as compared to the larger computers.the cluster are also fault tolerence also.

Drawback of cluster is that the is used with seprate machines.Our system GPFS are also absed on the cluster computing.GPFS is also used in the 6 or 10 largest computers in the world.tardational supercomputers are also run on the clusters.But GFS also sucessfully overcome these problesms by using high sacalebility.

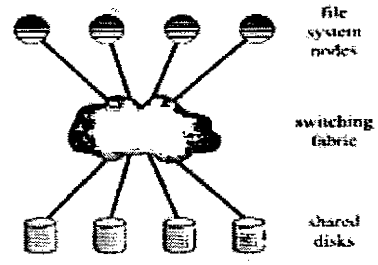


Figure 2.1 Shared Disk Environment [23]

GPFS fully support to data and meta data.all other nodes has connected to all disk in the file systemfurthermore the load alancing is also achived by the disk sub system.the swithing fabric shows in the digram connect the file sysyetm to disk , that consist of SAN.further more GPFS also uses the distributed locking to to syncronized the shared disk.

2.1.2.1 Data Striping

In order to get the data striping , GPFS also implements the striping in the file system.In GPFS the data is manged by the equal blocks , and different blocks in a round roban fusion fusion.Large blocsk also give same advantages.

GPFS Put the data in buffer pool and issue a request to parallel to as many disk in order to achive the bandwidth in which the switiching fabric is capable.

For handling of millions of files GPFS uses the extensive hashing , in order to organize the directorty within the directory. The directory grow hashing add new directory block.

For the recovery machisum , in the large file system is not poble but in gpfs record all metadata record that effects the file system consistancy.all of the nodes recorded in the log and af failure can be detected by using the log file

2.1.2.2 How to manage the paralish and consisitancy?

The GPFS grandtee the single node equillent the posix semantics for the systems operactions across the cluster.

Inorder to achive the true syncronization there are two methods are used.

Distributed Locking: every file operaction require apporate read write operactions.

Centralized opercations: all opercations are performed to one node.GPS is futhermore based on the distributed locking.in order to permed these opercations the GPFS implemented a lock nmanager.The lock manger uses the Lock Tokens.These are used to send or release of messages between the nodes.

Furthermore GPFS usese the byte range locking to scnronized the read and writes the file data.inorder to achive the byte range ,the overall opercations is as follws.first nod is used to write a file will require byte token for whole file system.when second file is used that revoke the byte range of the first node.

In case of Meta data , one node is used for the meta node for the file.only the metanode read or writes the from the innode from the disk.

Allocation maps: in GPFS the allocation maps are used for the free/used space in the disk.in oreder to achive the dsik space we can siamltanisoly scncronized between the nodes.furthermore an allocation manager is used which can constantly moniter the disk space statics.in case any

node is deleted , there is necessray to upadte the allocation space in the current table of allocation manger.

GPFS also uses the Global metadata including the file sysytem configration data , space usage and control acccess attributes.for all system file formats.

GPFs Testing:-Gpfs can be experimented by the several hundrad customer sites on cluster ranging from the few sites which less than the 1 tear byte of the site , and it has sucessfully achived all goals.

GPFS is based on the cluster computing , the advantage of cluster computing is that computing can perfomed more faster, similaraly the drwaback of the cluster computing is that programms must be partationed to run on multi machines.and it is very difficult to mange the all resaourses held by cluters.

Similaraly in case of disk space and relibility feature , the GPFS has sucessfully achive all results.it has support upto 4096 disks upto 1 TB in each size fot total of 4 petabytes per file system.further more for more effeiecy large files are dived into equal disk blocks for processing.As directory grows hashing adds new directory blocks at one time.

with respect to data/metadate it has sucessyly achived all results in the experiment.GPFS is also good in Scalibility.it also allow the adding , deleting , replacing the disk in existing system.it has also global metadata including system configration data space ,access control list etc.

In more seniors of Logging and recovery machium it is better than the Perivios all systems.

Further more in case of locking the GPFS Has performed excillent performance , it is based on distributed locking machisuim , different nodes performed the processing on the differnet nodes at one time.so we said that it can be achived better than the centralized apporch.another advantage is that load balancing.the drawback of this apporch is that more cost effective with rsepect to the forwarding the request to the central node.

In case of reponseness , GPFS used the number of optimizations in token protocol that significantly reduce the cost of the token management and responsivness as well.

In case of fault tolernece at node faiulre, GPFS achives the successful result.

Similarly the communication failure , not good beacue GPFS membership protocol will not give the gaurntee that how long it take the each node to recive the process faiulre notification.

In experiment shows that it is significant impact on highly parallel applications.but large GPFS has experienced a micro faiulre problem during changing of disk.

GPFS was built in many ideas during the years of reserch.the idea is that to build better system for supercomputers with extra feature in existing GPFS.GPFS is based on cluster computing . for faster processing .GPFS is based on distributed locked and recovery technology.the idea is that to achive saclibilty and other relevent features.further more during expermenting authers achive some results that are expecting but draw back also.Several significant cahnges are sugest by the authes in the exsiting GPFS.There are also some drawbacks , cluster computing are also expensive.with reaspect to fault tolrence it has achived some raesults.

Jaudgemnt of article:-while studying the current paper I closly observe some points.I have already read the advantages and disadvantages of the current GPFS.Obiousvly the advantages are much more than the disadvantages.I said that this is a good apporch but still some progess has been made.the authers sucessuly achived the basic objective.much more progress is needed with respect to Use of cluster Computing.The use of cluster computing is too costly.similaraly this cost factore can be analysed in distributed locking also.

With respect to the communication protol much effeort is needed due to node failure and recovery machanisum.[23]

2.1.3 The ITC Distributed File System

The main purpose is the transparency, user mobility and compatibility with existing operating system interface. In security point of view security of the design is important and the key element is cashing for entire file of the workstation. The design system is to support different kind of workstation and heterogeneity. The consideration of distributed file system over sharing is to high transparency, security, heterogeneity, user mobility, performance, availability, scalability and integrity. The structure of the system is vice and virtue, the shared files stored on vice and the responsibility of virtue is to access the files from vice and two different

programming design for vice and virtue. Release and modification of information are considered as security issues. Cache is basic implementation management in vice and the total capability of cache is less the vice. The server design is base on cluster server where on each server one UNIX process that deals with each user that communicate with server. [24]

This paper presents the design and basics of a distributed file system for a network consist on more than 5000 personal computer workstations which present basis and background for this decision.

Objective of the paper is to develop a mechanism to support sharing of information between workstations and also develop a sharing mechanism that should not forbade ordinary and natural moment of users. The other main objectives of this paper are:

1. Design a mechanism to combine the personal computer and time sharing model.
2. To support user mobility via location transparent.
3. Design to support a heterogeneous design in existing operating system.
4. Clustering to develop the locality of usage and replication of read only file system
5. The access to the user is flexible and independent from workstation and network security.

2.1.3.1 System Overview

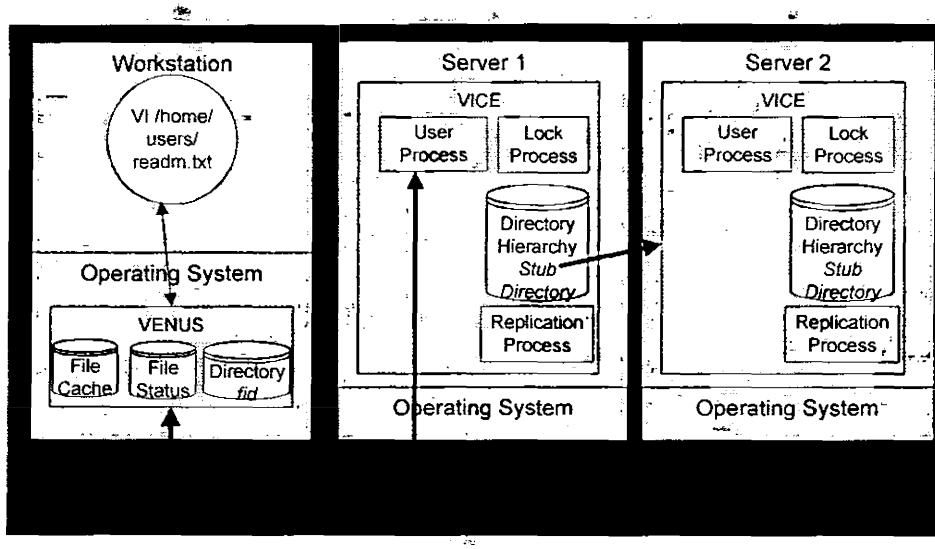


Figure 2.2 Distributed File System Model [24]

2.1.3.2 Limitations in Model

1. Too much stat call degraded performance
2. This was solved by reducing the cache lookup
3. Sever side overload due to too many processes
4. Network resources in the kernel frequently exhausted
5. Since location information was stored in each server, moving files across servers became difficult.
6. Was not possible to implement Disk Quotas

2.1.3.3 Drawbacks of ITC Model

There are three main drawbacks in presented model

There is no modeled file creation and deletion, the sharing controls are not precise, and it is not validated in this system. File creation and deletion change the state of the file system permanently.

The second drawback is with modeling file sharing claims that there are two kinds of file sharing:

- concurrent and
- Serial.

This System can control the overall degree of file sharing but cannot specify the exact proportions of concurrent and serial sharing. It believes that this is a problem and the solution is to monitor the degree of each kind of sharing, so that it at least be able to match the degree of each kind of sharing with performance data. Finally, since the workload model is aimed at distributed systems of the future, it not has been able to come up with traces against which this system can be validated. This is certainly a serious drawback.

2.1.3.4 Benefits of ITC Distributed File System

1. AFS: Local File, NFS Remote File
2. Having a combined approach to achieve best world.
3. The first access to the file will be remote access.
4. The file will then download on a low priority.

5. Partial download of the file, the server need not know how much file is downloaded by the client.
6. Subsequent operation can work on the local file.
7. Transfer only Changes back to the server.[24]

2.1.4 XML-based Grid File Storage System

Grid Security Infrastructure (GSI) provides new capabilities to be used on top of different Grid middle wares. The solution is specifically implemented on gLite and accomplishes the access to data storage Grid resources in a uniform and transparent way. The proposed approach is based on XML data encapsulation to store encrypted data and metadata as well as data owners, X509v3 certificates, AES keys, data size, and file format. [4]

This work improves the latter aspect through the use of asymmetric RSA (algorithm for public key cryptographic) and symmetric AES (Advanced Encryption Standard process, the process used in choosing an algorithm for standardization as AES) cryptographic algorithms applied to the Grid file storage system. It points out specifically the details on how to provide a data encryption solution for the Grid Storage Element (SE), since the default middleware implementations are based on traditional, not encrypted storage systems.

This paper is an architectural improvement of the gLite middleware (is the middleware stack for grid computing used by the CERN LHC experiments and a very large variety of scientific domains) concerning security mechanisms. The security system introduced involves many parts of the infrastructure, as well as resources access, secure data storing and secure data transmission. It gives an answer to the requests pointed out by business companies that wish to use the Grid for commercial applications.

The main advantage of our proposed solution refers to the independence from the environment on which it is deployed. [4]

2.1.5 View-Only File System

The challenges such as spyware, removable media, and mobile devices which make the sensitive contents disclosed. VOFS (View-Only File System) depends on trusted computing primitives and virtual machine technology to ensure high level of security in the current system. Before allowing the user to view the data VOFS clients disable non essential device output before allowing the user to view the data to non-secure VM (Virtual Machine). After finishing work, VOFS sets the machine to previous state and start normal device activity. VOFS does a difficult task of prevention of data linkage from authorize user to some extend using VMM (virtual machine monitor) technology. In the client architecture VMM acts as guest VM, and is referred to as primary guest VM. This machine has not sufficient access to the resources. Another guest virtual machine has complete administrative access to all the resources. A third guest say SVFS VM is responsible for downloading sensitive contents, storing it, and telling the domain 0, when to save or restore primary guest state and when to enable or disable device output. [6]

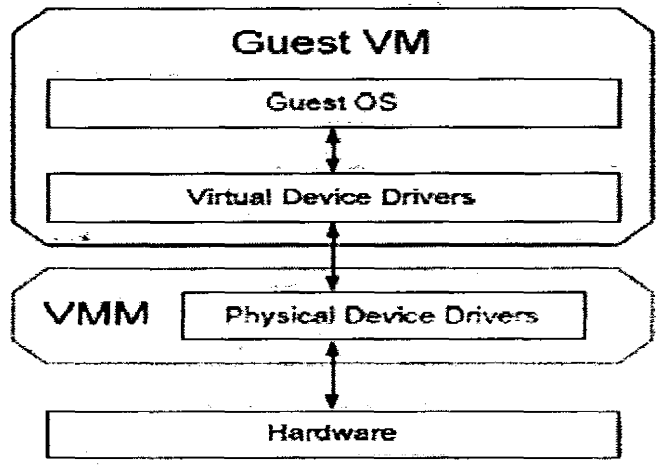


Figure 2.3 Virtual Machine Architecture [6]

The VOFS clients also have designation of trusted platform module (TMP) technology. The clients also take the benefits by using an integrity measurement architecture which uses trusted boot to enable remote verification of trusted components by content providers. If a user

wants to see only file in VOFS, then the primary VM requests to SVFS VM. It requests the content provider to start an authorization session.

2.1.5.1 Goals of View Only File System

To propose a design of a view only files system to ensure security of sensitive contents from unauthorized users. Developing a methodology which do not allow even authorize users to disclose the data to other unauthorized users. The system should reset the system after user has finished his work. The system should assure remote verification of software integrity. It proposed a system which provides high level of security.

2.1.5.2 Limitations in View Only File System

1. Instead VOFS provide high level of security, it still relay on the integrity of trusted components.
2. The VOFS file system still does not provide full protection against all type of attacks.
3. The attacker can install arbitrary software on the primary guest VM when it is running.
4. The attacker can replace operating system VOFS clients when it is turned off.
5. The attacker can access the hard drive or any other removable media in offline mode by removing them from the system.
6. VOFS is not recommended for end consumer digital rights management.
7. Implementing the proposed security mechanism in the VOFS consume high cost.

2.1.5.3 Improvements

1. Making VOFS recommended for end consumer digital rights management.
2. The VOFS has removed the overhead as compared to other file systems, but removing remaining overhead is possible by storing the files in the same system instead of separate system.
3. Giving full concentration to the security issues so as to provide full security from attackers.

4. Developing such a mechanism for security which is cheap.
5. Some if the issues remain to be implemented like client and server applications for content transfer that utilize TMP technology, disabling of output from non-network devices.
6. The authors plan to iteratively perform a thorough security evaluation.
7. The authors hope to identify and eliminate any secret communication channels or weaknesses in the original system that could be used to bypass security mechanisms and leak sensitive information.

2.1.6.1 Distributed File System

Conventional method used for authentication was" reference monitor". This method was truly centralized, now a day's world is switching to distributed environment for this requirement to be fulfilled centralized approach should not be change. In conventional reference monitor method had problems of scalability and resource protection. In traditional reference monitor method Access Control List (ACL) was maintained, to attain the resources every user must be entered in the ACL. ACL was not scalable, so there should be a way to modify the ACL approach to give access to the new user. Another drawback of ACL was the need of the full time monitoring the requests for resources. There is also a need to discuss about the issues of protection and scalability.

2.1.6.2 Cryptographic Access Control in a Distributed File System

This method eliminates the centralized approach and handles both issues resource protection and scalability. In proposed method encryption is applied at the client and the encrypted data is stored at the server without decrypting the file. The creator of any file is the generator of the keys for encryption and decryption and he is the only who can modify the file. The server saves the file along with keys and the server also maintains the binary tree at server side so fast access can be done in case of searching of the keys. The benefit of saving the key along the file is if any trouble with the binary tree happened then the key is not compromised. Due to the distributed in the nature the proposed method provides scalability, controlled access

to the resources, gets rid from constant monitoring because of the encryption the only user can decrypt the file who owned the decryption key (means only the trusted user). The author claimed that authentication, integrity are achieved through digital signature and hash function. [7]

2.1.7.3 Performance Comparison between NFS and CNFS is

Operation	NFS		CNFS		Overhead
	Mean	SD	Mean	SD	
8K Read	0.79 ms	0.09	0.63 ms	0.02	-20%
8K Write	0.89 ms	0.11	23.062 ms	1480.16	+2600%
1MB Read	792.84 ms	83.11	874.168 ms	150.85	+10%
1MB Write	1187.89 ms	156.30	1480.16 ms	222.16	+25%

Table 2.1 Comparison between NFS and CNFS [7]

1. For cryptography use only one algorithm that provide authentication, integrity, secrecy.
2. There should be a mechanism to ensure trusted clients
3. Multiple encryptions and decryptions cause less efficiency.
4. Need better caching mechanism to attain better performance.

2.1.8.1 Encrypted File System

Modern distributed computing system is difficult to limit reliability access to sensitive data. Network often unselectively broadcast data to far reaching and unpredictable places remote login facilities create new opportunities for trespassers and distributed file system. To reduce these risks cryptographic techniques make it possible to limit the data access while taking the advantages of untrustworthy network and services. Data encryption technique attempts ensure that only those who have the decryption key can take the original data. Commercial applications of encryption technique protect communication links. Endpoints are under the control of a single entity and trust a common authority. Keys is assigned and changed, the problem is to ensure that sender and receiver agree with current keys and key is discard when no longer use. Ensuring

assesses by third party when the keys are lost or unavailable is rarely an issue. Public key technique is easy to allow two parties to establish a secure link. [25]

2.1.8.2 Key Management in an Encrypted File System

Cryptography can be used to protect file data and file encryption take place at the application level with some tools such as UNIX crypt command. In a secure communication system keys are distributed and synchronized geographically and keys are used to for dual purpose authentication and reliability. In a file system there is a little need to distribute keys geographically and files are encrypted and decrypted at the same locations. Cryptographic technique can be used to detect the unauthorized tampering with the file data. If differently key management problem is said to be distributed temporarily. Keys have longer lifetime then the communication and if the key is lost or unavailable then file is rendered useless. The key distribution centre and public key cryptographic protocol is developed for geographically distributed communication system. Because of some difficulties with key management the sensitive files are rarely encrypted even when encryption tools are available. In business environment where ensuring the the availability of data to authorized users is very important. A toolkit (Crypt Breaker's Workbench) is available in internet for purpose of decrypting files enciphered with UNIX crypt program.

In the organizational information system cryptographic file protection have several problems not addressed by the traditional key management schemes.

The main features provided are

1. Ensuring protection of communication link and sensitive data from unauthorized users
2. The author used the new algorithm based on DES cipher for online encryption of file data in secure and efficient manner that is used in smartcard

2.2 Low Bandwidth File system

File system support for low-bandwidth channels typically the wireless and mobile ad-hoc networks where devices are mobile, typically the problem arises for wireless networks where browsing, downloading and exchanging files can halt the network traffic and a situation of deadlock occurs, the problem is increased twofold when mobile device users wants to share large files or want to collaborate each other [2]. Two approaches were adopted before the proposed solution

1) Fetch the file metadata only

2) Fetch both metadata and file contents

2.2.1 Limitations of File system for low-Bandwidth

Metadata just gives information about file size, type etc which in some cases is not appropriate (e.g. video files etc), Option 2 is impractical because of resource constraints.

In proposed remote file system the metadata of the file is modified and thumbnail information is added, when a user want to access a file remotely, first of all only metadata of the file will be sent to him, as metadata also contains thumbnail so after seeing thumbnail user can decide whether to open the file or not, so author claimed to reduce significantly bandwidth consumption.

2.3 A Low-bandwidth Network File System

The consideration of high bandwidth consumption over slow network is an important feature for performance and efficiency, the issue got less attention as for as user perspective is considered

It became more important in interactive session on slow networks, where performance is key issue, over slower networks interactive programs freeze, like during file I/O, batch commands the execution time increases many fold. In case of File I/O either users have to keep local copies of each or they can use remote login and edit the same file where it is placed. Remote login becomes frustrating for long latency networks.[1]

The paper proposed a new network file system LBFS (Low bandwidth network file system). the main idea behind LBFS is to exploit the similarities between the versions of a file placed on different systems over LAN. LBFS avoids the sending of data that is already present in client's cache. The method is close-to open consistency. After a file has been written and closed by a client the edited file will be available to all the subsequent users. Moreover, once a file is successfully written and closed, the data resides safely at the server. LBFS assumes that the entire working of a client will remain on client's cache and the communication is solely for maintaining consistency. When client modifies a file, it must be transmitted to server, similarly when a client opens a file the server sends him the latest version of file.

LBFS creates chunks of a file and keep the indexes of these chunks in a chunk data base which will reside on both server and client, the indexes are created using SHA-1, which is collision resistant, the chunk boundaries are created on the basis of file contents if any of client or server modifies the file, the modification only affects the surrounding chunks. Chunk boundaries called region are created using Robin fingerprints, Expected chunk size will be 2^13 = 8192 = 8 KBytes + 48-byte breakpoint window size.

2.3.1 Low-bandwidth Network File System Provides

1. Transmission of files over slow LAN.
2. Maintain file consistency while more than one user is modifying the same file over LAN.
3. Significant bandwidth reduction for common workloads.

SHA-1 provides collision resistant, so the probability of having same Hash code of two chunks having different contents is very low.

1. Robin Fingerprints is used for chunk boundaries
2. Files are not encrypted/compressed

2.3.2 Bandwidth utilization for LBFS and other file systems

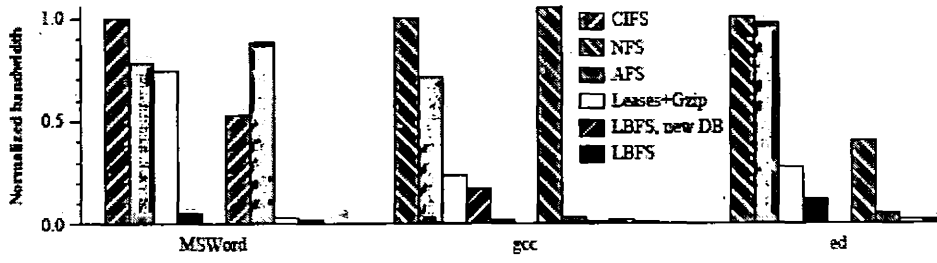


Figure 2.4 Bandwidth Utilization of LBFS and other File Systems [1]

- LBFS provides significant bandwidth reduction for common workloads.
- Efficient Bandwidth utilization in case of interactive session and file transmission can be achieved by exploiting commonalities between different versions.

2.3.2 Improvement in Low-bandwidth Network File System

1. Some other Hashing algorithms can also be used instead of SHA-1
2. Some other mechanism can also be developed for all applications to benefit from LBFS

File sent on network will be compressed and encrypted chunks will be calculated on the basis of original file contents and Hashing will be performed on the original file while data will be sent compressed and encrypted we will use L-Z compression and RC4 encryption, for chunk boundaries we will use overlapping data (48 K. byte) window size and will use some other mechanism instead of Robin Fingerprints,

Some MAC or HASH codes will be used for authentication and for confidentiality RSA can be used. [1]

2.4. How to Improve Bandwidth Efficiency

Peer to peer storage system attached the very low client system. This technique apply at the client side, that techniques reduce the bandwidth use to store and access the file. The peer in this technique cannot be trusted because data is unencrypted format; to make technique trusted author make use of HASH. Author has use old technique that is called OceanStore prototype. The early systems have to use high bandwidth but new research to reduce the bandwidth at client side.

Author target the users at home and mobile users that connected to the network by using very low bandwidth. In this paper author assumes infrastructure FARSITE or Ocean Store. This infrastructure provides serialization, which is responsible for updating and verifying recent version of file or data. In serialization he uses Byzantine agreement algorithm and threshold signatures. [26]

The limitation of this paper is reference chunks or the block of data is varying sizes, because use Rabin fingerprints technique. Other limitation is data privacy cannot ensure. Future work of this paper is measure the performance increase over a wider range of real-world workloads. [26]

2.5 Wide Area Network

Wanax is a flexible and scalable WAN accelerator targeting rising area Internet access is a scarce commodity in the developing regions and it is expensive and slow. Zambia example [Johnson et al. NSDR'10] presented that 300 people share 1Mbps satellite link and paid $1200 per month. Web proxy cashing is not enough. [2]

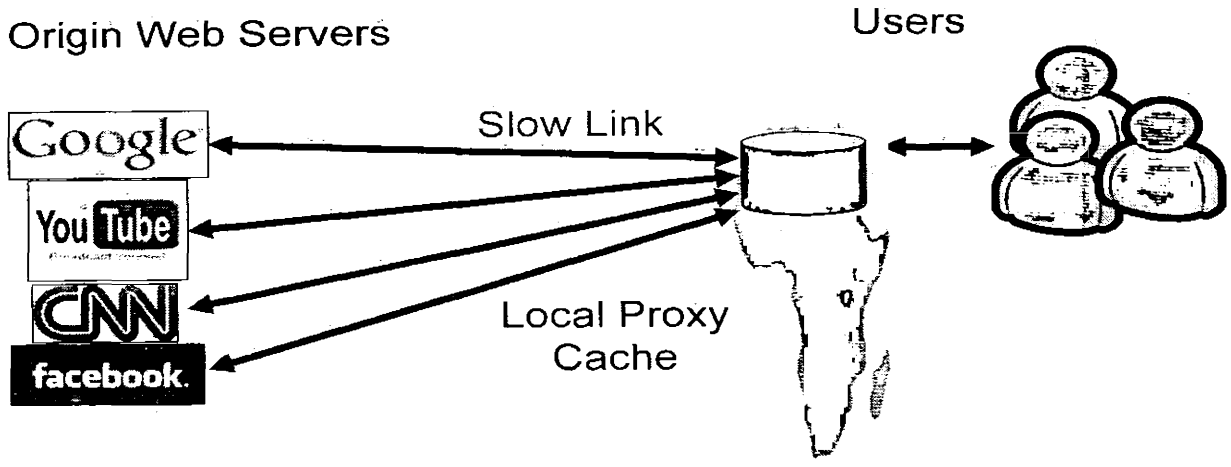


Figure 2.6 Wide Area Network Model [2]

Above figure shows that whole objects, designated cacheable traffic only. Zambia: 43% cache hit rate, 20% byte hit rate.

Wanax make the most of the mesh network surroundings being organized in the mounting world, as a substitute of just the star topologies common in venture branch offices.

- WAN acceleration focus on cashes misses such as Packet-level (chunks) caching
- Mostly for enterprise
- Two (or more) endpoints, coordinated

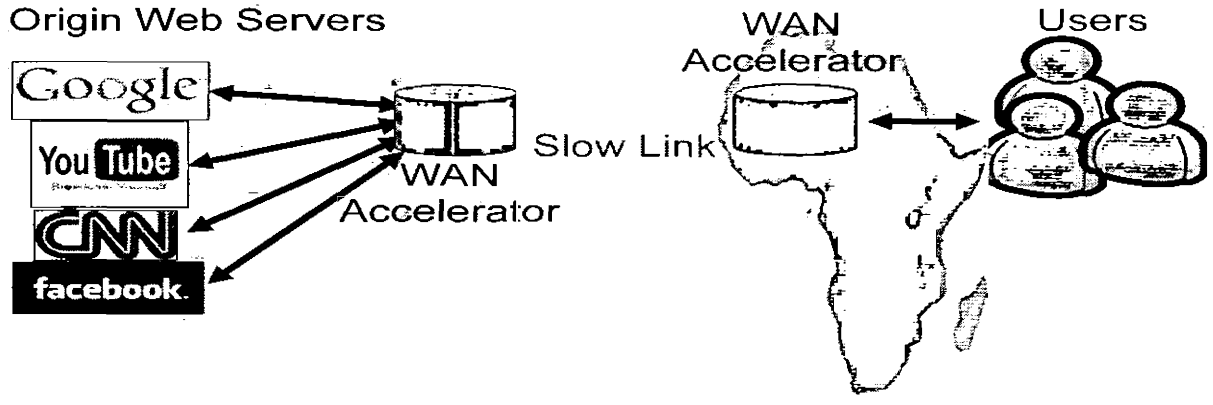


Figure 2.7 WAN Accelerator Model [2]

Content fingerprinting (CF) forms the basis forWAN acceleration, since it provides a position-independent and history-independent technique for breaking a stream of data into smaller pieces, or chunks, based only on their content

- Split content into chunks
  - Rabin's fingerprinting over a sliding window
  - Match n low-order bits of a global constant K
  - Average chunk size: 2n bytes
  - Name chunks by content (SHA-1 hash)
  - Cache chunks and pass references
- Chunk data
  - Usually stored on disk
  - Can be cached in memory to reduce disk access
- Chunk metadata index
  - Name, offset, length, etc.
  - Partially or completely kept in memory to minimize disk accesses

2.5.1 How It Works

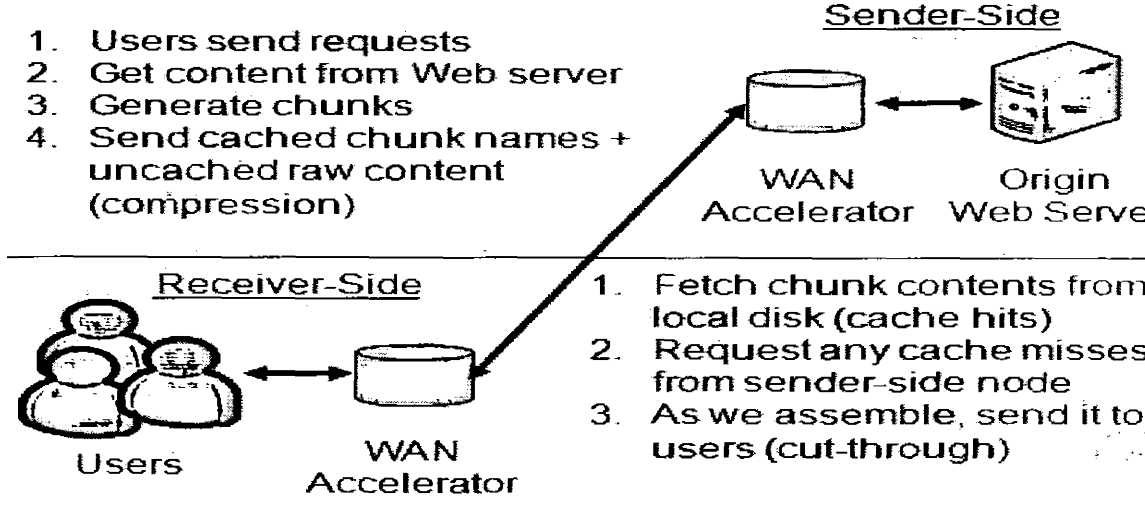


Figure 2.8 WAN Accelerator Working [2]

2.5.2 WAN Accelerator performance

- Effective bandwidth (throughput)
  - Original data size / total time
- Transfer: send data + refs
  - Compression rate is High
- Reconstruction: hits from cache
  - Disk performance is High
  - Memory pressure Low

2.5.3 Developing World Challenges

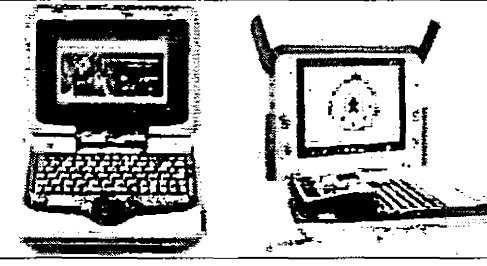
Enterprise	Developing World
O Dedicated machine with ample RAM	O Shared machine with limited RAM
O High-RPM SCSI disk	O Slow disk
O Inter-office content only	O All content
O Star topology	O Mesh topology
	

Table 3.2 Developing World Challenges [2]

2.5.7 WANAX: High-Performance WAN Accelerator

- Design Goals
  - Maximize compression rate
  - Minimize memory pressure
  - Maximize disk performance
  - Exploit local resources

- Contributions
  - Multi-Resolution Chunking (MRC)
  - Peering
  - Intelligent Load Shedding (ILS)

2.5.8 Scalable Parallel De duplication for chunk Based File Backup

Scalable de duplication technique is for non-traditional backup to build up individual files which do not have repeated files in same region in same time.

Due to lack of locality, accessible techniques do not perform properly on these workloads. Intense Binning exploits resembling file instead of locality, and make it possible that only one disk accessible to be seen for chunks in every file which ensures reasonable throughput.

Using stateless routing algorithm it is made possible to give each file to one node. which allows maximum parallelization, and each backup node is independent with no dependency across nodes.

Data de-duplication is an essential component for backup system which reduces storage space requirement and all operation which depend on its throughput. Authored proposed Extreme Binning parallel de-duplication technique which is suited for workloads that are made up of individual files with no locality among consecutive files And used file similarity instead of locality which access only one disk at time which provide reasonable throughput Chunk-based de-duplication is popular de-duplication Technique which first divides input data streams into fixed or variable-length chunks and chunk sizes are 4 to 8 kB Inline de-duplication is de-duplication where the data is de-duplicated before data write on disk temporary. In this technique extra disk space is not require for data In this paper author splits the chunk index into two Tiers the One tier is small in size and reside in RAM and second tier is reside on disk makes a single Disk access for chunk.[3]

TH-8367

2.5.9 Generating MRC Chunks

- Detect smallest chunk boundaries first
- Larger chunks are generated by matching more bits of the detected boundaries
- Clean chunk alignment + less CPU [2]

2.5.8 Storing MRC Chunks

Store every chunk regardless of content overlaps

- No association among chunks
- One index entry load + one disk seek
- Reduce memory pressure and disk seeks
- Disk space is cheap, disk seeks are limited

2.5.9 Peering

- Cheap or free local networks
  (ex: wireless mesh, WiLDNet)
  - Multiple caches in the same region
  - Extra memory and disk
- Use Highest Random Weight (HRW)
  - Robust to node churn
  - Scalable: no broadcast or digests exchange
  - Trade CPU cycles for low memory footprint

2.5.10 Intelligent Load Shedding (ILS)

- Two sources of fetching chunks
  - Cache hits from disk

- Cache misses from network

- Fetching chunks from disks is not always desirable
  - Disk heavily loaded (shared/slow)
  - Network underutilized
- Solution: adjust network and disk usage dynamically to maximize throughput

2.5.11 Simulation Analysis

- News Sites
  - Web browsing of dynamically generated Web sites (1GB)
  - Refresh the front pages of 9 popular Web sites every five minutes
  - CNN, Google News, NYTimes, Slashdot, etc.
- Linux Kernel
  - Large file downloads (276MB)
  - Two different versions of Linux kernel source tar file
- Bandwidth savings, and # disk reads

2.5.12 Evaluation

- Implementation
  - Single-process event-driven architecture
  - 18000 LOC in C
  - HashCache (NSDI'09) as chunk storage
- Intra-region bandwidths 100Mbps
- Disable in-memory cache for content
  - Large working sets do not fit in memory

- Machines
    - AMD Athlon 1GHz / 1GB RAM / SATA
    - Emulab pc850 / P3-850 / 512MB RAM / ATA

2.5.13 Conclusion from Wide Area Network Acceleration

- Wanax: scalable / flexible WAN accelerator for the developing regions
- MRC: high compression / high disk performance / low memory pressure
- ILS: adjust disk and network usage dynamically for better performance
- Peering: aggregation of disk space, parallel disk access, and efficient use of local bandwidth

Chapter 3

Problem Definition

3.0 Problem Definition

Mostly applications in the network require 10 Mbps or more bandwidth over LAN or campus area network therefore there is the need to develop a network file system which requires comparatively less bandwidth [1-3]. In the wireless network there is a big issue of the bandwidth due to many factors which does not effect in wired communication medium[2]. These factors may include fading, lightning and attenuation. Delays in the propagation are one of the many factors due to wireless nature. Wireless network now becomes the need of the moment. There are many application of wireless network in simple it can be said that the future is of wireless network. As the wireless networks are the desires of the time but there are two main issues to be resolved for wireless networks[2].

1. Bandwidth
2. Security

Due to wireless nature propagated signals faces reduction in power. Because signals are transmitted into the space so there are much chances of the intrusion. This intrusion must be avoided, especially when you need to transmit some security sort of information over the wireless network. In the scene this is not a big issue as compared to the wireless communication [25].

By keeping the two issues of the wireless networks there should be some sort of network file system which is efficient and bandwidth affected as well[1]. In the perspective of the end user security in some cases is not negligible but the bandwidth can be compromised to some extent. It is more considerable in the view of the network file system designers. Especially when there are interactive sessions the less bandwidth results into less efficiency and can halt too, like batch commands the execution time increases manifold [cryptographic access control]. File copying form remote site becomes a frustrating for long latency networks[1,2].

In the final statement we can say the wireless networks demands considerable security which is the main concern in secret information exchange and low bandwidth is required too.

Over slower networks interactive programs freeze, like during file I/O, batch commands the execution time increases manifold. In case of File I/O either user has to keep local copies of each or they can use remote login and edit the same file where it is placed. Remote login becomes frustrating for long latency networks [1,2,3].

Wireless networks require more security and having low bandwidth than traditional wired networks, so a file system is required which ensures reliable and low bandwidth network operations.

Chapter 4

Proposed Solution

4.0 Proposed Solution

Our proposed solution work as, First the hash code of the file is generated. Then the file is broken into two chunk of equal size and their hash code is generated. These chunks is further divided be into four chunks and hash code of each chunk is calculated and so on till an appropriate level k where $k=\log_2 n$ as in Figure 4.1.

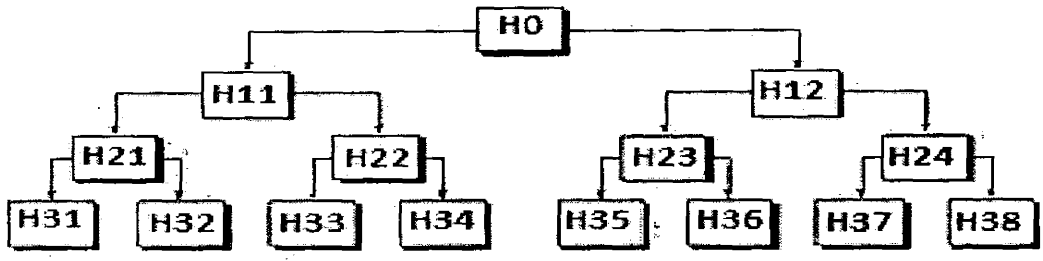


Fig 4.1: Hash code Tree generation

Hash code calculated for each chunk forms a hierarchal hash tree of chunks.

The file is encrypted each time before write back. Encrypted Hash code of the whole file is sent to client. Client decrypts it and match with the Hash code of file already present in his memory. If result is same, this implies that Server and Client having same version of the file, if result is different than client asks next level hash code of hash tree present in server memory, at each level hash codes are compared and the chunk having different data are identified, after identification of chunk whose data has been changed, client asks server to send the specific chunk

Encrypted data is transmitted on communication channel by server and client, If client & server data chunks produce same code, they do not transmit data.

4.1 Working of Proposed Solution

The file is encrypted each time before write back then Encrypted Hash code of the whole file is sent to client. Client decrypts it and match with the Hash code of file already present in his memory. If result is same, this implies that Server and Client having same version of the file, if result is different than client asks next level hash code of hash tree present in server memory, at each level hash codes are compared and the chunk having different data are be identified, after identification of chunk whose data has been changed, client asks server to send the specific chunk. Encrypted data transmitted on communication channel by server and client, If client & server data chunks produces same code, they do not transmit data.

If a client brought a file from the server in its local cache and did not do any work on it and started some other work. After some time it need to do some work on that file, but it is a possibility that may be during that time some other client had taken that file and updated that file or may not. To ensure consistency and freshness, it needs to consult the server to check whether the file is it in the same state when it took or it has been updated. Then it consults the server and send it File name and File handle as shown in Figure4.2.

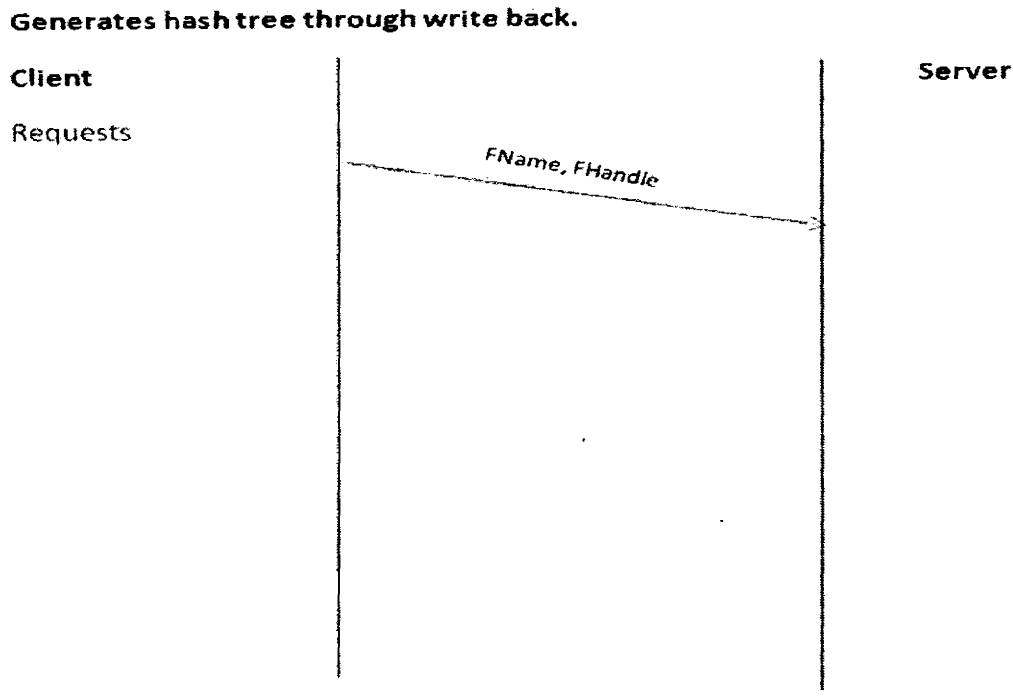


Figure 4.2 Client sends File name and File handle to Server

In return the server too sends file Name, File handle and the generated tree to the client as shown in Figure 4.3.

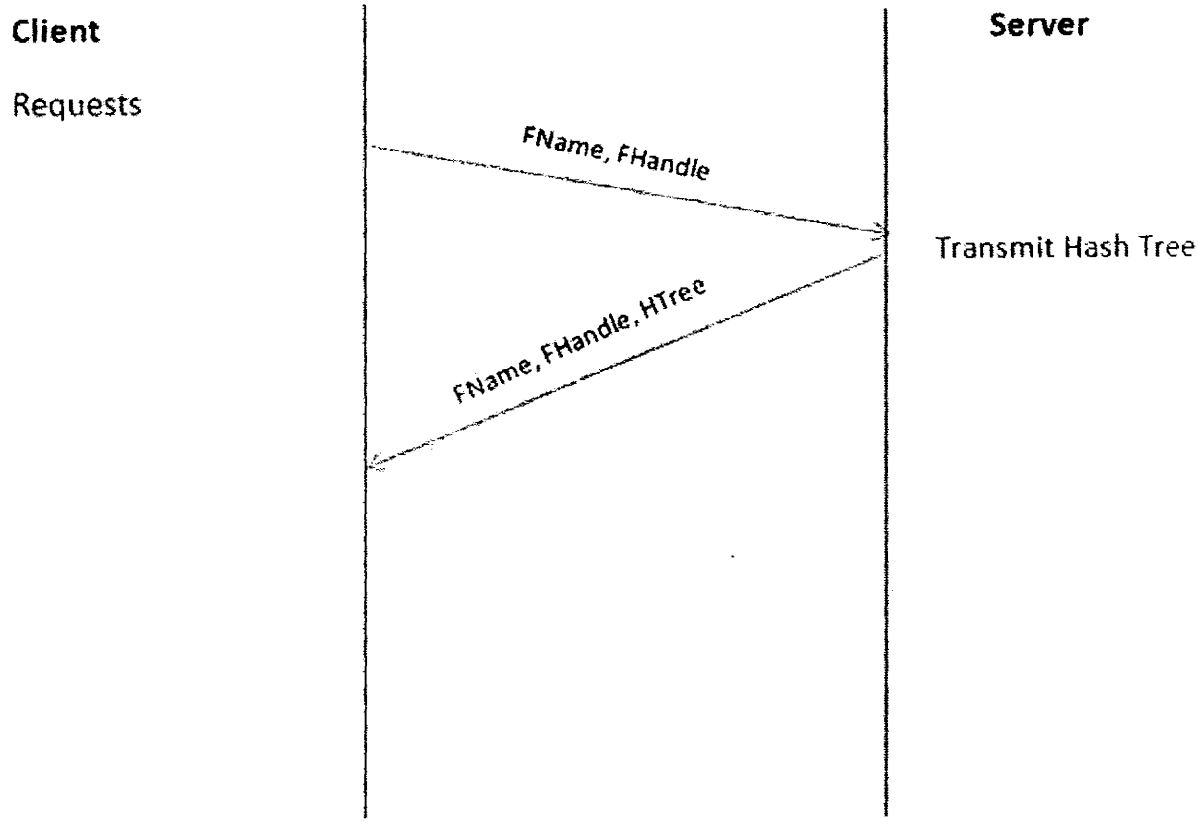


Figure 4.3 Client sends File name and File handle to Server

Then the client compares the both trees in hierarchal way and after that process it sends File Name, File Handle and the Number of the Hash Number mismatched and in return to that the server. As shown in Figure 4.4.

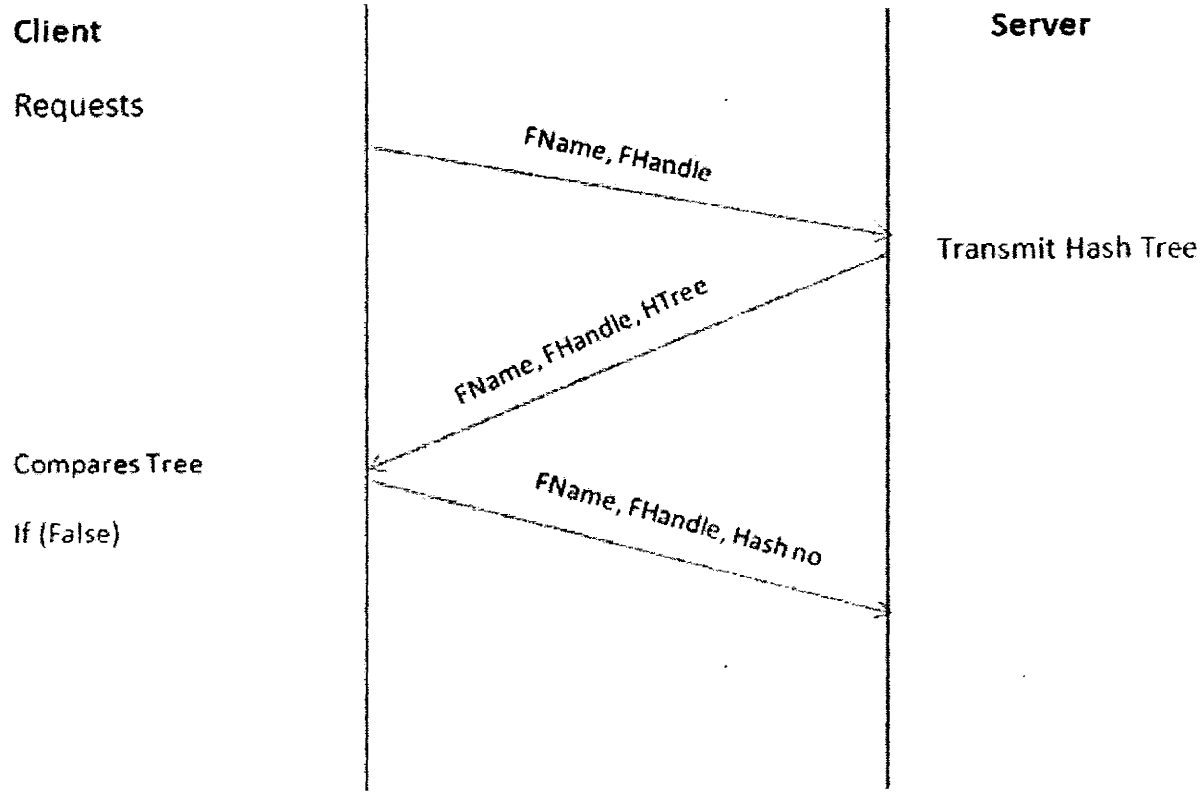


Figure 4.4 Client sends File name and File handle to Server

Then server gets the hash code of the mismatched part of the server and in return Server sends the updated data only instead of the whole file. As shown in Figure 4.5.

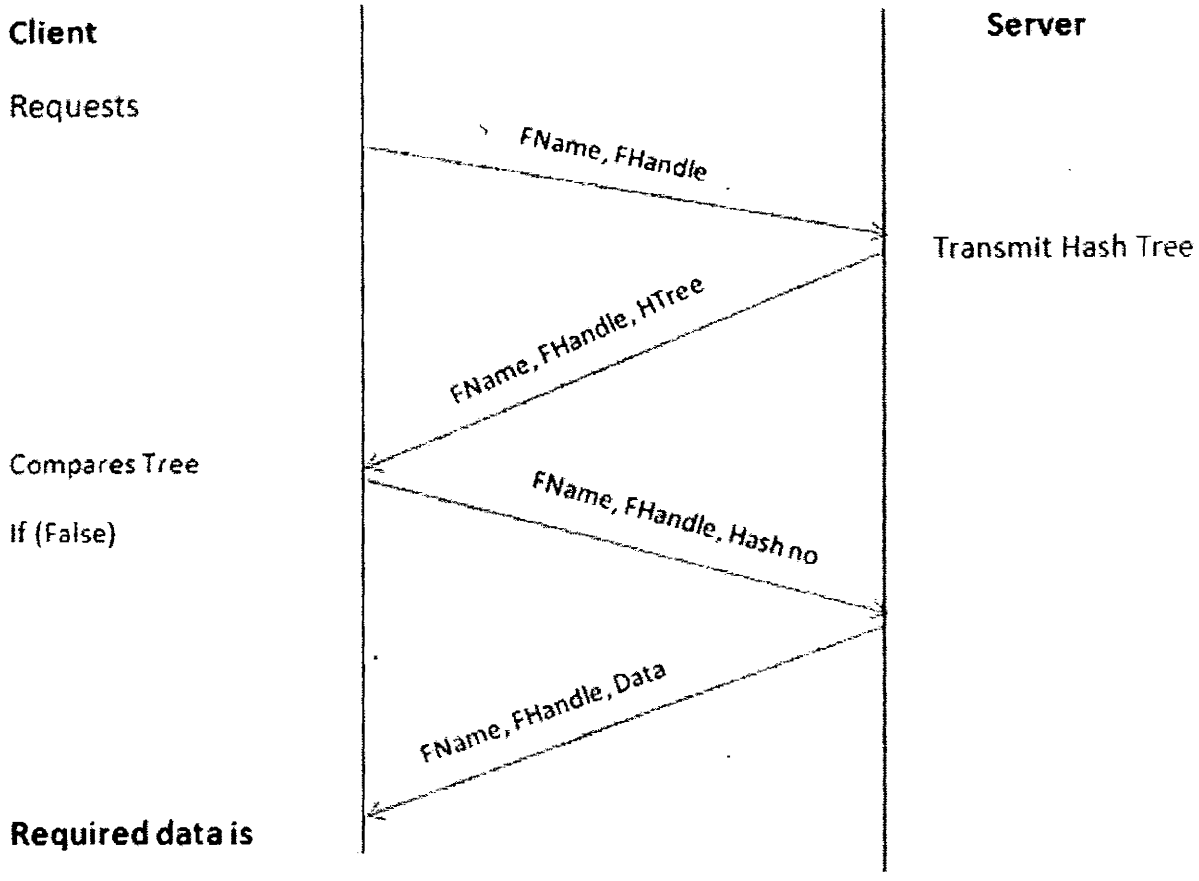


Figure 4.5 Server sends File name and File handle and data to Client

I have implemented all the graphically represented work in java language and After Implementing this graphically represented data in Java and done all the functionality like

1. First the hash code of the file is generated.
2. Then the file is broken into two chunk of equal size and their hash code is generated. These chunks is further divided be into four chunks and hash code of each chunk is calculated and so on till an appropriate level.
3. Hash code calculated for each chunk forms a hierarchal hash tree of chunks.
4. The file is encrypted each time before write back Encrypted Hash code of the whole file is sent to client. Client decrypts it and match with the Hash code of file already present in

his memory. If result is same, this implies that Server and Client having same version of the file, if result is different than client asks next level hash code of hash tree present in server memory, at each level hash codes are compared and the chunk having different data are identified, after identification of chunk whose data has been changed, client asks server to send the specific chunk Encrypted data is transmitted on communication channel by server and client If client & server data chunks produce same code, they do not transmit data.

And after implementing all above my code ensures the following results

1. File Consistency
    - If file is not in client cache or not up-to-date
    - Client fetches new version from server
2. Reading a file from server
3. Writing a file to server
4. Saving bandwidth by taking advantage of commonality between files (similar segments)
5. Securing the data
6. To ensure minimum running time for Hash matching using hierarchal hash tree
7. Providing message integrity

4.2 Security Mechanism

Encryption is used to transform plain text data into cipher text in order to conceal its meaning and to prevent any unauthorized recipient from retrieving the original data. Hence, encryption is mainly used to ensure secrecy. Proposed System encrypts data before transmission to ensure that the data is secure during transit. The encrypted data is sent over the public network and is decrypted by the intended recipient. Encryption is done by using the RC4 Algorithm. Both the sender and the receiver know this key which may be used to encrypt and decrypt the data.

4.3 Freshness and Integrity

Every time when file is different from the copy on the server data freshness is maintained and there is no needs to transmit the complete file again and again which will cause communication overhead over head, as only difference of the hash codes is sent.

Client calculates the hash code. After calculating the hash function on client and server side, the client will compare the calculated hash values with the hash values received from the server, if both received and calculated hash values are same then the received data is reliable. At the end of each successful communication session, the client will update the file to ensure the data freshness in each data communication session.

Proposed architecture can consume over an order of magnitude less bandwidth than traditional file systems and performs efficient use of communication link. It also makes transparent remote file access a viable and less frustrating alternative to running interactive programs on remote machines.

4.4 File Transfer Algorithm

Client Side	Server Side
1. Client Send request for file by File Name and File handle	
	2. Server receives the request 3. Generate Hash tree 4. Encrypt Hash tree using secret key 5. Transmit complete Hash Tree
6. Receive the Hash Tree	
7. Decrypt the received Hash Tree by secret key	
8. Generate Hash Tree and compare until whole Tree is matched or mismatch occur if file is already in Cache Else Create mismatch at level 0 i.e. first Hash	
9. Transmit mismatch Hash number along with File name and file Handle	
	10. Receive Mismatched Hash number

	11. Encrypt Mismatched Hash data 12. Transmit data of Encrypted chunk to requesting client
13. Receive data transmitted by server 14. Decrypt the data 15. Replace existing chunk data by newly received chunk data.	

4.5 Pseudo Code of Binary Tree

Purpose: Insert Hash Code sx into tree and make a first node as a Root

Inputs: Hash Code sx and value (to be inserted).

Effects: Compare value with Root value if value is less than it is placed in left side of root otherwise it is placed in right side.

```
RootValue (String sx)
{
  If (counts==1)
       Node root=new Node (sx);
       Temp=root
       Counts++;
  Else
       Node node=new Node (value);
       node=root
       Insert (node, sx)
}

Insert (node, value)
{
  If (value Compare To (node.value) <0)
     If (node. Left not equals to null)
```

```
        Insert (node. Left, value)
      Else
        node. Left = new Node (value)
Else if (value Compare To (node.value) >0)
      If (node. Right not equals to null)
      Insert (node. Right, value)
      Else
        node. Right = new Node (value)
Else
Do nothing
}
```

4.6 Pseudo Code of Hash Code Comparison

Purpose: Compare Client Hash Code with Server Hash Code.

Inputs: Client Hash Code Array (arrc) and Server Hash Code Array (arrsr).

Effects: If Client Hash Code matches with Server Hash Code than display otherwise send affected Hash Code to server for verifying.

```
Compare (arrc, arrsr)
{
  For i=0 to arrc. Length
    Str = arrc[i]
      For k=I to arrsr. Length
        Str1=arrsr[k]
      If(arrc[i] equal to null OR arrsr[k] equals to null)
            Do Nothing
            Exit from inner loop
          Else if(Str Equals To (Str1))
```

```
        Display
                Exit from inner loop
Else if(Str Not Equal To (Str1))
        Send affected Hash Code to server back
                Exit from inner loop
}
```

4.7 Pseudo Code of Data Send From Client to Server & From Server to Client

Purpose: Client send file to server for generation of Hash Code and after generation of Hash Code server send Hash Code file back to client.

Inputs: Any txt file (which is send from client to server).

Effects: Client sends any txt file to server. Server generates Hash Code and send Hash Code file to Client. If Client Hash Code does not match with server Hash Code than client resend the affected Hash Code to server, server return corrected Hash Code to client.

```
ClientSide ()
{
 Establish connection with server
 SendFileTo Server ()
 GeneratesHashCode ()
 ReceiveFileFromServer ()
        If (Client hash Code Not Equal to Server Hash Code)
                Sends Affected Hash Code Location to Server
        Else
                Display ("All hash Code matches")
}
```

```
ServerSide ()
{
Accept Client Request
RecieveFileFromClient ()
 GeneratesHashCode ()
 WriteHashCodeIntoFIle ()
 SendHashCodeFileToClient ()
        If (Hash Code Location Received From Client)
                Correct Hash Code ()
                SendHashCodeToClient ()
        Else
                Wait For Client Message
}
```

4.8 RC4 Algorithm for Encryption

Stream cipher symmetric key

Use two arrays, **state** and **key**

1. 256-byte state table.

State [256]=[0 .. 255]

2. It has the capability of using keys between 1and 2048 bits.

Key [1...2048] = [.....]

Two phases

Key Setup

1. f = (f + Si + Kg) mod 4

2. Swapping Si with Sf

Ciphering (XOR)

1. i = (i + 1) mod 4 , and f = (f + Si) mod 4

2. Swaping Si with Sf

3. t = (Si + Sf) mod 4

Random byte St

Chapter 5

Conclusion and Results

5.0 Results and Analysis

A customized simulator was developed in java language; files of type doc, txt, pdf and mdf were analyzed through simulator. The experiment was conducted on Intel core i5 system with 4 GB of RAM and 320 GB of hard disk drive; we used Microsoft word 2003, adobe acrobat 6.0, MS Access 2003 and notepad.

The file server and clients were connected through full duplex 100MB Ethernet through Dlink---- router.

The proposed architecture is content-based, that is it takes advantages of common contents between a versions of a file residing on different machines one on server others on clients. Proposed solution can only reduces bandwidth usage if and only if there is some overlapping in contents of version of a file, the amount of network traffic reduced is the only the common contents.

Table 5.1 summarizes the application type, file extension, original file size, and amount of new data, overlapping between two versions of a file, number of chunks / comparisons and percentage of transmitted data.

5.1 Results from Proposed System

Data	given	Data Size	New Data	Overlap	Number of Chunks/ Comparisons	Transmitted Data	
						KB	%
MS Word	.Doc	238kb	23kb	90%	7	60	26
Notepad	.txt	71kb	3kb	79%	11	4.44	6.25
Acrobat	.pdf	43kb	7kb	84%	7	10.75	25
MS Access	.mdb	129kb	52kb	60%	5	65	50

Table 5.1: Results calculated from Proposed System

A file of MS word of size 238 KB was reconstructed by seven comparisons and transmission of 60 KB data, which is 26% of whole file, similarly a file of extension .txt of size 71 KB was reconstructed by eleven comparisons and a transmission of 4.44 KB data, which is just 6.25%, An acrobat file of size 43 KB was reconstructed by seven comparisons and 10.75 KB of data. which is 25% of whole file, at last experiment was performed on a file of MS Access of size 129 KB, only 50% data was transmitted and only 5 comparisons were performed.

5.2 Results from Low Bandwidth File System

Data	given	Data Size	New Data	Overlap	Number of Chunks/ Comparisons	Transmitted Data	
						KB	%
MS Word	.Doc	238kb	23kb	90%	7	68	28.6
Notepad	.txt	71kb	3kb	79%	11	6.45	9.1
Acrobat	.pdf	43kb	7kb	84%	7	12.28	28.6
MS Access	.mdb	129kb	52kb	60%	5	77.4	60

Table 5.2: Results from Low Bandwidth File System

Table 2 is summary of analysis of LBFS , the experiment performed on same files as in proposed solution. Proposed solution transmitted 26% of data in case of MS Word file when 90% data overlapped, while LBFS transmit 28.6 % of data, in case of notepad file proposed solution

transmitted 6.25% while LBFS transmit 9.1% data, only 25% data was transmitted of acrobat file in proposed solution while LBFS transmit 28.6% data and proposed solution transmit 50% data while LBFS transmit 60% data.

5.3 Results from Network File system

Table 5.3 is summarization of data transmitted when Network file system was used, here in every case and for any file system data transmitted is 100%, that if only single character is changed in a file on server the client has to ask for whole file, which will be transmitted by server.

Data	given	Data Size	New Data	Overlap	Number of Chunks/ Comparisons	Transmitted Data	
						KB	%
MS Word	.Doc	238kb	23kb	90%	1	238	100
Notepad	.txt	71kb	3kb	79%	1	71	100
Acrobat	.pdf	43kb	7kb	84%	1	43	100
MS Access	.mdb	129kb	52kb	60%	1	129	100

Table 5.3 Results from Network File System

5.4 Comparison Graph of Proposed System LBFS and NFS

A comparison of Network file system, Low bandwidth network file system and proposed solution for four different applications is provided in Figure 5.1.

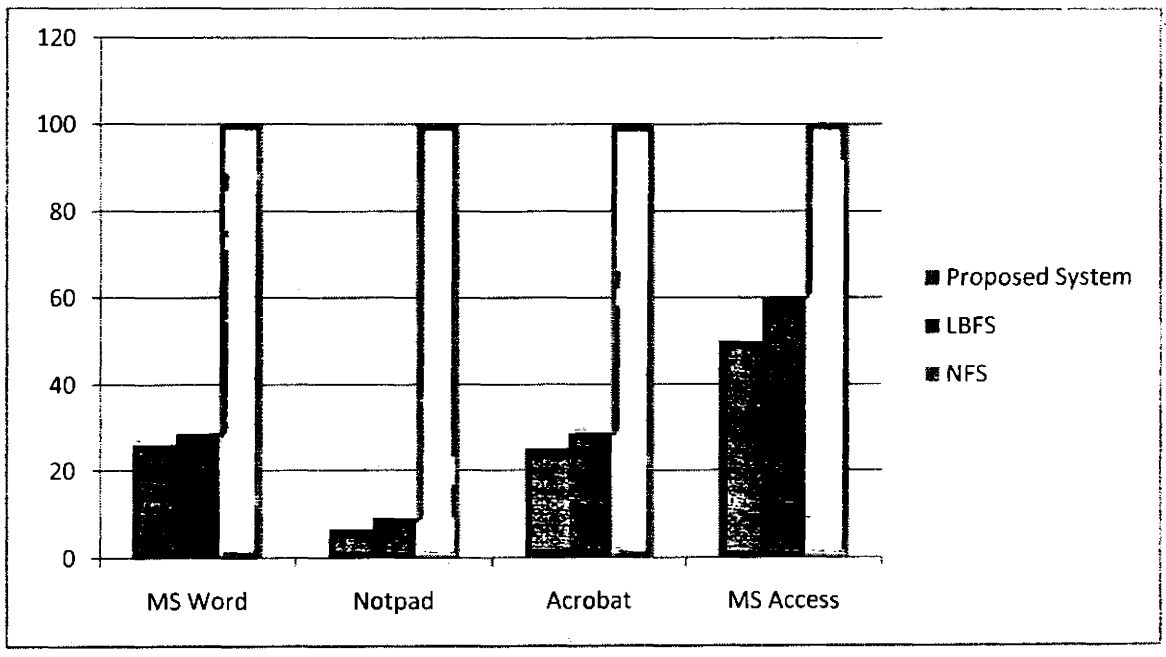


Figure 5.1: Comparison between NFS, LBFS and proposed System

5.5 Scenarios

5.4.1 Scenario 1.0 (Difference for Same Files)

In the first scenario I have taken a file in discussion which has not been changed and have the same copy on both server and client side. The hash code on both sides is generated and when comparison is started, on first comparison when the base hash H0 will match as shown in figure 5.2 it means that file is same on both sides and it will stop comparing hashes and do not need to bring file from server.

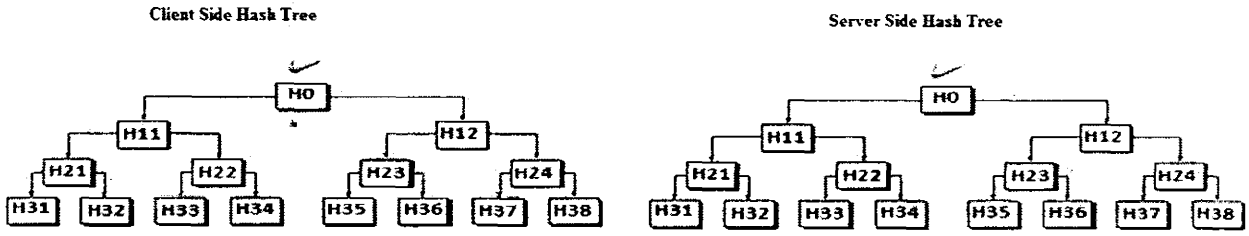


Figure 5.2: Scenario 1.0 (When Both Files are same)

5.4.2 Scenario 2.0 (When File has been changed)

In this scenario I took a file have 84% resemblance with the original file located at server. Hash of the file on both sides will be generated. It will continue comparing the hashes of both client and server until the child nodes of all the nodes at same level has been matched. After seven comparisons all the nodes at level 2 on client side has been found same as all the nodes at level 2 at server side as shown in figure 5.3, therefore it will stop comparing hashes and server will transmit the data having hash codes H11 and H12.

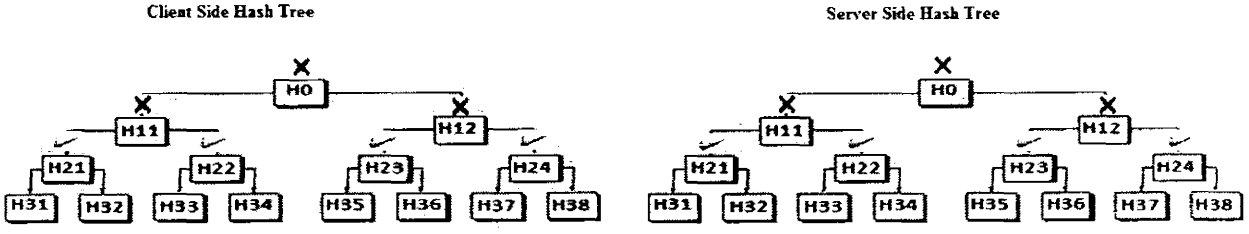


Figure 5.2: Scenario 2.0 (When File has been changed)

The result of this scenario is shown below in all three categories.

5.4.2.1 Results from Different Systems when file has been changed

Data	given	Data Size	New Data	Overlap	Number of Chunks/ Comparisons	Transmitted Data	
						KB	%
Acrobat	.pdf	43kb	7kb	84%	7	10.75	25

Table 5.4: Results for Proposed System when file has been changed

Data	given	Data Size	New Data	Overlap	Number of Chunks/ Comparisons	Transmitted Data	
						KB	%
Acrobat	.pdf	43kb	7kb	84%	7	12.28	28.6

Table 5.5: Results for LBFS when file has been changed

Data	given	Data Size	New Data	Overlap	Number of Chunks/ Comparisons	Transmitted Data	
						KB	%
Acrobat	.pdf	43kb	7kb	84%	1	43	100

Table 5.6: Results for NFS when file has been changed

5.4.3 Scenario 3.0 (When File is completely different)

In Scenario 3.0 a file has been taken which is totally different from the file located at server side. Hash of the file on both sides will be generated. It will continue comparing the hashes of both client and server until the child nodes of all the nodes at same level has been matched. After comparing all the hash codes it will find that none of the hash code matched with the hash code at server side as shown in figure 5.4, it implies that the file is totally different from the original file at server side. So the complete file from server will be sent. This is the worst case.

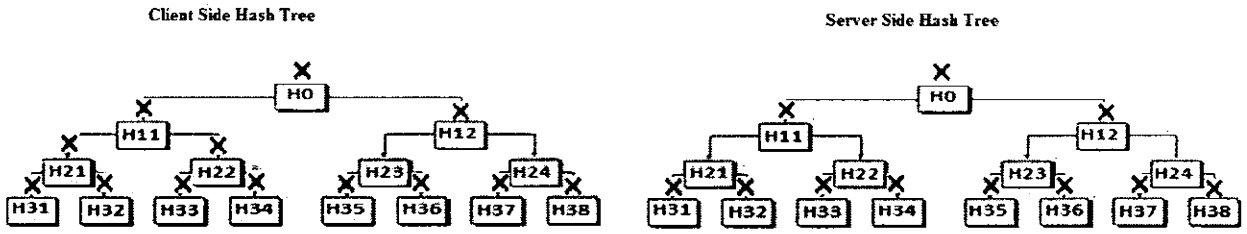


Figure5.4: Scenario 3.0(When File is completely different)

5.5 Conclusion

The proposed architecture can save bandwidth by taking advantage of commonality between files, ensures integrity and freshness of a remote file, Operations such as editing documents and compiling software, proposed architecture can consume over an order of magnitude less bandwidth than traditional file systems and performs efficient use of communication link. It also makes transparent remote file access a viable and less frustrating alternative to running interactive programs on remote machines. To ensure minimum running time for Hash matching we used hierarchal hash tree.

Simulation results are evident of the performance achieved for our proposed scheme. Proposed methodology can save maximum bandwidth by taking advantage of cross file commonalities, in best case analysis it will just transmit a packet containing just hash tree.

Chapter 6

Conclusion and Future Work

6.1 Conclusion and Future Work

The proposed architecture can save bandwidth by taking advantage of commonality between files, ensures integrity and freshness of a remote file, Operations such as editing documents and compiling software, proposed architecture can consume over an order of magnitude less bandwidth than traditional file systems and performs efficient use of communication link. It also makes transparent remote file access a viable and less frustrating alternative to running interactive programs on remote machines. To ensure minimum running time for Hash matching we used hierarchal hash tree.

Simulation results are evident of the performance achieved for our proposed scheme. Proposed methodology can save maximum bandwidth by taking advantage of cross file commonalities, in best case analysis it will just transmit a packet containing just hash tree.

Appendix-A

References

[1] Athicha Muthitacharoen, Benjie Chen, and David Mazi `eres,, "A Low-bandwidth Network File System" in Proc. 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04), pp. 81–88, Apr. 26–27, 2004

[2] S.Ihm, K. Park, V.S.Pai, "Wide Area Network Acceleration for the developing World", 2010

[3] Deepvali Bhagwat, Kave Eshghi, "Extreme Binning: Scalable Parallel Deduplication for chunk Based File Backup, 2009

[4] F. Tusa, M. Villari and A. Puliafito "Design and Implementation of an XML-based Grid File Storage System with Security Features", 2009

[5] Jonathan Ledlie, "File system for low-Bandwidth Thumbnails", Nokia research center Cambridge, US, May 6, 2008

[6] Xin Zhao, Atul Prakash "Securing Sensitive Content in a View-Only File System Kevin Borders" University of Michigan , 2006

[7] Anthony Harrington, Christian Jensen, "Cryptographic Access Control in a Distributed File System", ACM 2003

[8] S. Sobti, N. Garg, C. Zhang, X. Yu, A. Krishnamurthy, and Y. Wang. Personalraid: Mobile storage for distributed and disconnected computers. Proceedings of the FAST 2002 Conference on File and Storage Technologies, 2002.

[9] S. Sobti, N. Garg, F. Zheng, J. Lai, Y. Shao, C. Zhang, E. Ziskind, A. Krishnamurthy, and R. Wang. Segank: A distributed mobile storage system. Proceedings of the Third USENIX Conference on File and Storage Technologies, 2004

[10] David Mazi`eres. A toolkit for user-level file systems. In Proceedings of the 2001 USENIX Technical Conference, Boston, MA, June 2001.

[11] Haifeng Yu and Amin Vahdat. Design and evaluation of a continuous consistency model for replicated services. In Proceedings of the 4rd Symposium on Operating Systems Design and Implementation, San Diego, CA, 2000.

[12] Bj¨orn Gr¨onvall, AssarWesterlund, and Stephen Pink. The design of a multicast-based distributed file system. In Proceedings of the Third Symposium on Operating System Design and Implementation, pages 251–264, New Orleans, LA, February 1999.

[13] David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Westley Weimer, Christopher Wells, Ben Zhao, and John Kubiatowicz. Oceanstore: An exteremely wide-area storage system. In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, pages 190–201, Boston, MA, November 2000.

[14] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. ACMTransactions on Computer Systems, 18(4):263–297, November 2000.

[15] Andrei Broder, "Compression and Complexity of Sequences", pages 21–29, 1997

[16] Bj¨orn Gr¨onvall, AssarWesterlund, and Stephen Pink. The design of a multicast-based distributed file system. In Proceedings of the Third Symposium on Operating System Design and Implementation, pages 251–264, New Orleans, LA, February 1999.

[17] L. Huston and P. Honeyman. Disconnected operation for afs. Proceedings of the USENIX Mobile and Location-Independent Computing Symposium, 1993

[18] Yui-Wah Lee, Kwong-Sak Leung, and M. Satyanarayanan. Operation-based update propagation in a mobile file system. In Proceedings of the USENIX Technical Conference, Monterey, CA, June 1999.

[19] Chris Dahlberg, Harvey Mudd College "A survey of file systems for mobile computers" 2009

[20] Frank Schmuck, Roger Haskin "GPFS: A Shared-Disk File system for large computing cluster" 2003

[21] JH Howard "The ITC Distributed File System" 1999

[22] Matt Blaze "Key Management in an Encrypted File System" Proceedings of the USENIX Technical Conference 2002

[23] Patrick Eaton. Emil Ong. and John Kubiatowicz "Improving Bandwidth Efficiency of Peer-to-Peer Storage" IEEE 2004