Linux Peer-to-Peer File Sharing System

Technical Report No. CS04-10-00

Gilchrist Fortune Mushwana
gmushwan@cs.uct.ac.za

Mzondeleli Boltina
Mboltina@cs.uct.ac.za

Sebastian Rabele Melamu
Smelamu@cs.uct.ac.za

Supervised by:
Prof Ken MacGregor
ken@cs.uct.ac.za

ABSTRACT

In recent years, the evolution of innovative network architecture called peer-to-peer has been witnessed. Such systems are mainly characterized by direct access to computers or devices, rather than through centralized servers. According to the peer-to-peer working group, p2p is defined as the sharing of resources by direct exchange.

File sharing is the dominant application you can find on the internet today. These applications involve the sharing of resources and services with other computer system through direct exchange of information. Such resource includes information and content files, processing cycles, cache storage and disk storage. This kind of an application was made famous by the introduction of the server based file sharing application in 1999 called Napster. Napster was one of the first generation p2p systems to be created; today P2P computing has advanced towards the third generation P2P systems. Such a generation includes the likes of Chord [1], CAN [2] and Tapestry [3].

This paper is aimed at researching how P2P communication can be applied in a Linux operating system. This is done so by designing a file sharing system that allows computer systems to share each other memory. This system provides security, reliability and availability by distributing multiple encrypted files to a network of different PC's. We conclude by analyzing how successful this architect will be in future.

Categories and Subject Descriptors

H1 [operating systems]: Information storage and Retrieval systems.

Keywords

Linux peer-to-peer files system, personal computer, availability, reliability, security, trust and backup.

1. INTRODUCTION

Recently, a new model of communication and computation, called peer-to-peer networking, has started to gain significant acceptance [4]. Contrary to the traditional client-server model,

Peer-to-peer computing enables all clients to act as servers and all servers to act as clients. In this way, clients not only take a more active role in the information dissemination process, but also may significantly increase the performance and reliability of the overall system, by eliminating the traditional notion of the "server" which could be a single point of failure and a bottleneck in the overall system [5].

The first and most widely-known peer-to-peer system called Napster is considered the first file sharing utility that has enabled a hundreds of thousand of users to efficiently share files. Today the likes of windows have adopted this technique of sharing information. Windows allows personal computers to interconnect one another and share information securely and efficiently using its operating system.

In this paper we present an architect for distributing information in a server less distributed network of computers. This system will be deployed into Linux operating system to analyze how possible it to share file in Linux. Further more, to securely and reliably share information. This system will adopt the implementation of a backup system, whereby files are replicated, encrypted and stored in many places.

The remainder of this paper describes our system architecture, results, and analyzes these results in terms of our proposed architecture.

2. BACKGROUND PROBLEM

Peer-to-peer file sharing introduces an existing method of sharing which is currently applied on the internet and windows into the Linux operating system. This report is aimed at bringing sharing possible between computers without using a central server. The success of this architecture will be based on the implementation of three main components: Distribution of information, protocols for information discovery and security. To illustrate how data sharing will be achieved in this architecture, a network of peers is required. In this network, peers should be able to satisfy the three main components as indicated above

- Distribution of information: The distribution of information involves a situation where by a file has to be written or read away. As a result a decision has to be made on where to write or read it and how to retrieve or store it. If it's backing-up information, preferably this information has to be replicated and distributed into more than one place to achieve reliability and availability of the stored information. And if It involves getting a shared file, searching is highly important and this in most cases is determined by the structure of the network.

-Protocols for information discovery: protocols for information discovery involves the implementation of rules that will help other users to discover each other, bind with one another and share resources. Same kind of resources can be adopted from the JXTA platform.

-Security: Security in this case involves protecting the information distributed such that no peer is able edit it except its owner.

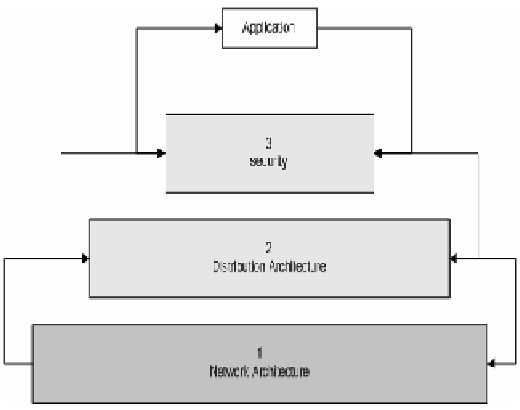

Figure 1High level architecture of the Linux peer-to-peer file sharing system

Application

This is the highest layer of this system. Every data sharing request is invented in this layer and then passed into the security layer.

Security

This layer receives information from the application layer, encrypts it and passes to the distribution layer.

Distribution Architecture

This layer defines rules that determine how replicated information should be distributed between computer systems and how to retrieve them to form the original replicated file.

Network Architecture

This is last layer of this system. This layer implements all the information discovery protocols such that peers are able to connect, publish, discover and share.

To solve this problem, the design of this architecture will be divided into four implementation components. The network, distribution, security and the application as described above. The next section describes the components of each design, including its implementation.

3. Goals

When designing our system, we adhered to the following goals:

- Reliability. Our top priority was to ensure that our system was as reliable as a secondary storage device. We want to be able to retrieve data even when some hosts holding our data are down.

- Storage space. We want to minimize the amount of data stored in each of our peers. We don't want full replication for our data.
- Speed of recovery. Recovery from a disk failure should be dominated by the network bandwidth and disk IO speed for writing the recovered files to disk.
- Ease of use. Users should be able to backup a file by inputting at the command prompt without having to worry about where his files are stored and the files should be automatically recovered in the event of disk failure without the user having to specify how or where from to recover the files.
- Cost. Our backup system should not cost the user any additional hardware or resources other than slack resources.

4. DESIGN

Linux peer-to-peer file sharing system provides distributed data storage and retrieval. It is structured as a collection of peers that share excess resources for storage. This system allows peers to send as much data as it can to other peers within this network. An algorithm is used to break files into blocks and store them in different destination peer nodes. In addition to this, this system also provides an offer for recovering the file in case it was accidentally deleted. The design of this system is made up of four components, namely: The network, distribution, security and application.

4.1 Network Architecture

Information, and the way in which it is represented and distributed, is fundamental to the success of any P2P network. Without the successful transmission of data between the peers that constitute the network nodes, P2P computing would not be feasible. This design is aimed at designing a distributed file sharing. This system should be able to provide distributed data storage and retrieval. The structure of the system is that of collection of peers sharing excess resources for storage. Each peer can store as much data as it wants in other peer's storage. In this system, there is no central server. Every user acts as both client and server. It should be able to discover users, files and store information in their memory. At the same time each user should also be able to provide files to other users connected to it. Basically the final product obtained in this design, provides a foundation to our back-up protocol discussed in the section.

This network will be designed based on three object, peer, storage, and group. In this section we propose a partially centralized hybrid network that allows users to form a group centrally controlled by an index node. Each node has storage and files.

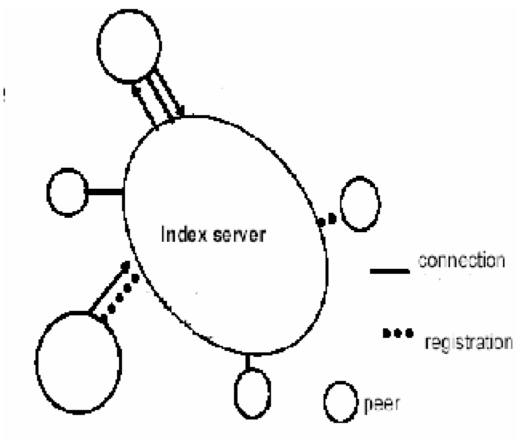

Figure 2 High level network design

The index node maintains a registry a of user information connected to it. It allows peers to quickly discover other peers within a group and to publish advertisements by making one request. A group of index nodes connect to one another to make a peer to peer network. In addition to this, the nodes can directly communicate with one another by creating a virtual direct communication.

Virtual direct communication is created by any peer to directly communicate with other peers. This is done by a peer directly requesting available peers from the index node and directly connecting to them. The index node is not a stand alone node; it performs all the tasks done by other peers and extra more management.

The implementation of this network is made of the client and server components.

Basically the client duties are to send request and receive result from any server. The server listens for request process them and sends results. Each peer is designed to run both at the same time.

4.1.1 Server

A class is written, which triggers every time the server gets an event. The server contains a list of functions that implement the required processing for the clients request.

4.1.2 Client

To interact with the system, the client provides a list of functions with requests to be processed by the server functions.

4.1. 3Peer

The peer class spawns the server and client processes, such that each peer is able to process both client and server requests. This allows each peer to publish and discover resources with others.

4.2 Distribution Architecture

This system is based on the concepts of distributed virtual memory where the files and information for each computer can be resident on another computer system. Each peer's storage device is divided into two:

- The user space – The user has full control of this space and can do anything he wants to do in it.
- The system space – The user has no control whatsoever of this space as it is the space donated for the system. All the backed up files of other users are stored in this space.

From the command line, the user can automatically backup a file without worrying about where the file is stored. Similarly when retrieving or deleting a file the user will just have to specify the name of the file and everything will be done automatically. This section is made up two parts, Back up and Recovery.

4.2.1 Back up process

The backup process consists of following steps:

- Specifying the name of the file you wish to backup
- Break up the file into multiple blocks
- For each block of file

    I. Hash the contents of the file block.
    II. Concatenate the contents of the file block with the hash (new block).
    III. Encrypt the new block.
    IV. Send the encrypted new block to peer.

- Save the status and location of the file.

4.2.1.1 Breaking up the file into multiple pieces

If the file the user entered exists the file is broken into blocks. The number of blocks in which the file is broken into depends on the number of peers in the network. The number of blocks in which the file is divided into is equal to the number of peers in the network. Therefore if there are six peers in the network, then the file will be divided into six blocks. This decision was based on the routing algorithm that I have come up with and will ensure that at most 50% of the storage peers can be faulty or damaged at the time of retrieval and the file will still be able to be retrieved.

4.2.1.2 Sending each piece to peer

To ensure reliability there must be replication. Our system supports exact-copy block replication to increase the reliability of a backup. Blocks are stored on several different peers, and if one peer fails then the blocks can be retrieved from any of the remaining peers.

4.2.1.3 How do the transferred blocks look like

After the file has been broken into blocks, the content of each block is still pure text(M). Some cryptographic hash function is applied on this pure text to produce some hashed text (H(M)). This hashed text is then concatenated with the pure text to form the new block. This new block is then encrypted using symmetric encryption. This encrypted block is the one that is stored in the network. The figure below illustrates the process described above

4.2.2 Recovery process

The recovery process is essentially the reverse of the backup process. The process follows these steps:

- Specifying the name of the file you wish to restore.
- Use the users Backup.txt file to retrieve the blocks from their location
- For each block recovered
    I. Decrypt the block
    II. Check if the block was tampered with
    III. Separate the hash from the contents
    IV. Reassemble the file

Each of the steps is explained in detail in the sub sections that follow

4.2.2.1 Specifying the name of the file you wish to restore

The user is presented with a text based menu where he can choose what operation he would like to achieve. After the user has chosen option 10 which is to retrieve a file he would be asked to enter the name of the file he would like to restore. If more than one version of the file is backed up the user is also asked for the version number. The Backup.txt file is then used to check whether the file has been backed up or not. If not the user is notified and taken back to the main menu.

4.2.2.2 Using the Backup.txt to retrieve the blocks and reassemble the file

If the file was backed up, the blocks will then have to be retrieved from their locations. Because Backup.txt tells us which block is stored which on location, for each block we are trying to retrieve we get all the locations where it has been saved. We go through the list of locations and we check if the peer at the location is active meaning that there is nothing faulty with it. If the peer is active we get the block and stop going through the list. If however we go through all the locations and none of the peers were active the user is told that the file could not be retrieved. On arrival of each block some cryptographic mechanisms are applied to the block to ensure that the file was not tampered with. If it appears that the file was tampered with the other location storing the same block is visited and the same checks are applied. For each block that is retrieved the hash is removed at the bottom such that we end up only with pure text. Once all the blocks are retrieved the blocks are concatenated to reassemble the file and the file is stored in the user's space.

4.3 Security

The main objective of the overall system is to design and develop a secure peer-to-peer file sharing system that will run on a Linux machine. Unlike other file sharing systems, this system allows users to share their own hard-drive space. This further enables the users to backup their files in other peer's memory storage. One of the main things to consider about security in file sharing systems is data integrity. To solve that, this system was designed work as follows:

A user joins the network by registering his/her computer as a peer and supplying all the required details. Once connected a user can discover other peers, backup file, retrieve backed up files, view other shared files and download any shared files. When a user shares of backs up his/her files the following occurs:

- The file is divided into blocks; the number of blocks is equal to the number of peers in the network e.g. if there are eight peers then each file is divided into eight blocks
- Each block is hashed
- The hash gets appended on the original block
- The new entire block is then encrypted and sent over the network to all the available peers.

The encrypted block is later sent over the network to destination peers for storage. A registry file is then kept for the retrieval process.

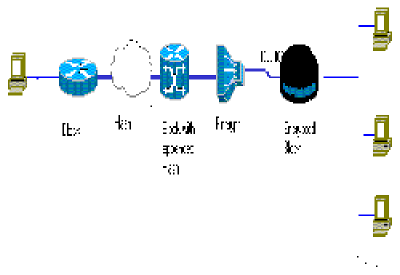

Figure 3 Entire system flow diagram

To implement security in this system the following classes were implemented namely, hashing class, encryption class and controller class.

4. 4.3.1 Hashing Class

The hash class has one major function i.e. hashing. The algorithm takes in a file that has maximum length of 264 bits and produces as output a 160-bit message digest. The input is processed in 512 bit blocks. The following steps describe how SHA-1 works.

Step 1: append padding bits. The block is padded so that its length is congruent to 448 modulo 512, this means that the padded block will be 64 bits less than a multiple of 512 bits.

Step 2: Append length. A block of 64 bits is then appended to the padded block. This 64-bit block is an unsigned 64-bit integer that contains the length of the original block.

These two steps yield a block that is an integer multiple of 512 bits in length.

Step 3: Initialize MD buffer. A 160-bit buffer is used to hold the intermediate and final results of the hash function. The buffer is implemented as five 32-bit registers.

Step 4: Process message in 512-bit block. This step makes use of the compression function that has four rounds of processing of 20 steps each. The four rounds have a similar structure but each uses a different primitive logical function.

Step 5: Output. After all the 512-bit blocks have been processed, the MD buffer outputs a 160-bit message digest.

4.3.2 Encryption Class

The encryption class encrypts and decrypts files to ensure confidentiality. The chosen algorithm (Blowfish) uses dynamic S-boxes, an XOR function and a binary addition. The S-boxes are generated as a function of the key.

The algorithm consists of two parts: a key-expansion part and a data- encryption part. Key expansion converts a key of at most 448 bits into several subkey arrays totalling 4168 bytes. Data encryption occurs via a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Subkeys

Blowfish uses a large number of subkeys. These keys are pre-computed before any data encryption or decryption. These keys are generated as a function of the key.

Encryption

The input is a 64-bit data block, this means if a block is bigger than 64-bits then it will be divided and padded as needed. The XOR function is used along with the binary addition to mix the original bits with the subkey bits. The process finishes after 16 rounds.

Decryption

Decryption is exactly the same as encryption, except that the subkeys are used in the reverse order.

4.3.3 Controller Class

This class makes use of the hash class and the encryption class. It takes in the blocks to be encrypted, call the hashing method to hash the block and append the hash into the original block. It then calls the encryption method in the encryption class to encrypt the block. Encryption is complete and the file can be backed up. If the file is to be shared, then only the hash will be encrypted and appended in the original block.

For retrieval, the block is decrypted first. Original hash is removed and a new hash is computed. The two hashes are then compared and if they are identical then the block is verified.

In the case of downloading a shared file, only the verification is done. The original hash is removed, a new hash is computed and the two hashes are compared. If they are identical then the block is authentic.

4.4 Application

This is the highest level of the system. It provides the users with a menu to directly interact with the file sharing system.

5. Performance

We have tested the network and distribution layer on eleven Pentium 4 processor machines with 200GHz and 516 MB RAM connected through 100Mbps LAN and presented our findings below.

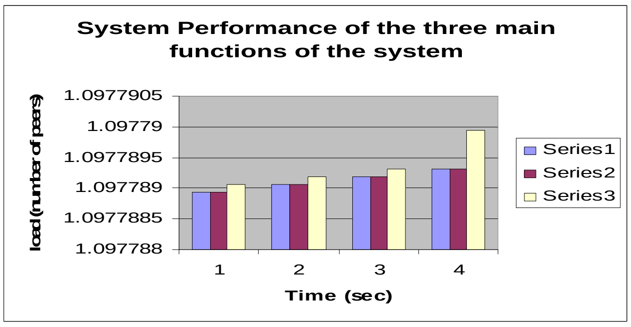

Figure 4 System performances on three functions of the network. Peer discovery, store and get files

Active Peers	Non Active Peers	Percentage of file retrieved
0	9	0%
1	8	60%
2	7	70%
3	6	80%
4	5	90%
5	4	100%

Table 1: Table showing results from a test with a network of peers

Active Peers	Non Active Peers	Percentage of file retrieved
0	10	0%
1	9	64%
2	8	73%
3	7	82%
4	6	91%
5	5	100%

Table 2 : Table showing results from a test with a network of peers

According to figure 4, a partially centralized hybrid network performance is indirectly proportional to the load factor of the network. This is due to the fact that when more users join the network, the load of work of the index node increases and its performance reduces too. In this case the results are highly affected by the assumption that all users are declared to have equal capabilities in terms of system power. As a result, if each user is given the token of being an index node based on merit. The results would improve from been a straight line to a concave distribution. To improve on these results, the number of user joining a GROUP should be based on a certain percentage benchmark such that index nodes don't get bottlenecked and abused by other users.

From table 1 it comes out that even if at most 44% of the peers in the network could be faulty or malicious during any time of the retrieval, file will still be restored.

The results that follow are the results obtained from a network of 11 peers with 10 storage peers. And From these results it comes out that even if at most 50% of the peers in the network could be faulty or malicious during anytime of the retrieval, the file will still be restored.

Comparing the results from the two tables it is clear that the system is more reliable when there is an even number of storage peers meaning an uneven number of peers in the network.

6. Related Work

Linux File Sharing System was inspired by the abundance in peer-to-peer systems today such as Gnutella [4] and Kazaa [6].

The skyrocketing capacity of disk storage and the corresponding decrease in cost has led to very natural choice to consider storing data on slack resources at peers.

The Cooperative File System(CFS) is a peer-to-peer read-only storage system that provides provable guarantees for the efficiency, robustness, and load balance of file storage and retrieval [7]. The basic unit of storage in CFS is a block. CFS provides a distributed hash table using Chord for block storage.

OceanStore is a utility infrastructure designed to span the globe and provide continuous access to persistent information [8]. The authors of OceanStore envisioned a utility model in which consumers pay a monthly fee in exchange for access to persistent storage. This is not dissimilar to Internet backup sites offering reliable storage of data. It's not clear why any user would want to

store their data globally, although such a system guarantees persistent data in the face of catastrophe to continents [9]

7. Future Works

This paper describes a file sharing system. This system deploys a partially centralized network to connect computer systems. Reliability is always a problem when centralization is involved, investigating other topologies like CHORD can prove better for such a system.

Our system allows different versions of a file to be backed up in the network. Each version of a file is treated as a different file. Our system could be improved by storing only incremental changes between different versions.

8. Conclusion

Peer-to-peer networking introduces a way of allowing users to create a network without relying on servers. This project creates a prototype of how this networking can be implemented in Linux operating system. Not only does it do that, it also investigates how information can be distributed and how resources can be shared amongst a network of users. In this work, we have seen how computer systems can be connected to one another using this approach and the results and benefits of this type of communication. Not all organizations can afford servers; peer-to-peer gives all organization an ability to implement without expensive server. Further more, this project highlights how the utilization of unused memory storage can be achieved by applying peer-to-peer computing to share data.

This is simple prototype of a file sharing and it has proven a successful achieving our goals. This prototyped has been further evaluated and the future deployment of such a system in a better network will prove better for future opportunities, but the current result indicate that storing files at peer hosts is a viable option for a backup system.

References

[1] I. Stoica., R., Morris., D., Karger., M., Kaashoek., K and Balakrishnan., H. Chord: A Scalable Peer-to-peer Lookup Service.

Available at:

http://citeseer.ist.psu.edu/cache/papers/cs/32722/http:zSzzSznms.lcs.mit.eduzSzpaperszSzchord.pdf/stoica01chord.pdf

[2] D. Tucker. Survey of Searching Methods in Internet Peer-to-Peer Systems. Available from: http://www.cs.kent.edu/~javed/DL/surveys/IAD03F-dtucker/

[3] B. Zhao.Supporting Rapid Mobility via Locality in an Overlay Network.

Available from: http://www.cs.berkeley.edu/~ravenben/tapestry/MobileTapestry.pdf

[4] C. Yang, peer to peer networks. Available at:

www.aect.cuhk.edu.hk/~ect7010/ Materials/Lecture/Lec6.pdf

[5] B. Yang and H. Garcia-Molina. Comparing hybrid peer-to-peer systems. In Proc. of the 27th Intl. Conf. on Very Large Databases, September 2001

[6] Kazaa website.

http://www.kazaa.com

[7] F. Dabek, M. F. Kaashoek, D. Karger,

R. Morris, and I. Stoica. Wide-area

cooperative storage with CFS. In 18th
ACM Symposium on Operating Systems
202–215.

[8] J. Kubiatowicz et al. Oceanstore: An
architecture for Global-Scale Persistent
Storage. In 17th ACM Symposium on
Operating Systems Principles
(SOSP ’00), November 2000, pp. 190–
201.

[9]Gattu., N.,Huang., R.,Lynn.,J and Xia., H. Magnus: Peer to Peer Backup System. Available at:

www-csag.ucsd.edu/individual/huaxia/
academic/classes/sp03/cse291/Sp03_291Project.pdf

The text should be in two 8.45 cm (3.33") columns with a .83 cm (.33") gutter.

5. TYPESET TEXT

5.1 Normal or Body Text

Please use a 9-point Times Roman font, or other Roman font with serifs, as close as possible in appearance to Times Roman in which these guidelines have been set. The goal is to have a 9-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code text. If Times Roman is not available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged.

5.2 Title and Authors

The title (Helvetica 18-point bold), authors' names (Helvetica 12-point) and affiliations (Helvetica 10-point) run across the full width of the page – one column wide. We also recommend phone number (Helvetica 10-point) and e-mail address (Helvetica 12-point). See the top of this page for three addresses. If only one address is needed, center all address text. For two addresses, use two centered tabs, and so on. For more than three authors, you may have to improvise.[1]

5.3 First Page Copyright Notice

Please leave 3.81 cm (1.5") of blank text box at the bottom of the left column of the first page for the copyright notice.

5.4 Subsequent Pages

For pages other than the first page, start at the top of the page, and continue in double-column format. The two columns on the last page should be as close to equal length as possible.

Table 1. Table captions should be placed above the table

Graphics	Top	In-between	Bottom
Tables	End	Last	First
Figures	Good	Similar	Very well

5.5 References and Citations

Footnotes should be Times New Roman 9-point, and justified to the full width of the column.

Use the standard Communications of the ACM format for references – that is, a numbered list at the end of the article, ordered alphabetically by first author, and referenced by numbers in brackets [1]. See the examples of citations at the end of this document. Within this template file, use the style named references for the text of your citation.

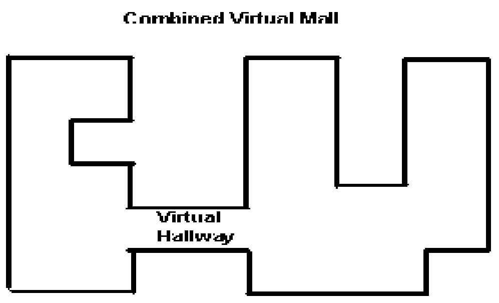

Figure 1. Insert caption to place caption below figure.

The references are also in 9 pt., but that section (see Section 7) is ragged right. References should be published materials accessible to the public. Internal technical reports may be cited only if they are easily accessible (i.e. you can give the address to obtain the report within your citation) and may be obtained by any reader. Proprietary information may not be cited. Private communications should be acknowledged, not referenced (e.g., "[Robertson, personal communication]").

5.6 Page Numbering, Headers and Footers

Do not include headers, footers or page numbers in your submission. These will be added when the publications are assembled.

6. FIGURES/CAPTIONS

Place Tables/Figures/Images in text as close to the reference as possible (see Figure 1). It may extend across both columns to a maximum width of 17.78 cm (7").

Captions should be Times New Roman 9-point bold. They should be numbered (e.g., "Table 1" or "Figure 2"), please note that the word for Table and Figure are spelled out. Figure's captions should be centered beneath the image or picture, and Table captions should be centered above the table body.

7. SECTIONS

The heading of a section should be in Times New Roman 12-point bold in all-capitals flush left with an additional 6-points of white space above the section head. Sections and subsequent sub-sections should be numbered and flush left. For a section head and a subsection head together (such as Section 3 and subsection 3.1), use no additional space above the subsection head.

7.1 Subsections

The heading of subsections should be in Times New Roman 12-point bold with only the initial letters capitalized. (Note: For subsections and subsubsections, a word like *the* or *a* is not capitalized unless it is the first word of the header.)

7.1.1 Subsubsections

The heading for subsubsections should be in Times New Roman 11-point italic with initial letters capitalized and 6-points of white space above the subsubsection head.

7.1.1.1 Subsubsections

The heading for subsubsections should be in Times New Roman 11-point italic with initial letters capitalized.

[1] If necessary, you may place some address information in a footnote, or in a named section at the end of your paper.

7.1.1.2 Subsubsections

The heading for subsubsections should be in Times New Roman 11-point italic with initial letters capitalized.

8. ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

9. REFERENCES

[1] Bowman, B., Debray, S. K., and Peterson, L. L. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst., 15*, 5 (Nov. 1993), 795-825.

[2] Ding, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[3] Fröhlich, B. and Plate, J. The cubic mouse: a new device for three-dimensional iput. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '00)* (The Hague, The Netherlands, April 1-6, 2000). ACM Press, New York, NY, 2000, 526-531.

[4] Lamport, L. *LaTeX User's Guide and Document Reference Manual.* Addison-Wesley, Reading, MA, 1986.

[5] Sannella, M. J. *Constraint Satisfaction and Debugging for Interactive User Interfaces.* Ph.D. Thesis, University of Washington, Seattle, WA, 1994.

Columns on Last Page Should Be Made As Close As Possible to Equal Length