Reliability Analysis in Net-centric Computing

Term 2 report

Supervisor

Professor LYU Rung Tsong Michael

Markers

Professor FU Wai Chee Ada
Dr. MOON Yiu Sang

prepared by

Chong Ka Lung
(98080070)

Department of **C**omputer **S**cience and **E**ngineering
The **C**hinese **U**niversity of **H**ong **K**ong

Table of Content

Abstract:

Nowadays, net-centric computing is growing important. It is an emerging model for client/server computing. Due to its lower cost and robust capabilities, it is useful in building intranet based network. It provides many advantages than the traditional client/server computing and its easy maintenance and administration make it popular. Cost/effective study on the comparison between net-centric computing environment and traditional client/server computing environment. Reliability and Fault Tolerance Reliability is a key component in today's network. A reliability model using fault tree is used to evaluate the reliability. It provides evidence on a better reliability is obtained. In addition, applying fault-tolerant technique to net-centric computing environment empowers the reliability and availability of it.

2. Introduction:

In traditional client/server computing, there are clients and servers. Client, typically a PC or a workstation, performs processing associated with an user interface and applications processing that can be done locally (e.g. word processing) while the server, which may be anything from PC to a supercomputer, performs processing tasks in support of its client (e.g. storage and retrieval of a centralized database).

Net-centric computing is a new paradigm of client/server computing. It is a model in which applications are resided, managed, supported and executed on a server. It uses a multi-user operating system and a method for distributing the presentation of an application's interface to a client device. It is different from the traditional client/server computing and network computing.

The benefits on net-centric computing are single-point management, physically and technically secure, predictable ownership costs, bandwidth-independent performance and universal application access. The total cost of ownership is greatly reduced and the maintenance job is much easier. Therefore, it is rapidly becoming the most reliable way to reduce the complexity and total costs.

As the total cost of ownership is greatly reduced, will the performance and reliability of the system be affected? This is a crucial issue on net-centric computing. Performance measures and reliability issues are discussed. A comparison in the reliability between traditional client/server computing and net-centric computing is in the remaining sections.

The net-centric architecture will be discussed in section 3. Net-centric computing can make use of the wireless technology as well as wire network. Using a wireless network interface card in the network computing devices, they can connect to the net-

centric server via the access point. The advantages of using wireless LAN are discussed in section 3.2.

The configuration of a demo system is discussed in section 4. The cost/effective study on the net-centric computing environment and the comparison between traditional client/server and net-centric computing environment in section 5. A performance study is in section 6 and some weakness on the evaluation will be stated out. Reliability is one of the important issues in net-centric environment. An analytic reliability model using fault tree diagram is built to analyze the architecture together with some hypothetical results will be shown in section 7. Fault tolerance is another issue in this architecture. Several fault tolerant techniques such as N-version programming, recovery block scheme, reusable software component and server farm concept will be introduced in section 8. Ongoing works will state out in section 9 and a summary in section 10. References are listed out in section 11.

3. Net Centric architecture:

3.1. Descriptions:

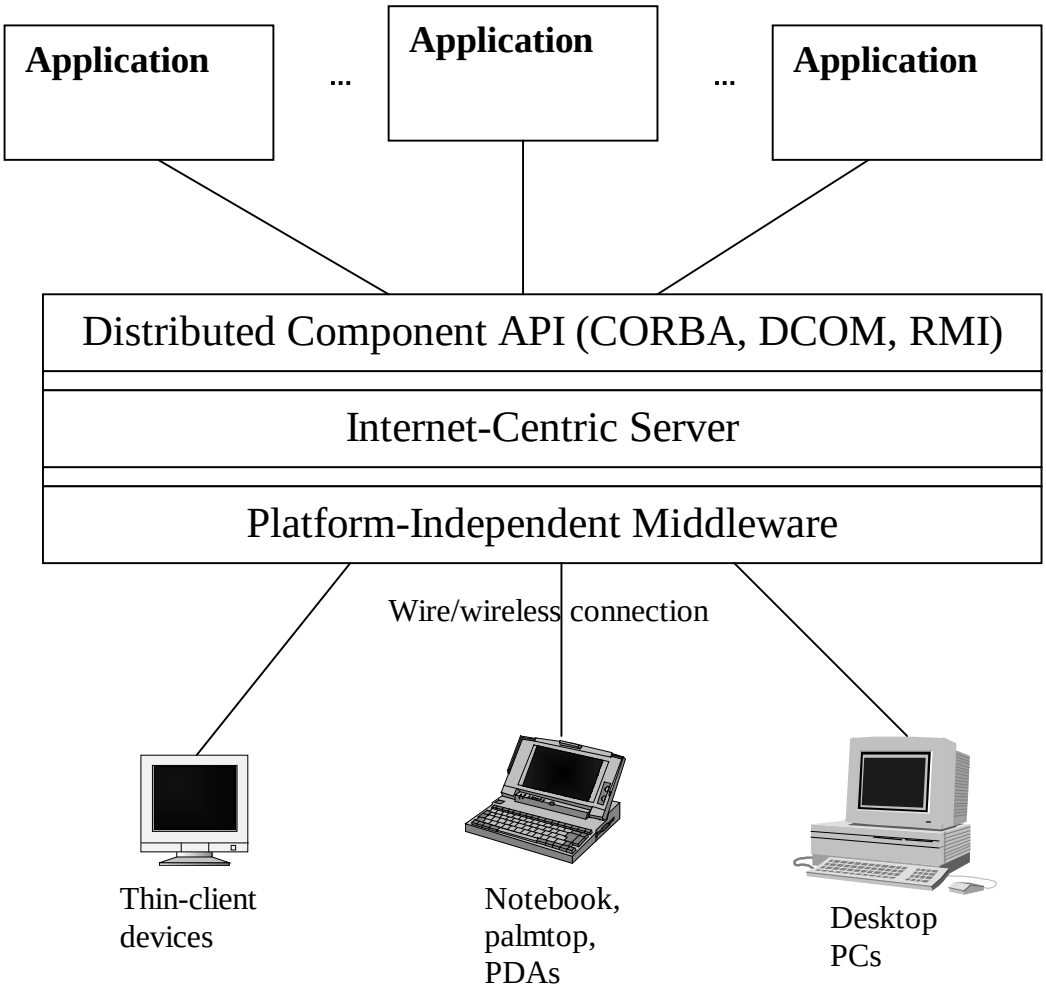

Fig. 1: Net-centric architecture

The net-centric architecture consists five components: client devices, platform-independent middleware, internet-centric server, distributed component API and applications resided in the server. Client devices can be desktop PCs, notebook, palmtop, personal digital assistant (PDA) and thin-client devices. They are used to connect to the internet-centric server through the wire or wireless connection.

There is a platform-independent middleware which supports for heterogeneous computing environments to connect to the server. Citrix MetaFrame is a product which acts as a middleware client devices and internet-centric server. The middleware is resided in the server.

The internet-centric server which is the place for processing tasks and for centralized database. The distributed component API (CORBA, DCOM, RMI) is to make the components sharable.

The applications are in the server side and execute when there is client request on the application. The server will process the task by executing particular application and after the processing completed, it sends back the screen updates to the client.

3.2. What is a Wireless LAN?

A wireless local area network (LAN) [1] is a flexible data communications system implemented as an extension to, or as an alternative for, a wired LAN. Using radio frequency (RF) technology, wireless LANs transmit and receive data over the air, minimizing the need for wired connections. Thus, wireless LANs combine data connectivity with user mobility.

Wireless LANs have gained strong popularity in a number of vertical markets, including the health-care, retail, manufacturing, warehousing, and academia. These industries have profited from the productivity gains of using hand-held terminals and notebook computers to transmit real-time information to centralized hosts for processing. Today wireless LANs are becoming more widely recognized as a general-purpose connectivity alternative for a broad range of business customers.

The widespread reliance on networking in business and the meteoric growth of the Internet and online services are strong testimonies to the benefits of shared data and shared resources. With wireless LANs, users can access shared information without

looking for a place to plug in, and network managers can set up or augment networks without installing or moving wires. Wireless LANs offer the following productivity, convenience, and cost advantages over traditional wired networks:

- Mobility: Wireless LAN systems can provide LAN users with access to real-time information anywhere in their organization. This mobility supports productivity and service opportunities not possible with wired networks.
- Installation Speed and Simplicity: Installing a wireless LAN system can be fast and easy and can eliminate the need to pull cable through walls and ceilings.
- Installation Flexibility: Wireless technology allows the network to go where wire cannot go.
- Reduced Cost-of-Ownership: While the initial investment required for wireless LAN hardware can be higher than the cost of wired LAN hardware, overall installation expenses and life-cycle costs can be significantly lower. Long-term cost benefits are greatest in dynamic environments requiring frequent moves and changes.
- Scalability: Wireless LAN systems can be configured in a variety of topologies to meet the needs of specific applications and installations. Configurations are easily changed and range from peer-to-peer networks suitable for a small number of users to full infrastructure networks of thousands of users that enable roaming over a broad area.

An access point can extend the range of an ad hoc network, effectively doubling the range at which the devices can communicate. Since the access point is connected to the wired network each client would have access to server resources as well as to other clients. Each access point can accommodate many clients; the specific number depends on the number and nature of the transmissions involved. Many real-world applications exist where a single access point services from 15-50 client devices.

Access points have a finite range, on the order of 500 feet indoor and 1000 feet outdoors. In a very large facility such as a warehouse, or on a college campus it will

probably be necessary to install more than one access point. Access point positioning is accomplished by means of a site survey. The goal is to blanket the coverage area with overlapping coverage cells so that client might range throughout the area without ever losing network contact. The ability of clients to move seamlessly among a cluster of access points is called roaming. Access points hand the client off from one to another in a way that is invisible to the client, ensuring unbroken connectivity.

Flexibility and mobility make wireless LANs both effective extensions and attractive alternatives to wired networks. Wireless LANs provide all the functionality of wired LANs, without the physical constraints of the wire itself. Wireless LAN configurations range from simple peer-to-peer topologies to complex networks offering distributed data connectivity and roaming. Besides offering end-user mobility within a networked environment, wireless LANs enable portable networks, allowing LANs to move with the knowledge workers that use them.

4. Configuration:

The demo system of net-centric configuration include a dual-process NT server, thin-client and network computing devices (including PDAs, laptops, sub-laptops), wireless access points, and client/server middleware software for fast server accessing and computation. Figure 2 depicts the configuration of net-centric computing used.

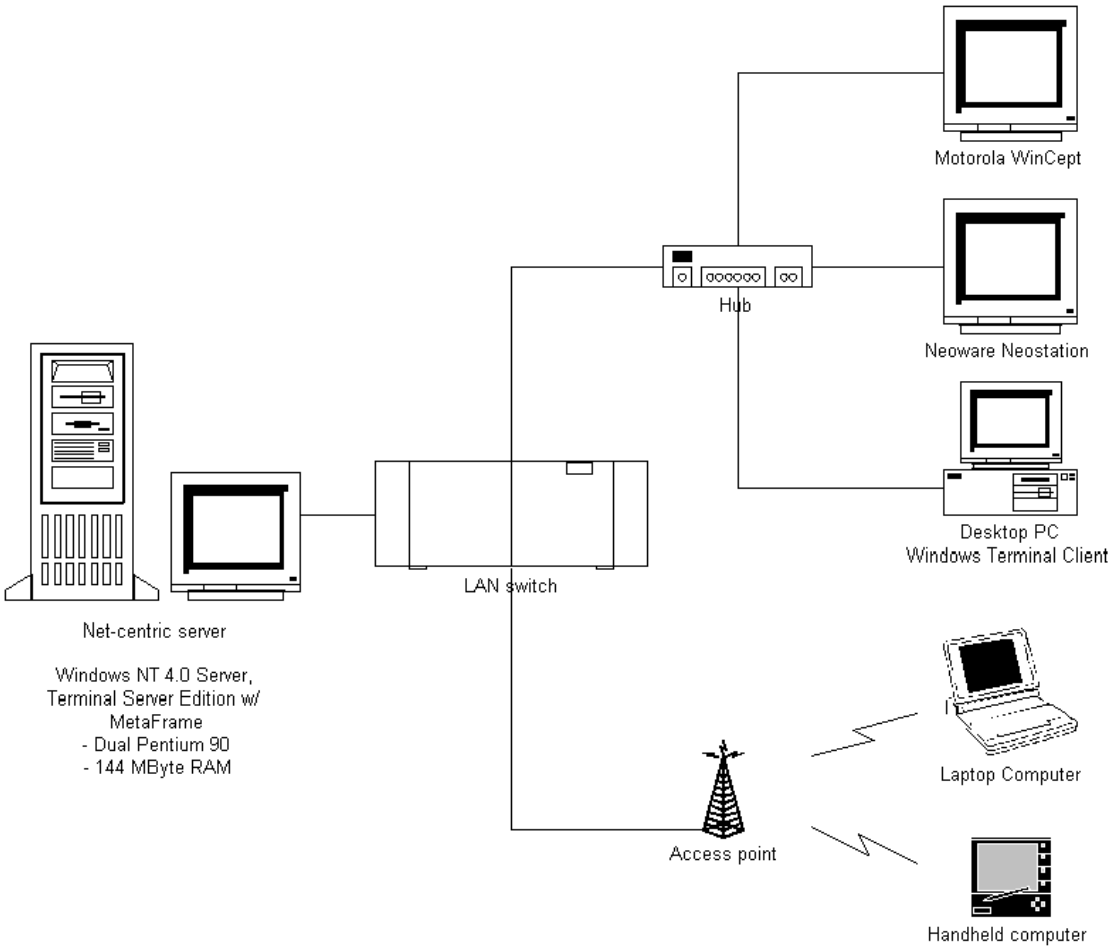

Fig. 2: The configuration of net-centric computing

The configuration of net-centric computing is similar as the traditional client/server computing. There are also clients and net-centric servers. Client machine can be a PC or workstation, or a new type called "thin client"[2]. They are "thin" because they are often very minimally configured. Most of them have no disk drives at all because they

do no long-term local storage. Thin Client refers to the (very small) size of the client operating system. Little or no computation will be carried out in the client side. One characteristic of thin client is its cheaper cost compared with a PC. The server bears a prominent role in net-centric computing as centralized application and client management in it. The server mainly influences the performance of the system.

As the server's computing power will alter the system's performance, a dual-process NT server is used instead of one processor. It provides more capacity in performing computations. The server used is Microsoft® Windows NT™ Server 4.0, Terminal Server Edition. The middleware software used inside the server is Citrix MetaFrame™[3]. It is thin-client/server system software for Microsoft Windows NT Server 4.0, Terminal Server Edition. It extends Windows Terminal Server with additional client and server functionality - including support for heterogeneous computing environments, enterprise-scale management and seamless desktop integration, hence, different platform of operating system or devices can make a connection to the server with less effort.

Besides, in net-centric computing, client devices, whether "flat" or "thin", have instant access to the applications via the server without application rewrites or download. Client performs processing associated with a user interface and applications processing that are done on the server side, and the server after performs processing tasks, it sends back the result and paste it on the screen through the network. A highly efficient computing technology that separates the application's logic from its user interface, so only keystrokes, mouse clicks and screen updates travel the network. Bandwidth used for keystrokes, mouse clicks and screen updates in travelling the network is tiny. As a result, we can say that application performance is bandwidth-independent.

Fig. 3: A demo system for the Net-Centric Computing environment

Figure 3 shows the picture of a demo system for the net-centric computing environment. With network computing devices (including PDAs, laptops and sub-laptops), they can connect to the net-centric server using wireless card via wireless access point.

5. Cost/Effective Study:

One characteristic of net-centric computing is that the total cost of ownership is lower. The cost is even lower than the traditional client/server computing. And the maintenance of the whole system is easier than traditional client/server. Besides, it is more effective when upgrading the software or hardware of the system because centralized management in server side.

Comparing the cost of ownership between traditional client/server computing and net-centric computing. Assume the environment consists of 10 clients to be served and the client machines used in net-centric computing are all thin-client. Table 1 shows the approximate number of each hardware components needed in different computing environment.

Component	**Net-centric computing**	**Traditional client/server computing**
Thin-client devices	10	—
Middleware (MetaFrame™)	1 (with 10-user license) (server)	—
Number of processors	1 dual (server)	10 (client) + 1 (server)
Hard disk	1 (@12G) (server)	@2G (10 clients + 1 server)
Memory	1 (144MB Ram) (server)	@32MB Ram (10 clients + 1 server)
Disk drives	1 (server)	10 (client) + 1 (server)
Monitor, keyboard, mouse	10 (client) + 1 (server)	10 (client) + 1 (server)

Table 1: Number of hardware components needed in net-centric computing environment and traditional client/server computing environment

Hardware like monitor, keyboard, mouse are essential components in both computing environment. From table 1, it is crystal clear that in net-centric computing, most components are used in server machine while in traditional client/server computing, most components are used in client machine. When the system needs upgrading, it is simple enough to upgrade one machine instead of a bundle of machines.

The total cost of ownership is lower in net-centric computing than in traditional client/server computing. It provides a competitive advantage for company who are using net-centric computing. Figure 4 shows the cost comparison between net-centric computing and traditional client/server computing when there are 10- 30- and 50 clients. The cost on hardware components such as monitor, keyboard and mouse are not included in the associated cost due to its common usage for both computing environments. There is one assumption that the client machines used in net-centric computing are all thin-clients, with no PCs using windows terminal client.

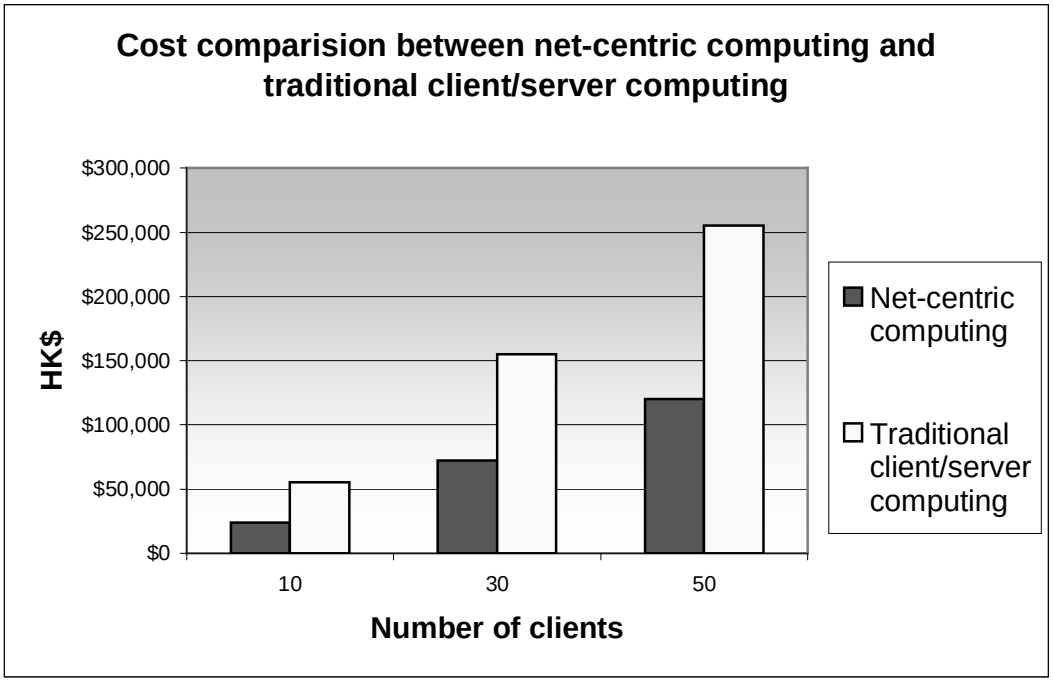

Fig. 4: Cost comparison between net-centric computing and traditional client/server computing[4]

The thin-client costs less than the client machine in traditional client/server computing. From figure 4, it can deduce that when the number of client increases, the expense on buying new client machines in traditional client/server computing is a huge number than using net-centric computing.

Another scenario of upgrading new hardware such as replacing a high speed processor with a slow one or adding more memory storage, the amount of cost will greatly vary between different environment. Figure 5 shows the difference of cost used in upgrading new hardware in different computing environment.

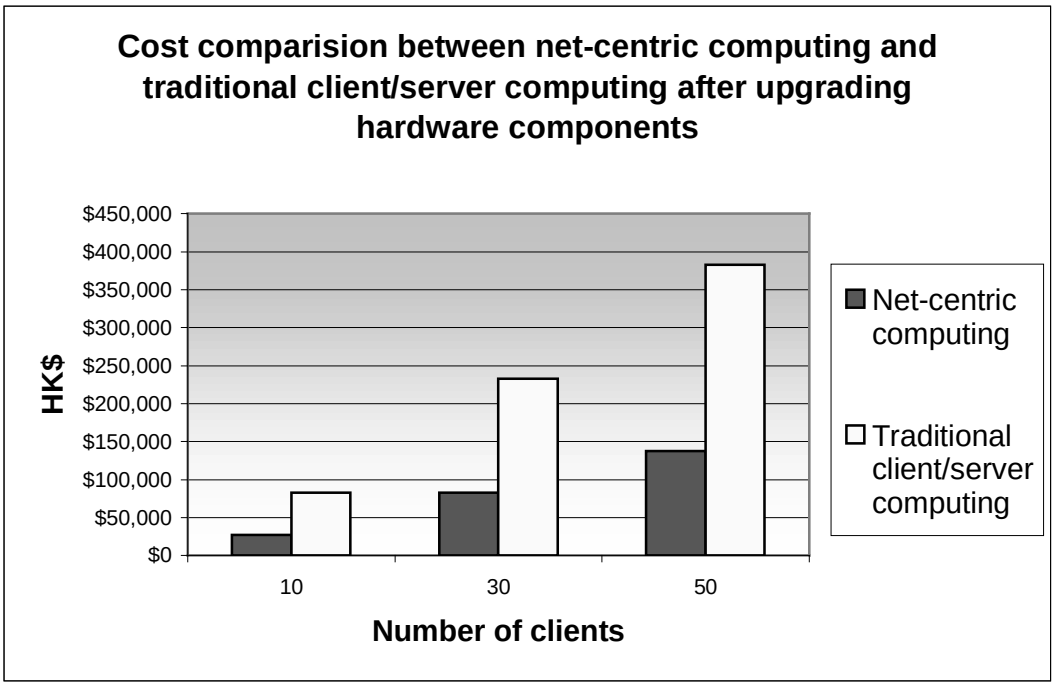

Fig. 5: Cost comparison between net-centric computing and traditional client/server computing after upgrading hardware components[4]

Consequently, net-centric computing is a very cost effective solution on building client/server architecture in enterprise. First, it is a fraction of the cost of traditional client/server computing. Besides, it minimizes network bandwidth consumption by sending only keystrokes, mouse clicks and screen updates to the client. With more clients need to be served, buying one or more servers can fulfill the requirement. In a

situation when the computing environment switches from traditional client/server computing to net-centric computing, the cost used up only in buying a powerful server. It can save expenses from the financial point of view.

6. Performance Study:

Performance [5] is a major factor in determining the overall productivity of a system, performance is primarily tied to availability, throughput and response time. A measurement on the performance of our net-centric architecture is studied.

6.1. Assumptions

In the real net-centric computing environment, a server with a larger number of processors, faster processors, and more memory is more able to support either larger numbers of users or users with greater performance demands. However, the server we used is a dual process 90MHz with a 144Mbytes RAM, it is not powerful enough to generate a general picture on the performance of net-centric computing environment. So, the server is assumed to be powerful to serve several clients in this study.

6.2. Evaluation results

The evaluation studies based on the processor utilization and memory consumption of the system. Figure 6 shows the processor utilization in logging on and off to the server. The first oscillating part is the client logging into the system and the second oscillating part is the server logging out the system. Performance monitor is used to collect the data. It can see that the logging process induces large processor utilization. Afterwards, the processor becomes idle when client becomes idle after log-in process.

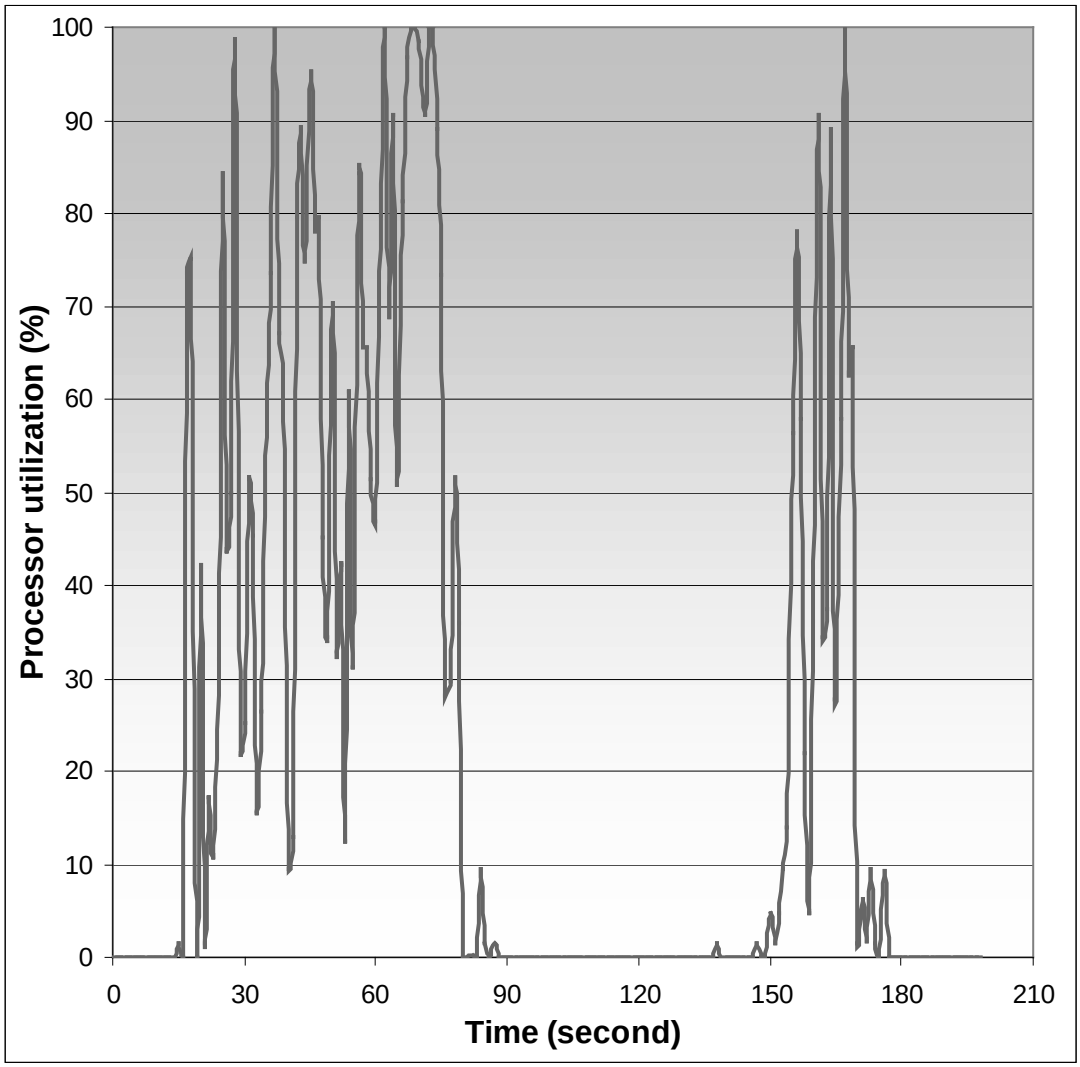

Fig. 6: Processor utilization on logging on and off to the system

The memory used in each client is about 8Mbytes to 10Mbytes. It depends on the user's demand. If the user's activity is large, then it will consume more memory, otherwise, it will consume less memory. The performance is influenced by the factors of number of active users and the intensity of their work.

There is a way to enhance the performance by using load balancing services provided by Citrix. It is a management add-on to Citrix MetaFrame server that allows

administrators to group multiple MetaFrame servers into scalable "server farms" to deliver the best application performance and server resource utilization.

Consequently, the evaluation is limited to several factors: it is a specific performance to the specific configuration and the processor of server using is not powerful. A general analytic model on net-centric computing architecture will be studied in the future and the measure on throughput and response time on a powerful demo system will be examined.

7. Reliability Study:

The reliability of a system is defined as the probability that the system remains operational by time t given that it was started in perfect condition at time 0 [6-7]. In mathematical definition, let R(t) be the reliability of a system, T denote the time of first failure and F(t) its distribution function. Then

$$R(t) = P(T > t) = 1 - F(t).$$

7.1. The model

In modeling the reliability of net-centric computing, the modeling environment is in figure 7. Several clients are connected to the same hub. The hub connects to the LAN switch which it connects to the server. The system is said to be failure if one of the three situations occur: 1). Network failure; 2). Hardware failure; 3). Software failure. Network failure is composed of client network failure or server network failure. Hardware failure may be processor failure, memory failure or hard disk drive failure. In the environment, suppose there are 3 software applications that resided in the server side. Software failure means the application may failure due to faults occurring and it can withstand for most 2 applications to fail.

Fault tree model is used to model the reliability of net-centric computing. Fault trees [8] represent all sequences of failures leading to system failure in a tree-like fashion. The root of the tree represents the event that the entire system has failed, and its *i*th child represents the event that the *i*th subsystem has failed. Each subsystem can be further decomposed into smaller subsystems, thereby giving an arbitrary tree structure. The label on a node indicates how the failure of the children affects the node. Figure 8 shows the fault tree model of net-centric computing.

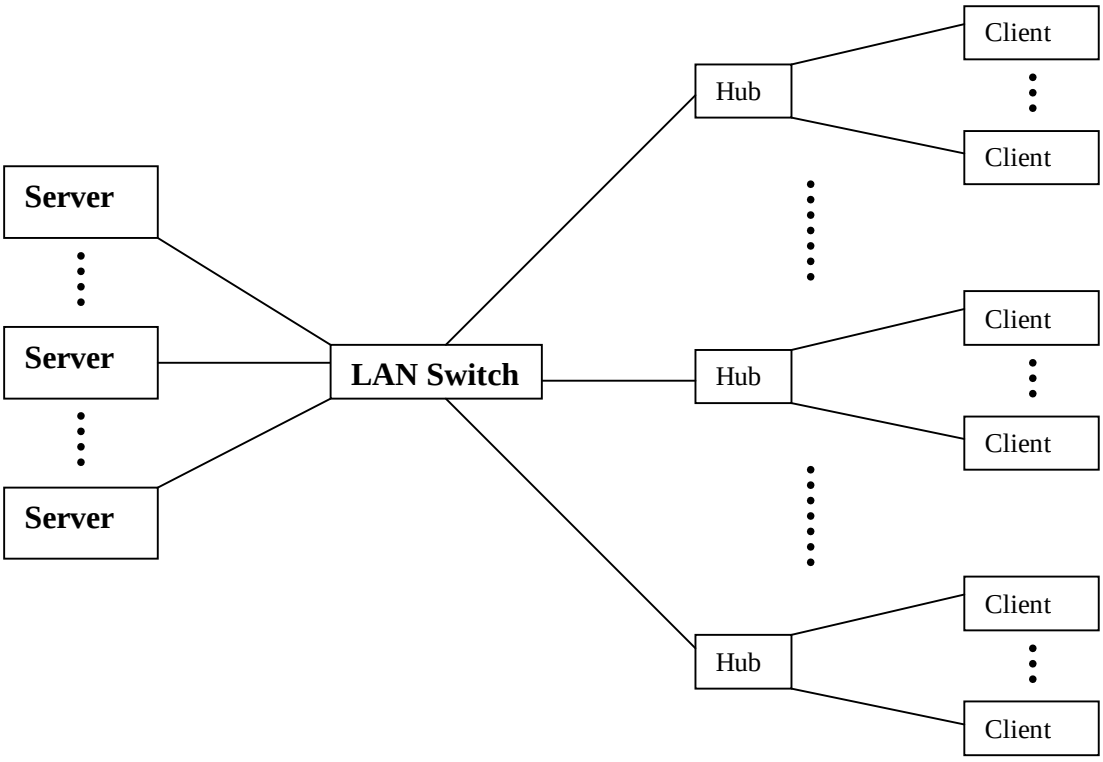

Fig. 7: General model of net-centric computing

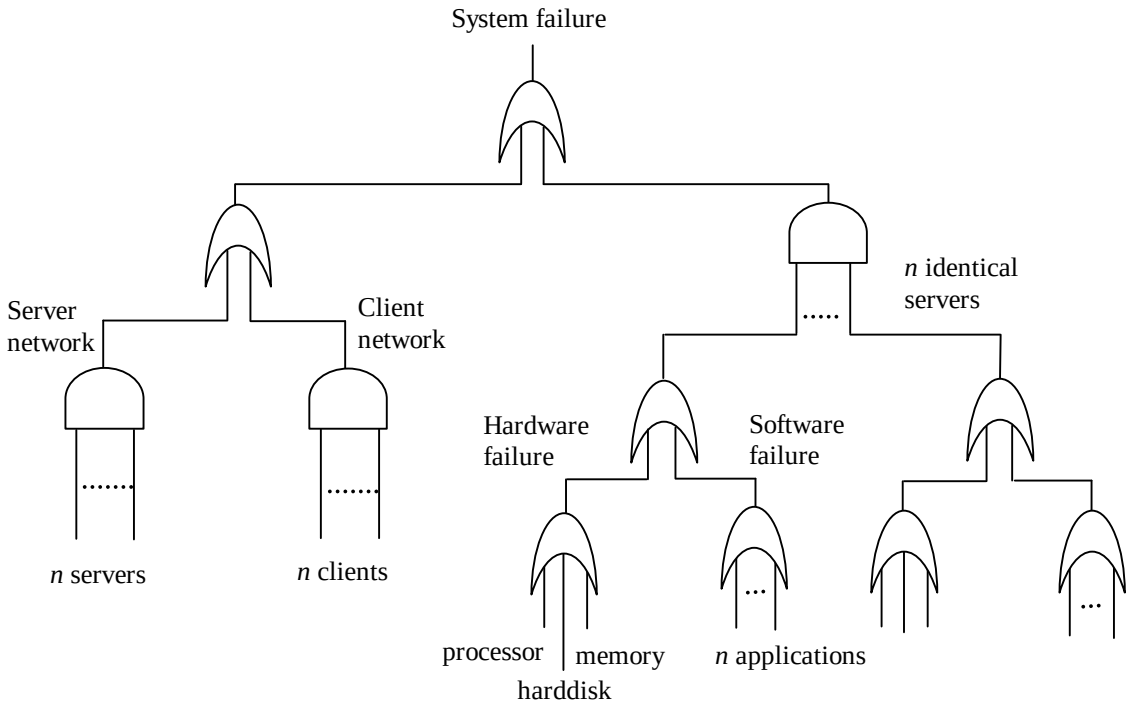

Fig. 8: Reliability model on net-centric computing using fault tree model

7.2. Assumptions

In the reliability model, components are 2-state: either good or failed. Assume there are 10 clients and 1 server in the net-centric environment. The basic components which can fail are the networks (client network and server network), hardware components (processor, memory and hard disk drive) and software components (assume 3 independent applications with different failure rate) in the server side. Links can be unreliable except the link connected with client and the hub is assumed to be reliable. The basic components are assumed to be identically independent. That is when one component fails, it will not affect other component fail. Besides, Exponential distribution is used in modeling the failure rate of the basic components.

To evaluate the reliability of the model, Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE) [9-10] package is used instead of using a discrete-event simulation method. There are several benefits in using SHARPE. It is convenience to make use of SHARPE because it supports multiple model types and provides flexible mechanisms for combining results so that models can be used in hierarchical combinations. It allows its users to construct and analyze performance, reliability, availability and performability models. It gives users direct and complete access to the models without making any assumptions about an application domain.

7.3. Analysis

Using SHARPE package, it can obtain the evaluation result quickly. Below are several scenarios on the reliability of our model. Figure 9 is the reliability of basic reliability model, or called a control model. It is used to compare the result after changing the value of parameters.

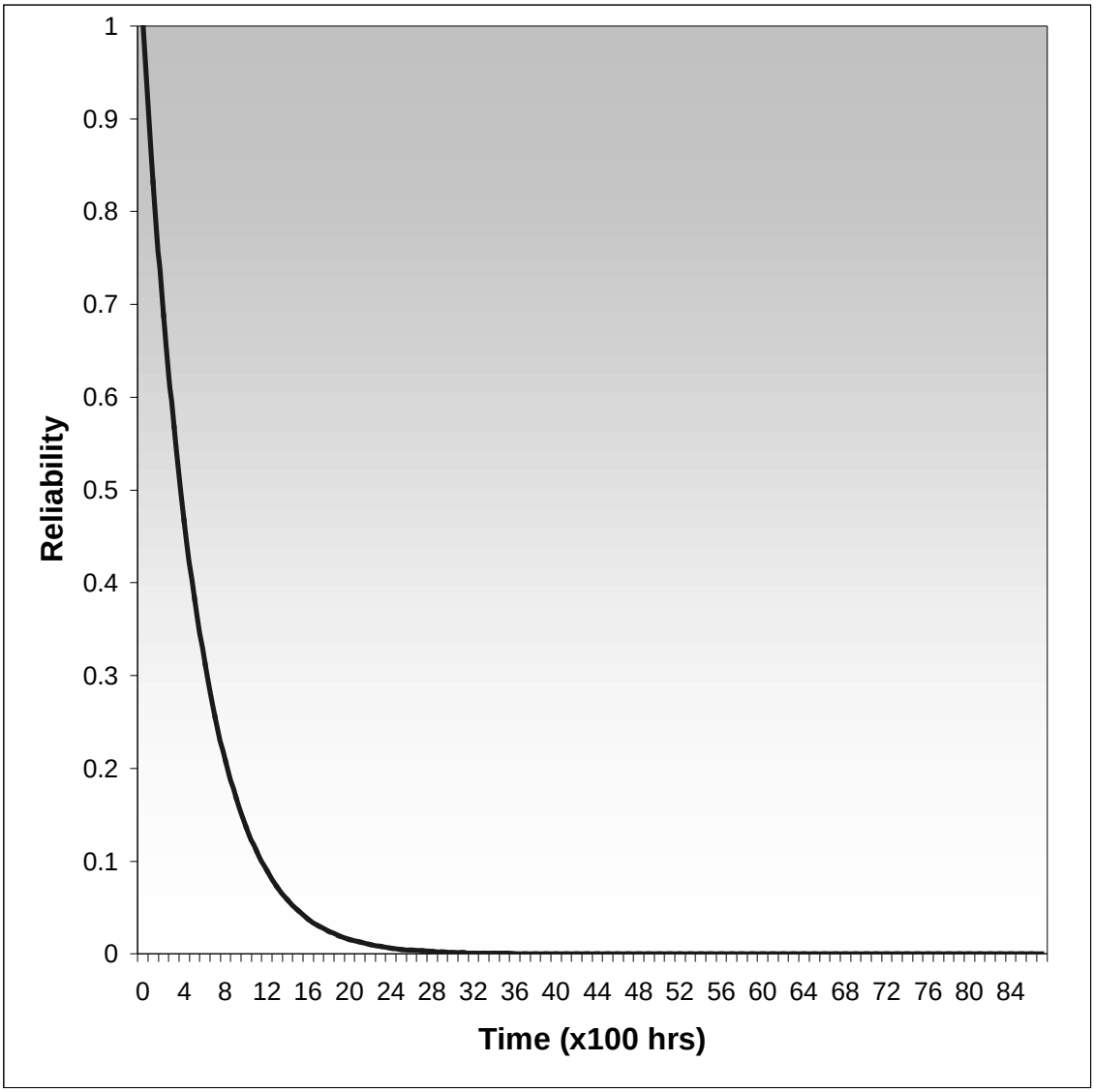

Fig. 9: Reliability of the basic model

After evaluating the reliability of the basic model, we decrease the value of the failure rate on the hardware components, networks and software components one by one and plot its reliability versus the basic model in figure 10.

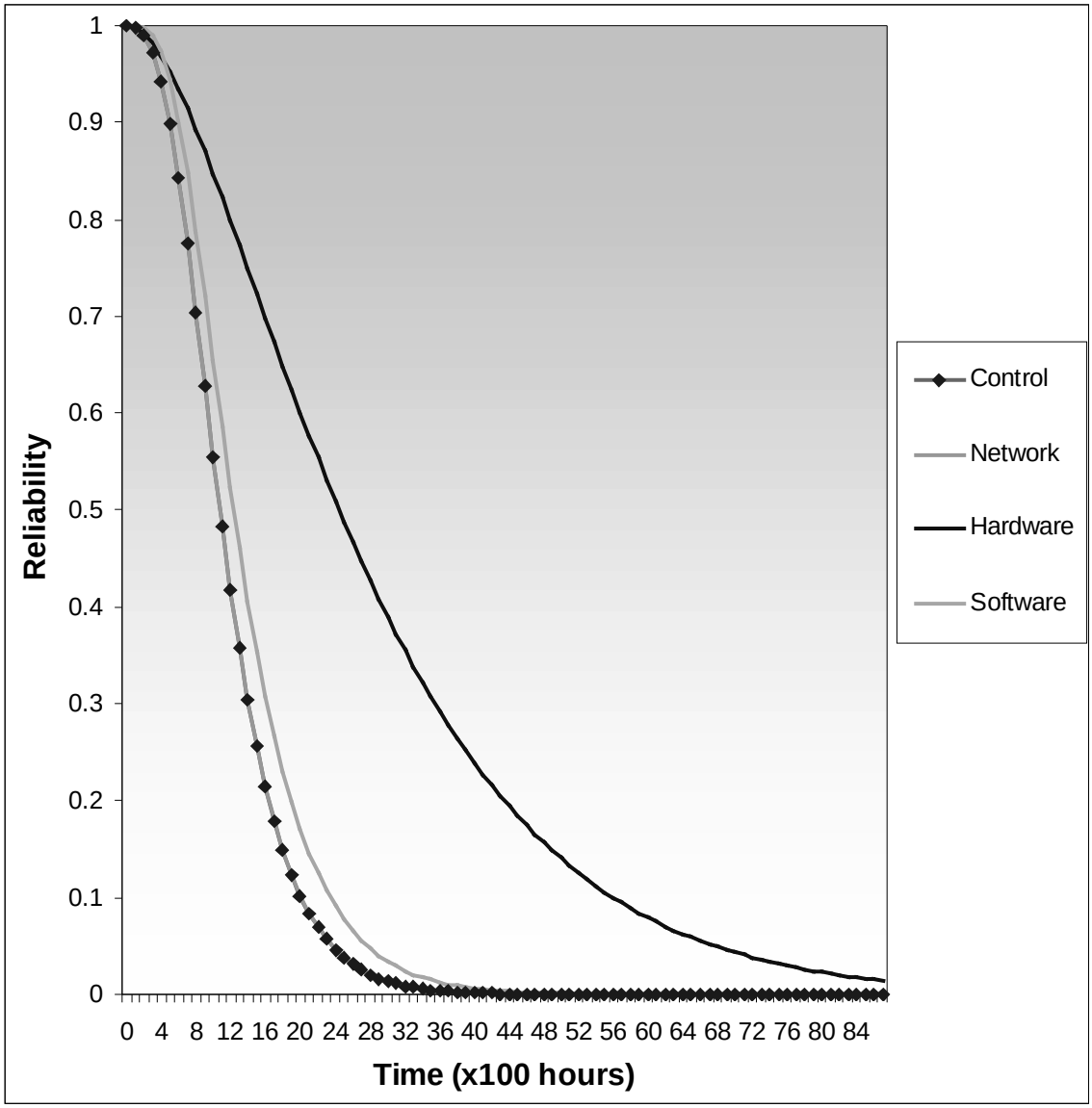

Fig. 10: Reliability using lower failure rate on different fail modes (hardware, software, network)

In the above figure, it can deduce that the failure rate of network influences the reliability of the whole system lightly. However, the failure rate of hardware alters the system greatly. The reason is that if the hardware components fail to operate, the server will fail immediately.

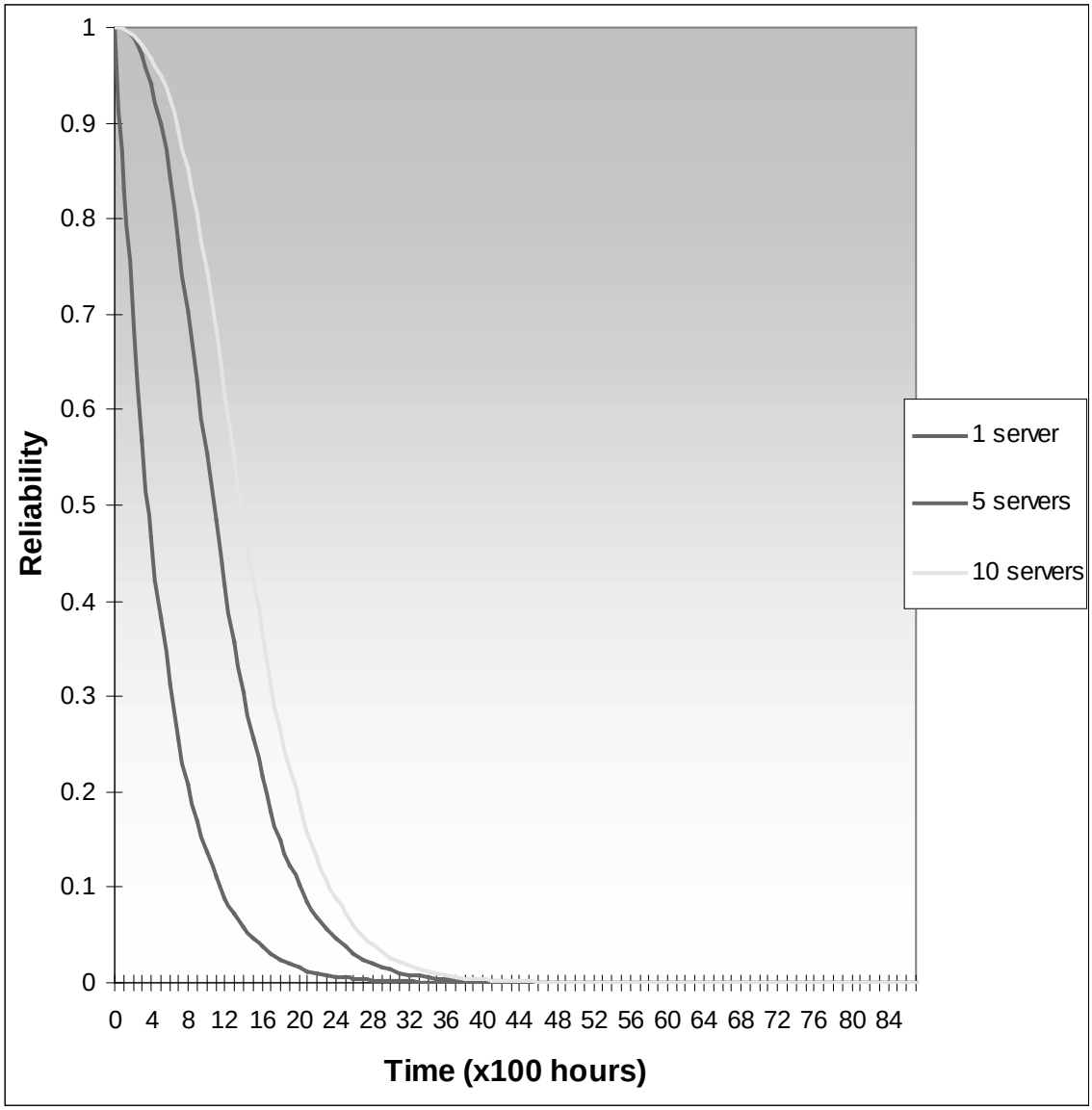

Fig. 11: Comparison of reliability on a fault tolerant system (using 4 and 9 redundant servers) and a non fault tolerant system (I server only)

In the basic model, one server supports ten clients. There is no redundancy on the server. In other words, there is no fault tolerant in the basic model. Figure 11 shows the comparison of reliability when fault tolerant technique is used. There are 4 redundant servers and 9 redundant servers examined in this example. From figure 11, fault tolerant technique provides a better reliability to the system. The operating time is longer when there are redundant servers. However, redundant components will increase

the cost of ownership. As a result, there is a tradeoff between the cost and number of redundant components.

7.4. Comparison of traditional client/server computing and net-centric computing

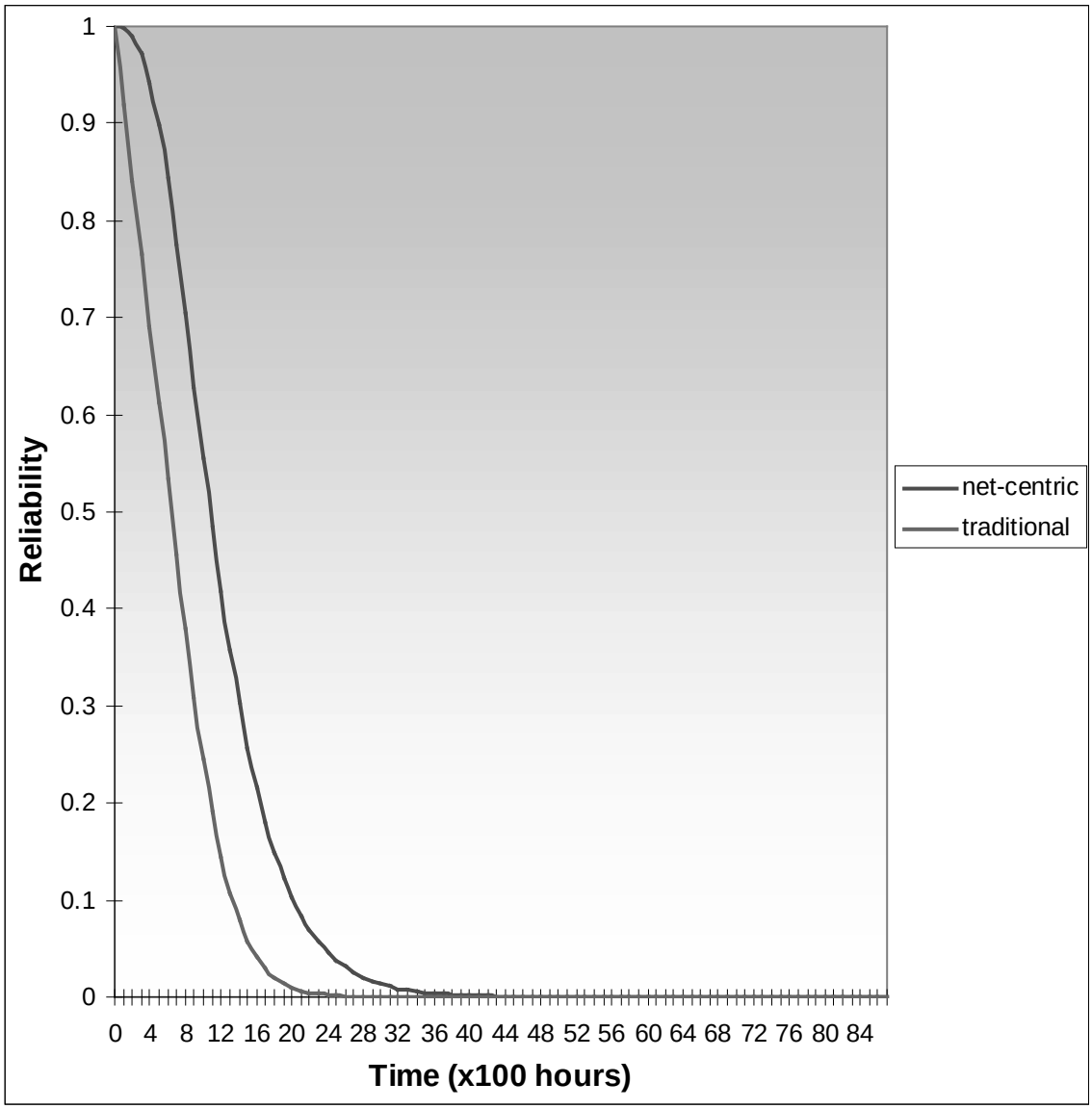

Fig. 12: Reliability of traditional client/server computing model and net-centric computing model

Based on using the same failure rate on the components, net-centric computing has a better reliability from figure 12. In addition, it is cost effective and it provides benefits. Its single-point management, bandwidth independent performance and physically secure together made the net-centric computing powerful.

In this reliability model, fault trees are used. However, they cannot easily represent non-independent behavior of components and are not easily generalized to incorporate performance considerations. Other model such as Markov chain model [11] can be used in determining non-independent behavior of components. It provides great flexibility for modeling reliability, performance, and combined reliability and performance (performability) measures. But they are not always intuitive and the size of their state space grows much faster than the number of system components, making model specification and analysis difficult.

8. Fault Tolerance Study:

8.1. Fault

What is fault? Fault can be simply defined as a condition existing in a hardware or software module that may lead to the failure of the module. It can further characterized into 2 types of fault: hardware fault and software fault [12-14].

Fault-tolerant is the use of protective redundancy to permit continued correct operation of a system after the occurrence of specified fault. The uses of replicated servers continue the services supplied to clients when the server becomes unavailable. This ensures the application can continue running without affecting client's operation.

There are increasing demands to make the application software systems we build today more tolerant to faults. Real time applications such as computer system used in airport, it cannot sustain any fault that found in the system. A result of a big accident will occur due to the faults. Therefore, fault-tolerant is an important issue addressed in the real time application.

8.2. Reusable software components

There are several approaches in regard to enhance the system fault tolerance. One useful fault-tolerant technique which uses reusable software components (watchd, libft, REPL) [15] that can be linked into an application and that they are as efficient as some of the operating system based methods. Those reusable software components are libraries which are efficient, reliable and portable across many platforms. A technique called checkpointing is applied. It is a technique for saving process state during normal execution and restoring the saved state after a failure to reduce the amount of lost work [16-17]. This can be used in addition to the previous two software fault-tolerant techniques, N-version programming and recovery block scheme.

8.2.1. Watchd

It is a watch dog daemon process that runs on a single machine or on a network of machines. It continually watches the life of a local application process by periodically sending a null signal to the process and checking the return value to detect whether that process is alive or dead.

When it detects that the application process crashed or hung, watchd recovers that application at an initial internal state or at the last checkpointed state.

Watchd can also watches one neighboring watchd (left or right) when uses on a network of machines. When a node failure is detected, watchd can execute user-defined recovery commands and reconfigure the network. In addition, it can also watch itself. A self-recovery mechanism is built into watchd in such a way that it can recover itself from an unexpected software failure.

It also facilitates restarting a failed process, restoring the saved values being checkpointed before and reexecuting the logged events and provides facilities for remote execution, remote copy, distributed election, and status report production.

8.2.2. Libft

It is a user-level library of C functions that can be used in application programs to specify and checkpoint critical data, recover the checkpointed data, log events, locate and reconnect to a server, do exception handling, do N-version programming, and use recovery block techniques.

Libft does not require a new language, a new preprocessor or complex declarations and computations to save data structures. It can be ported to several platforms without any changes to the operating system.

8.2.3. REPL

The third reusable component is REPL. REPL file replication technology provides facilities for on-line replication of user specified files on a backup node.

An obvious advantage is reusable. Those software components can be portable to other platforms.

8.3. N-version programming

N-version programming (NVP) [18] is a technique, using N-independent programs execute in parallel on identical input, and results are obtained by voting upon the outputs from individual programs. Different algorithms, techniques, programming languages, environments, and tools must be used in each effort to ensure the development of independent program versions. Figure 13 shows the architecture of N-version programming.

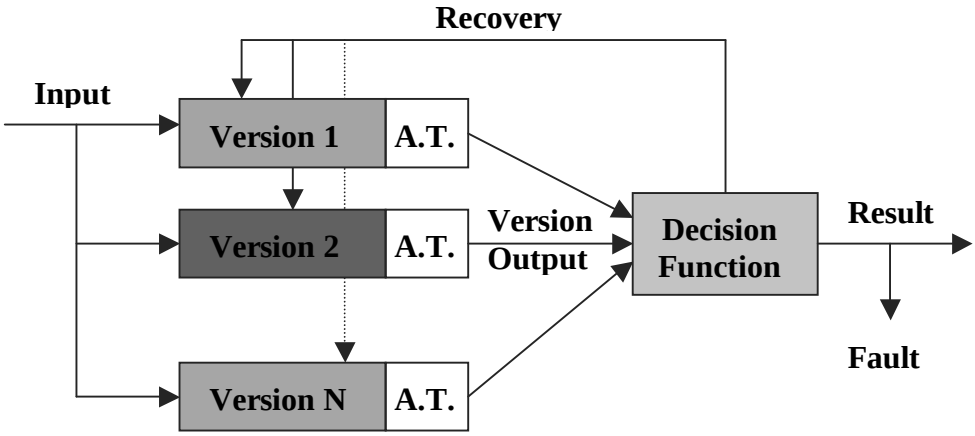

Fig. 13: N-version programming software fault tolerant architecture

The main objectives of the NVP process is to minimize the probability that two or more member versions will produce similar erroneous results that coincide in time for a voting action. However, this technique requires voting at program's end which may not be acceptable when applying to critical systems with real-time deadlines. One important point is that it cannot carry out voting in an immediate point.

8.4. Recovery block scheme

Another fault-tolerant technique is recover block scheme [19]. Recovery block scheme is composed of three elements:

- a primary module;
- an acceptance test and
- a set of alternate modules.

The primary module is to execute critical software functions. The acceptance test which tests the primary module's output after each execution and a set of alternate modules that perform the same function as the primary module and the architecture is shown in figure 14.

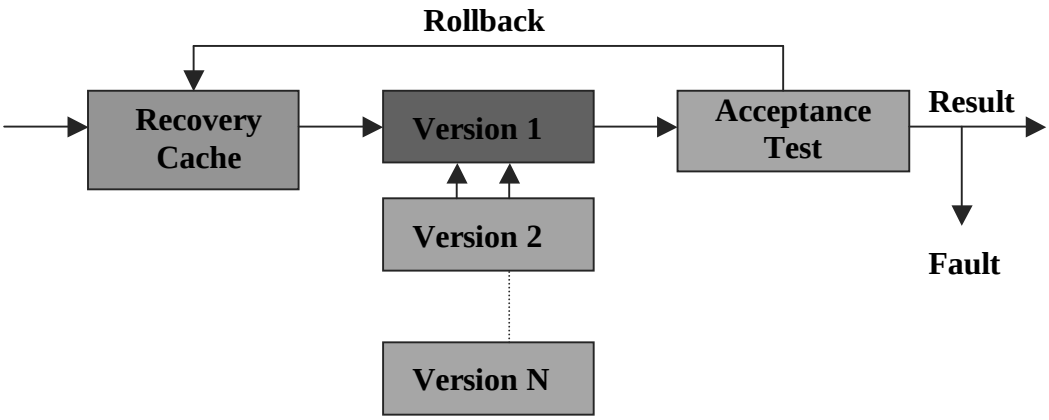

Fig. 14: Recovery block scheme software fault tolerant architecture

If all alternate modules fail, the system can assume to be crashed. One characteristic of this scheme is the modules are executed sequentially.

Another approach is the server "farm". In simple, that is using redundant components. It is a group of servers that are linked together as a 'single system image' to provide centralized administration and horizontal scalability. It shares the workload

among other servers and provides fault tolerant effect. It provides a location transparency to the user as well as failure transparency.

In net-centric computing, fault tolerance is also a significant issue. The centralized processing and database in the server may become unavailable due to the system failure. As a result, fault tolerant technique is a way to enhance the reliability of the system. In other words, it increases the system operation times by providing redundant components before the first fault turns up.

9. Onyoiny research:

The reliability model in this term uses fault tree model which they cannot easily represent non-independent behavior of components and are not easily generalized to incorporate performance considerations. Therefore, Markov chain model and stochastic petric net model will use in the ongoing to model the net-centric computing environment instead of using fault tree model. They provide great flexibility for modeling reliability, performance, and combined reliability and performance (performability) measures.

Besides, the Markov chain model is used to model the performance of the net-centric computing environment. It represents the non-independent behavior of components.

The net-centric architecture consists a distributed component application programming interface (API) and Common Object Request Broker Architecture (CORBA) will be used. It is an object-oriented component model for distributed and heterogeneous components. Research on the CORBA architecture and building common distributed component.

Apart from the modeling of reliability and performance of net-centric computing, software fault tolerant will be investigated. Although the fault tolerant in Windows NT server is good enough, finding another way to enhance the system will be studied.

10. Conclusions:

In this report, net-centric computing environment is examined. Several studies on the cost/effective study, performance study, reliability study and fault tolerance study on the net-centric computing are carried out. The architecture provides a easy-to-understand picture to conclude the net-centric computing environment. A demo system of net-centric computing is used to investigate those studies.

To conclude with those studies, it can deduce that using the net-centric computing is better than using traditional client/server computing. The net-centric computing is more cost-effective and the reliability study states that its reliability is better than the traditional client/server study. The benefits such as maintenance of the whole system and upgrading of hardware and software will be much easier to manage. The administrative job on the system is lightened.

References:

[1] http://www.proxim.com/learn/whiteppr/whatwlan.shtml, *What is a Wireless LAN?*, Proxim white paper

[2] http://www.sbsginc.com/citrix/demo/idea/thin.htm, *CITRIX Big On Thin: Thin-Client/Server Story*

[3] http://www.citrix.com/products/metaframe.asp, *MetaFrame*

[4] Hardware cost approximation from *PC Weekly No. 33*

[5] K. Kant, *Introduction to Computer System Performance Evaluation*, McGraw-Hill, Inc., 1992.

[6] Kishor S. Trivedi, Steve Hunter, Sachin Garg, Ricardo Fricks, "Reliability Analysis Techniques Explored Through a Communication Network Example", *CACC Technical Reports*, 1996

[7] Kishor S. Trivedi, Boudewijn R. Haverkort, Andy Rindos and Varsha Mainkar, "Techniques and Tools for Reliability and Performance Evaluation: Problems and Perspectives", *Computer performance evaluation : modelling techniques and tools, 7th international conference, Vienna, Austria, May 3-6, 1994 : proceedings*, edited by Gunter Haring, Gabriele Kotsis

[8] R. G. Bennetts, "On the Analysis of Fault Trees", *IEEE Transactions on Reliability*, vol. R-24, No.3, Aug. 1975, pp175-185

[9] Robin A. Sahner, Kishor S. Trivedi, Antonio Puliafito, *Performance and Reliability Analysis of computer systems: An Example-Based Approach Using the SHARPE Software Package*, Kluwer Academic Publishers, 1996.

[10] R. Sahner, K. S. Trivedi, "Reliability modeling using SHARPE", *IEEE Transaction on Reliability*, vol. R-36, 1987 Jun, pp186-193.

[11] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, Prentice-Hall, 1982.

[12] Victor P. Nelson and Bill D. Carroll, Tutorial: Fault-Tolerant Computing, *IEEE Computer Society Press*, Ch.1, pages 1-4.

[13] Flavin Cristian, "Understanding Fault-Tolerant Distributed Systems", *Communications of the ACM*, Vol. 34, No. 2, pages 56-78, February 1991.

[14] Victor P. Nelson, "Fault-Tolerant Computing: Fundamental Concepts", *IEEE Computer Society*, July 1990, pp19-25

[15] Yennun Huang and Chandra Kintala, "Software Fault Tolerance in the Application Layer", *Software Fault Tolerance*, edited by Michael R.Lyu,

Ch.10, pp231-248

[16] Yi-Min Wang, Yennun Huang, Kiem-Phong Vo, Pi-Yu Chung and Chandra Kintala, "Checkpointing and Its Applications", *In Proc. of 25th International Symposium on Fault-Tolerant Computing (FTCS-25)*, pp22-31, June 1995.

[17] Y. M. Wang, P. Y. Chung, Y. Huang, E. N. Elnozahy, "Integrating Checkpointing with Transaction Processing", *In Proc. of 27th International Symposium on Fault-Tolerant Computing (FTCS-27)*, pp304-308, June 1997.

[18] Algirdas A. Avizienis, "The Methodology of N-Version Programming", *Software Fault Tolerance*, edited by Michael Lyu, Ch. 2, pages 23-46

[19] Brian Randell and Jie Xu, "The Evolution of the Recovery Block Concept", *Software Fault Tolerance*, edited by Michael R.Lyu, Ch.1, pages 1-21