Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 36 (2009) 27–39
computers & operations research
www.elsevier.com/locate/cor

Disaster avoidance mechanism for content-delivering service

Chih-Chin Liang[a,*], Chia-Hung Wang[c], Hsing Luh[c], Ping-Yu Hsu[b]

[a] *Department of Accounting and Information System, Shih Chien University, Taiwan*
[b] *Department of Management, National Central University, Taiwan*
[c] *Department of Mathematical Sciences, National Chengchi University, Taiwan*

Available online 6 September 2007

Abstract

Many software applications have been developed on client-server and web-based architectures, with client software installed in numerous clerks' devices spread over the whole of Taiwan along the hierarchical organization of large-scaled service companies. The damage for a large-scaled service company would be caused by an inconsistent system, in which client software fails to receive updated content causing loss of emergency calls and possibility of recovery. Therefore, a reliable content-delivering service is needed for such a large-scaled service company to ensure system consistency whenever a disaster occurs. This study analyses the error situations of a proposed content-delivering service that has been implemented in a large-scaled company in Taiwan. Moreover, its disaster avoidance mechanism provides a reliability model with the waiting-time distribution of disconnected links. The proposed disaster avoidance mechanism prevents damage from terminating operations of a company caused by disasters, such as earthquakes, tsunamis, nuclear plant explosions or wars. Empirical results demonstrate that the proposed design run on 19 servers distributes content with less than 3.4 content-delivery failures per million hours.

Keywords: Disaster avoidance; DRP for information systems; Fault tolerance; Content delivery

1. Introduction

Inconsistent systems can be caused by natural disasters, such as earthquakes and tsunamis, or human disasters such as nuclear plant explosions or wars [1,2]. For content-delivering services, these disasters can cause computers to crash and disconnect communication of enterprises, thus terminating their operations [3–5]. Restated, disasters could prevent further activity, including distribution of emergency information, transmission of critical updated contents and continuing services. In terms of cost, if the services of a large-scaled service-oriented company are interrupted, then the company is likely to lose hundreds of millions of US dollars a day in revenue. Therefore, a large-scaled service company needs to employ a specified disaster recovery or disaster avoidance plan to continue content-delivering service to lower the loss caused by a disaster [6–11].

Companies, particularly those in the large-scaled service sector, regard uninterrupted customer service as one of the most important operational goals [12]. A typical large-scaled service company has various electronic devices, such as personal computers (PCs) and personal data agents (PDAs), which are applied by clerks in operation centers to process business transactions. Every device hosts numerous different client software applications, each linking to a system

* Corresponding author.
E-mail addresses: lgcwow@gmail.com (C.-C. Liang), 93751502_@nccu.edu.tw (C.-H. Wang), slu_@nccu.edu.tw (H. Luh), pyhsu_@mgt.ncu.edu.tw (P.-Y. Hsu).

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.09.001

in the back offices. The systems can be devised in client-server and web-based architectures [13,14]. A client-server architecture stores applications and some data, including promotion, area and customer codes, on the client devices. With both architectures, web-based systems can also write data to the devices for the references of client software of other client-server systems [15].

A company requires a feasible approach to sending content along the hierarchy to gain support from all business units [16]. Disaster avoidance is also important to ensure that failed servers on the distribution routes do not bring the distribution to a halt [11]. This study describes the experience of a large company in Taiwan, which originally applied one server to update the contents of all clerks' devices without a disaster avoidance mechanism, and has since learned from the experience that disasters can stop the content-delivering service. Therefore, the company has had to develop a new approach for delivering contents along a physical hierarchy. A content-delivering service called FnFDS (Fire and Forget Distribution System), which distributes the updated contents of the company, has been presented [11].

However, the content-delivering service in FnFDS might stop without a disaster avoidance mechanism. The disaster avoidance mechanism in FnFDS comprises an automatic fault tolerance design and failover mechanism to guarantee that the content-delivering service continues when servers or links fail. This work describes the disaster avoidance mechanism of the FnFDS derived from experienced error statuses, and illustrates the system reliability and the waiting-time distribution of disconnected links by analyzing the disaster avoidance mechanism. Additionally, a system fails if either no server is in operational mode, or all links are disconnected. A server that is functioning correctly and is reachable is said to be in operational mode. Different disasters cause different levels of damage, which can be characterized by a utility for measuring reliability, called a six-sigma capable process. The six-sigma capable process is a helpful quality assurance tool adopted in manufacturing industry, and can be used to measure the reliability of a software application [17]. Applying the six-sigma capable process to measure the reliability, the FnFDS tolerates a maximum of 3.4 failures per million hours. Analytical results indicate that 19 servers, with providing the probability of no service, i.e., 1.90735 occurrences per million hours, are adequate to run FnFDS with a six-sigma capable process, where the frequency of providing no service is 3.4 occurrences per million hours [18].

In practice, the company has utilized a system with 40 servers in three layers along the hierarchical organization, with a no-service frequency of 9.0945×10^{-7} occurrences per million hours. Restated, the reliability of FnFDS using 40 servers is better than that of a six-sigma capable process. This approach is clearly more reliable than a six-sigma capable process, but is also very costly. Hence, the company is scheduling a plan for cutting cost based on the result of this study, while using 19 servers.

The rest of this paper is structured as follows. Section 2 describes related works. Section 3 then describes the system architecture of FnFDS and system status when an error occurs. Next, Section 4 presents the disaster avoidance design applied by FnFDS. Section 5 discusses the reliability of the FnFDS. Section 6 presents the implementation of the FnFDS in a company, and the empirical results of the reliability of the FnFDS. Finally, discussions and concluding remarks are presented in Section 7.

2. Literature review

2.1. Content delivery

Content-delivery methods are generally categorized as push-based and pull-based [19,20]. In a push-based approach, a sender actively sends items to recipients. In a pull-based approach, the recipient transmits requests to a sender to retrieve items [21].

However, each approach has limitations for sending content. Push-based methods consume few network resources, but potentially lose content. A pull-based approach can avoid losing messages through repeated requests for retrieving contents, but places a heavy drain on network resources [20]. Network resources are important for a large-scaled company with limited network bandwidth. Therefore, FnFDS applies routing strategies to send content through a modified pull-based method with fewer requests to a sender than other pull-based methods, thus reducing the consumption of network resources. In FnFDS, a recipient only sends requests to a sender to retrieve content whenever the sender actively transmits a notification to inform the receiver that new content is ready for retrieval. Although this method decreases the consumption of network resources, it potentially stops content-delivering services, since a sender sends no active notification to a receiver when its server crashes or the network is disconnected [22–24]. Therefore, the FnFDS needs a fault-tolerant design to prevent failed servers and links from stopping the content-delivery service.

2.2. Fault-tolerant design

Existing fault-tolerant approaches for delivering content can be classified into two designs. The first approach is adding information to the content, such as its size, to verify its accuracy. The other approach is utilizing a notification system to enable senders and receivers to determine the next action [25–30].

These approaches ensure that a destination can continue receiving the entire content. However, the content-delivery service means that every client device in the operational mode must be able to retrieve updated content successfully along the hierarchical organization of a large-scaled company. Therefore, a fault-tolerant design must be properly adopted to continue delivering updated content along the hierarchical organization of a large-scaled company.

3. System description

FnFDS employs a multi-tier architecture comprising of layers of servers which are organized into balanced trees, meaning the depth of each sub-tree of the root is the same [28]. Fig. 1 shows the hierarchical organization of the servers which are organized into a balanced tree. The root server is in the first level, and a server has its parent server and child servers. For instance, Fig. 1 shows that server *A* is the parent server of server *B*, and server *C* is the child server of server *B*. The FnFDS adopts a modified pull-based design to transmit updated contents through servers to reduce network resource consumption. The FnFDS functions as follows. Every server apart from the root server receives updated contents from a parent server. When the parent server fails to transmit updated content to a child server, it notifies another child server in the same hierarchy level to pass the notification of the new updated content to the servers or clients connected to the failed server. If none of the child servers is available, then the parent server notifies all the servers in the next level to retrieve contents directly.

That is, devices retrieve content from servers only when they receive the notification of the need to retrieve content from servers. This modified pull-based method is designed such that at least one server, usually the root server, is in operational mode. If this server is a device that can connect to other servers and function correctly, all client devices would successfully retrieve the updated content from this one. Additionally, since client devices retrieve updated contents from servers, the content-delivering service stops when a communication failure occurs on a server. The potential process and communication failures for content-delivering service include crashes, general omissions of sent contents and disconnection [22–24]. A crash indicates that a server or a client device is malfunctioning, and requires time to recover. A general omission signifies that the server or the client device receives incomplete content. A disconnection means that the network is disconnected between two communicated devices. The possible error statuses of the content-delivering service include “servers are malfunctioning” and “network link is disconnected.” Table 1 presents the error statuses.

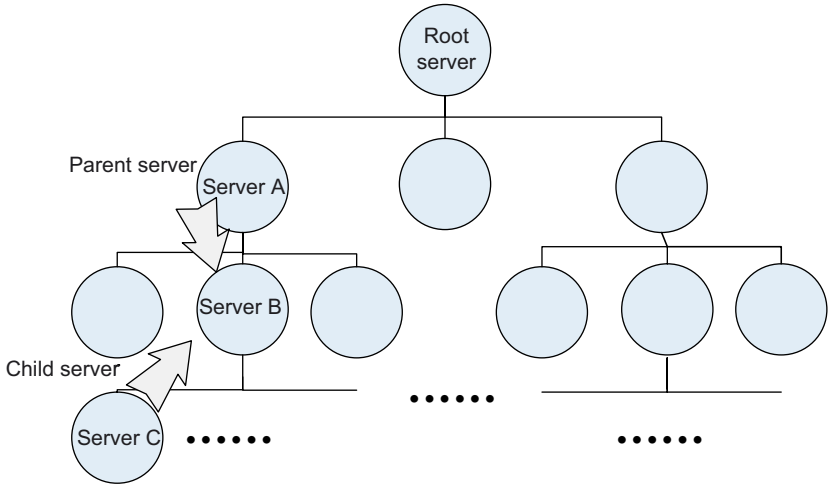

Fig. 1. The hierarchical organization with spanned tree schema.

Table 1
Error statuses

Error type	Server		Link
	Crash	General omission	Disconnection
Reasons	1. Hardware problems, such as malfunctioned network adapter, crash database 2. The server is burned	The server receives incomplete contents owing to transmission error	1. Network equipment is malfunctioned 2. The wire is cut-off 3. The network is congested

The further actions of a malfunctioned server include operator's intervention and automatic recovery. A crash requires intervention by an operator. A general omission can be solved automatically, because recipients retrieve incomplete content later. Partially disconnected cases, including a malfunctioning network router, require intervention by an operator, as well as time to be fixed. Other cases, such as a network disconnection owing to congested traffic, typically recover automatically within a specific period. Each error status has various mean times between failures and times spent for repair. The company assigns specific staff members to repair a malfunctioning server or link. The mean time between failures and the time spent for repair of a server in the system employed by the company are 1094.4 and 6.4 h, respectively. The mean time between failures and the time spent for repair of links are 358.6 and 0.8 h, respectively. The mean times between general omissions and the mean time of restarting a retrieval process are 300 and 1.83 h, respectively.

4. The disaster avoidance design of FnFDS

The FnFDS has specified a disaster avoidance design comprising fault-tolerant design and failover design for the above error statuses. The fault avoidance design is described as follows.

The FnFDS ensures that content can be retrieved by sending notifications and preventing the retrieval of incomplete content by comparing the physical size of delivery content and adding information about content size. The notification is adopted to inform the receiver; the recipient server and the client device retrieve content, or transmit an error when failures occur, allowing the receiver to determine the next action. A server or a client re-retrieves content from any server that has the latest version of the updated content when it recovers from a crash or disconnection. Moreover, FnFDS includes a management console to help the operator to monitor the status of servers and intervene in time when failures occur. A server that is not operating is marked on the management console to prevent it from connecting with clients and other servers. That is, a non-functional server is isolated for repair.

However, a fault-tolerant design has to consider the time sequence of the content-delivering service, as follows: a sending server sends a notification to a recipient server, which then retrieves the content from the sending server. For instance, a server that crashes before transmitting a notification to recipient servers is marked with "error" on the management console to prevent clients from retrieving updated content from it. Meanwhile, the crashed server can be repaired to recover and provide services. Tables 2 and 3 show the details of the fault tolerance design applied to FnFDS to deliver content among servers.

The failover design of the FnFDS is described as follows. Whenever a server marked on the management console as failed needs to be repaired by an operator, other servers are designated to be in charge of sending content to all recipient servers of the marked server. Furthermore, once failing to link to the original destination, a server must be designated to link others with the same functions as the original destination. Additionally, a failed link or a malfunctioned server has a specified repair team to fix it once it is marked on the management console.

However, the server hosts at headquarters controlled by the content-delivering operator require a specific failover plan. The company prepares backup servers of the server hosts at the headquarters. These backup servers can replace failed servers immediately. Every server on the second layer can act as a server host at the headquarters to handle the crash situation. If the server hosts at the headquarters are disconnected from other servers, then the operator can deliver the updated content to one of the server hosts on the second layer to deploy it. If the server hosts at the headquarters are disconnected from other servers, then the operator travels to any site of the second layer as quickly as possible to allow the updated content to be delivered.

Table 2
Fault tolerance design (FTD) for server

	Crash						General omission
Situation	Before sending notification to recipient servers	After sending notification to a recipient server and before the recipient server receives the content	After recipient servers receives content	After receiving notification from the recipient server	Before receiving notification from the sending server	Before getting content from the sending server	Server retrieves incomplete content
FTD	The crashed server is marked as "error" on the management console, and waits for repair. This design prevents client devices from retrieving the wrong version of updated content from the crashed server. The other servers on the same layer as the crashed server would send notification to the recipient servers of the crashed server	The recipient server retrieves no content from the crashed server. The recipient server retrieves the updated content from any other server with the latest version of updated content	No fault-tolerant design is needed since the content has been delivered. The crashed server is marked as "error" on the management console, and waits for repair	Following a fixed time interval, if a recipient server still retrieves nothing from the sending server, the sending server sends a notification: "A server XXX retrieved no content," to the other recipient servers, which notify the recipient servers of the crashed server to retrieve lost contents from them			If updated content is not completely transferred because of general omission, then the content is repeatedly re-retrieved until the transfer is successful

Table 3
Fault tolerance design (FTD) for link

	Disconnection		
Situation	Before the notification of the updated content is received	After the notification of an updated content is delivered, and before the updated content is retrieved	After the updated content is delivered
FTD	If the sending server fails to resend notification to the recipient server 10 times, then it marks the recipient server as "error" on the management console and waits for repair. This design can prevent clients from retrieving the incorrect version of an update from the crashed server. The sending server sends the notification, "The server has retrieved no content," to the other recipient servers to ensure that the recipient servers of the disconnected server receive content from the other servers that retrieve contents successfully	If the recipient server fails to retrieve updated content from the disconnected sending server, then it retrieves updated content from any other server with the latest version of updated content	No fault-tolerant design is necessary, since the content has been delivered

With the above disaster avoidance design, the FnFDS ensures that a client retrieves updated contents whenever it is notified from a server in the same business region. A client device that is disconnected from the server in the same business region can still be notified that the new content is ready for retrieval from any other servers. Additionally, a client device that retrieves no updated content from a crashed server within a business region retrieves the updated content from another server.

Finally, the two following theorems can be obtained from the above design.

Theorem 1. *The content-delivering service can be continued whenever a server is in the error status.*

Proof. In the design described in Table 2, if a parent server crashes, the other parent servers on the same layer would send notification to the child servers of the crashed server. Hence, the child servers can receive notification and retrieve updated contents from other servers, but the clients would retrieve no updated content from the crashed one. Additionally, if the recipient servers are crashed, then they would be marked as a crashed state on the management console throughout other servers. A server that receives incomplete content retrieves it again from the server that is transmitting the content. When a server is disconnected from other servers, other servers can provide content-delivering service to the servers and client devices that were originally connected to the disconnected server. Servers with errors are marked on the management console to prevent other servers and clients from connecting to them. A server with an error resumes providing content-delivering services as soon as it becomes operable again, enabling it to send the latest updated contents. □

Theorem 2. *If at least the root server is in the operational mode, client devices can retrieve updated contents successfully.*

Proof. In this fault-tolerant and failover design, every server except for the root server receives updated contents from a parent server. A parent server that fails to send updated contents to a child server notifies another child server in the same hierarchy level to pass the notification of the new update contents to the servers or clients connected to the failed server. If no child server is available, then the parent server notifies all servers in the next level to retrieve contents directly.

A client retrieves updated contents after receiving the notification, and can retrieve updated contents from any other server in operational mode, mainly in the same business region. If a client device is disconnected from the server in the same business region, then the client device can still be notified from any other servers. Additionally, a client device that retrieves no updated content from a crashed server within a business region retrieves the updated content from another server.

Therefore, a client device can always receive updated contents as long as the root server is in operational mode. □

Through the above design, the FnFDS can provide a disaster avoidance content-delivering service.

5. Reliability

To understand the reliability of FnFDS, this study obtains the FnFDS queueing model and analyzes it [31–34]. This investigation assumes $s \geqslant 1$ servers in the system for sending downward messages, which have independent identically distributed (i.i.d.) exponential lifetimes. According to the architecture of the proposed method, FnFDS has x links in the system to connect servers, where $x \geqslant 1$. Additionally, the lifetimes of these links may be interdependent.

5.1. Measure of system reliability

This investigation calculates the probability that no server is in operational mode, using the following procedure. The notations $1/\alpha_c$ and $1/\alpha_G$ denote the mean time from regular operation to the crash state, and to the general omission state for each server, respectively. This study assumes that the times taken by each server to reach the crash state and the general omission state follow i.i.d. exponential distributions. A crashed server needs a mean repair time of $1/\beta_c$. Moreover, repairing a failed server with general omission takes a mean repair time $1/\beta_G$. These two repair times are assumed to be i.i.d. random variables with exponential distributions.

The state transition rate diagram (Fig. 2) is a Markovian system with period 2, signifying that future behavior is a function only of the present, and not of the past. This study indicates the steady state probability as follows:

$$\pi = (\pi_0, \pi_C, \pi_G),$$

where π_0, π_C, and π_G denote limiting probabilities of the regular operation, crash, and general omission states. The interpretation of π_j is the probability of finding the Markov process in state j after the process has been in operation for

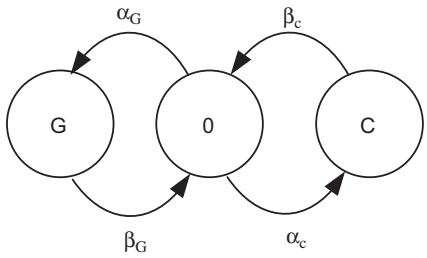

Fig. 2. The state transition rate diagram.

a long duration. In addition, the interpretation of π_j could be described as the long run mean transitions of time that the process spends in state j [35]. A transition probability matrix is derived,

$$T = \begin{bmatrix} 0 & p_G & p_C \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

where $p_G = \alpha_G/(\alpha_G + \alpha_C)$ and $p_C = \alpha_C/(\alpha_G + \alpha_C)$. The transition probability matrix leads to the following two transition probability matrices:

$$T_{even} = T^{(2n)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & p_G & p_C \\ 0 & p_G & p_C \end{bmatrix} \quad \text{for } n = 1, 2, \ldots$$

and

$$T_{odd} = T^{(2n-1)} = \begin{bmatrix} 0 & p_G & p_C \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \text{for } n = 1, 2, \ldots .$$

Using T_{even} and T_{odd}, the stationary probabilities indicate:

$$\pi_0 = \tfrac{1}{2}, \tag{1}$$

$$\pi_G = \frac{\alpha_G}{2(\alpha_G + \alpha_C)}, \tag{2}$$

$$\pi_C = \frac{\alpha_C}{2(\alpha_G + \alpha_C)}. \tag{3}$$

With modern technology, the operation time of a server should far exceed the down time. To make a conservative calculation, we then assume that the servers spend the same amount of time in each of the three states after each transition. π_0, π_C, and π_G therefore denote the average time spent in each corresponding state in the worst case. Under this assumption, FnFDS has the following limiting probability of no server in operational mode in the system:

$$\Pr\{\text{no server in operational mode}\} = (\pi_C + \pi_G)^s = \left(\frac{1}{2}\right)^s. \tag{4}$$

Additionally, this study computes the limiting probability that all links are disconnected. The procedure for finding the limiting probability that links are disconnected is described as follows. The term $1/\alpha_d$ denotes the mean time from the connected state to the disconnected state for each link. Repairing a disconnected link has a mean repair time of $1/\beta_d$. This investigation assumes that lifetime and repair time follow exponential distributions, with r repair teams existing for repairing disconnected links.

At time t, $N_{x,r}(t)$ indicates the number of disconnected links in this Content-Delivering Service system, which comprises x links and r repair teams. The repair teams exist for fixing disconnected links in time. This investigation assumes that $x \geqslant r \geqslant 1$. This study considers $\{N_{x,r}(t), t \geqslant 0\}$ as a birth and death process on state space $\{0, 1, \ldots, x\}$

with birth parameter α_d and death parameter β_d. The equation $P_n(x,r)=\lim_{t\to\infty}\Pr\{N_{x,r}(t)=n\}$ denotes the limiting probability that n disconnected links exist in a system with x links and r repair teams. This investigation obtains the limiting probability by formulating it as a $M/M/r$ queueing system with finite population x. The limiting distribution is calculated from the following theorems (see Kleinrock [36]):

Theorem 3. *Let $N_{x,r}(t)$ represent the number of disconnected links at time t, in a system that contains x links and r repair teams. If*

$$P_n(x,r)=\lim_{t\to\infty}\Pr\{N_{x,r}(t)=n\},\quad 0\leqslant n\leqslant x,$$

then,

$$P_n(x,r)=\begin{cases}\dfrac{x!}{(x-n)!n!}\left(\dfrac{\alpha_d}{\beta_d}\right)^n P_0(x,r) & \text{for } n=1,\ldots,r-1,\\ \dfrac{x!}{(x-n)!r!r^{n-r}}\left(\dfrac{\alpha_d}{\beta_d}\right)^n P_0(x,r) & \text{for } n=r,\ldots,x,\end{cases}\tag{5}$$

where,

$$P_0(x,r)=\left[\sum_{n=0}^{r-1}\frac{x!}{(x-n)!n!}\left(\frac{\alpha_d}{\beta_d}\right)^n+\sum_{n=r}^{x}\frac{x!}{(x-n)!r!r^{n-r}}\left(\frac{\alpha_d}{\beta_d}\right)^n\right]^{-1}.\tag{6}$$

The term $L(x,r)$ represents the expected number of disconnected links in the system of x links and r repair teams, which are in the steady state. Using the limiting probability described in Theorem 3, the expected value is determined as

$$L(x,r)=\sum_{n=0}^{x}nP_n(x,r).\tag{7}$$

Little's law indicates that the average number of customers in a queueing system equals the arrival rate of customers reaching that system multiplied by the average time spent in it [37]. This study adopts Little's Law to find relations among the expected number of disconnected links, expected repair time of each disconnected link and disconnection rate.

Theorem 4. *Let $W(x,r)$ represent the expected time taken to repair a disconnected link in a system with x links and r repair teams in steady state, where α_d indicates the disconnection rate of a link. This value is computed as follows*:

$$W(x,r)=\frac{L(x,r)}{\alpha_d(x-L(x,r))}.\tag{8}$$

Proof. The expected time, $W(x,r)$, is derived using Little's Law with the effective disconnection rate $\sum_{n=0}^{x}(x-n)\alpha_d P_n(x,r)$. □

This investigation calculates the probability that all links are disconnected from the result of Theorem 3.

$$\Pr\{\text{all links are disconnected}\}=P_x(x,r)=\frac{x!}{r!r^{x-r}}\left(\frac{\alpha_d}{\beta_d}\right)^x\left[\sum_{n=0}^{r-1}\frac{x!}{(x-n)!n!}\left(\frac{\alpha_d}{\beta_d}\right)^n+\sum_{n=r}^{x}\frac{x!}{(x-n)!r!r^{n-r}}\left(\frac{\alpha_d}{\beta_d}\right)^n\right]^{-1}.\tag{9}$$

The term $P_x(x, r)$ denotes the period of time when all links are disconnected in the long run. The probability that a system is down is determined from the results of (4) and (9), as follows:

$$\begin{aligned}\Pr\{\text{system is down}\} &= \Pr\{\text{no server in operational mode}\} + \Pr\{\text{all links are disconnected}\}\\ &= \left(\frac{1}{2}\right)^s + \frac{x!}{r!r^{x-r}}\left(\frac{\alpha_d}{\beta_d}\right)^x \left[\sum_{n=0}^{r-1}\frac{x!}{(x-n)!n!}\left(\frac{\alpha_d}{\beta_d}\right)^n \right.\\ &\quad \left. + \sum_{n=r}^{x}\frac{x!}{(x-n)!r!r^{n-r}}\left(\frac{\alpha_d}{\beta_d}\right)^n\right]^{-1}. \end{aligned} \tag{10}$$

Eq. (10) obtains the probability that the content-delivering service system will be down through assigned numbers of servers and links s and x, respectively. The result of Eq. (10) could be used to compare with a six-sigma capable process. Restated, if the result shows that the number of occurrences of the status 'system is down', providing no service, is below 3.4 occurrences per million unit time, then the system is better than a six-sigma capable process. Additionally, the time that a disconnected link waits for repair is unknown. To understand these issues, the waiting-time measurement is discussed in Section 5.2.

5.2. Waiting-time distributions of repairing disconnected links

A disconnected link takes time to begin to be fixed by a repair team. The probability distribution for the waiting-time of the repair team to fix disconnected links must be calculated. Every disconnected link has an individual waiting-time. This investigation assumes that the queue discipline is first-in-first-out. The waiting-time is a random variable that is partially discrete and partially continuous. The waiting-time is a continuous random variable, except when the waiting-time is zero, because there is a non-zero probability that there is no delay. Restated, there is a non-zero probability that a link is repaired as soon as it is disconnected.

Let T denote the time spent waiting in the queue for the repair team. The term $F_T(t)$ represents the cumulative probability distribution of the time spent waiting for repair. The equation $T = 0$ indicates no waiting-time, which means that a disconnected link can be repaired (enter service) immediately. This case arises only when the number of disconnected links in the system, n, is strictly below the number of repair teams r. Therefore, $\Pr\{T=0\}=\sum_{n=0}^{r-1} P_n(x, r)$. The term $P_n(x, r)$ represents the limiting probability that there are n disconnected links when there are x links and r repair teams. This work demonstrates that

$$\begin{aligned}F_T(0) &= \Pr\{T \leqslant 0\} = \Pr\{T = 0\} = \sum_{n=0}^{r-1} P_n(x, r)\\ &= x!\left(\frac{(\alpha_d + \beta_d/\beta_d)^x}{\Gamma(1+x)} - \frac{(\alpha_d/\beta_d)^r\, {}_2F_1(1, r-x; 1+r; -(\alpha_d/\beta_d))}{\Gamma(1+r)\Gamma(1-r+x)}\right) P_0(x, r),\end{aligned} \tag{11}$$

where $\Gamma(y) = \int_0^\infty z^{y-1}\, e^{-z}\, dz$ represents the gamma function, and

$${}_2F_1(a, b; p; q) = \frac{\Gamma(p)}{\Gamma(b)\Gamma(p-b)}\int_0^1 z^{b-1}(1-z)^{p-b-1}(1-qz)^{-a}\, dz, \tag{12}$$

indicates the hypergeometric function. The value of $F_T(t)$ then needs to be found for $t > 0$.

If the system has n disconnected links when a disconnection arises, then all n disconnected links must have been repaired by time t for the repair of the disconnected link to begin at a time between 0 and t. Since the service distribution is exponential and thus has no memory, the distribution of the time required for n completions is independent of the time of the current disconnection, and is the convolution of n exponential random variables. As noted by Kleinrock [36], the sum of n i.i.d. random variables with an exponential distribution, has the Erlang-n distribution, and this applies to the time for n repair completions. Additionally, since the arrival distribution of disconnection events follows a Poisson distribution, the arrival points under a fixed numbers of events are uniformly distributed [35]. Therefore, the probability that an arrival encounters n in the system is identical to the stationary distribution of system size. Hence, the time

distribution of waiting for repair is derived

$$
\begin{aligned}
F_T(t) &= \Pr\{T \leqslant t\} = F_T(0) \\
&\quad + \sum_{n=r}^{x} [P_n(x,r) \cdot \Pr\{(n-r+1) \text{ completions by time } t | \text{arrival found } n \text{ in system}\}] \\
&= \sum_{n=0}^{r-1} P_n(x,r) + \sum_{n=r}^{x} \left[\frac{x!}{(x-n)!r!r^{n-r}} \left(\frac{\alpha_d}{\beta_d}\right)^n P_0(x,r) \cdot \int_0^t \frac{\beta_d(\beta_d z)^{n-r}}{(n-r)!} \mathrm{e}^{-\beta_d z} \, \mathrm{d}z \right] \\
&= \sum_{n=0}^{r-1} P_n(x,r) + P_0(x,r) \sum_{n=r}^{x} \left[\int_0^t \frac{\beta_d(\beta_d z)^{n-r}}{(n-r)!} \mathrm{e}^{-\beta_d z} \frac{x!}{(x-n)!r!r^{n-r}} \left(\frac{\alpha_d}{\beta_d}\right)^n \mathrm{d}z \right] \\
&= P_0(x,r)x! \left(\frac{(\alpha_d + \beta_d/\beta_d)^x}{\Gamma(1+x)} - \frac{(\alpha_d/\beta_d)^r {}_2F_1(1, r-x; 1+r; -(\alpha_d/\beta_d))}{\Gamma(1+r)\Gamma(1-r+x)} \right) \\
&\quad + P_0(x,r) \int_0^t \beta_d \mathrm{e}^{-\beta_d z} \sum_{n=r}^{x} \left[\frac{x!}{(x-n)!r!r^{n-r}} \left(\frac{\alpha_d}{\beta_d}\right)^n \frac{(\beta_d z)^{n-r}}{(n-r)!} \right] \mathrm{d}z
\end{aligned}
$$

for $t > 0$.

However, the total time that a link is disconnected comprises various repair processes, including waiting-time and the time spent repairing the disconnected link. This investigation denotes the total time by U, and its cumulative probability distribution by $F_U(t)$. The expression could be presented as follows:

$$
\begin{aligned}
F_U(t) &= \Pr\{U \leqslant t\} \\
&= \sum_{n=r}^{x} [P_n(x,r) \cdot \Pr\{(n-r+2) \text{ completions by time } t | \text{arrival found } n \text{ in system}\}] \\
&= \sum_{n=r}^{x} \left[\frac{x!}{(x-n)!r!r^{n-r}} \left(\frac{\alpha_d}{\beta_d}\right)^n P_0(x,r) \cdot \int_0^t \frac{\beta_d(\beta_d z)^{n-r+1}}{(n-r+1)!} \mathrm{e}^{-\beta_d z} \mathrm{d}z \right] \\
&= P_0(x,r) \sum_{n=r}^{x} \left[\int_0^t \frac{\beta_d(\beta_d z)^{n-r+1}}{(n-r+1)!} \mathrm{e}^{-\beta_d z} \frac{x!}{(x-n)!r!r^{n-r}} \left(\frac{\alpha_d}{\beta_d}\right)^n \mathrm{d}z \right] \\
&= P_0(x,r) \int_0^t \beta_d \mathrm{e}^{-\beta_d z} \sum_{n=r}^{x} \left[\frac{x!}{(x-n)!r!r^{n-r}} \left(\frac{\alpha_d}{\beta_d}\right)^n \frac{(\beta_d z)^{n-r+1}}{(n-r+1)!} \right] \mathrm{d}z
\end{aligned}
$$

for $t > 0$.

6. Implementation

6.1. FnFDS in a large-scaled company

In a large-scaled company in Taiwan, to build the FnFDS to fit with the hierarchical organization, one server is set up at the headquarters in the first layer, while other servers connect to the server in the headquarters in the administrative regional centers in the second layer. When the FnFDS starts, the system checks the availability of contents in the dispatching server of headquarters, and then notifies the dispatching servers in the administrative regional centers to fetch the data from it. After the data arrive at the regional administrative centers, the system then requests dispatching servers in the regional centers to fetch the contents from their corresponding dispatching servers in the regional administrative centers. Additionally, if a client is disconnected from the network when contents are distributed, then it can contact the dispatching server in the corresponding regional administrative center after reconnecting to the network. Fig. 3 shows the transmission schema for the company [11]. The number of disposed servers in the transmission schema is based on the number of layers. The company sends contents through 40 servers spread over three layers. The 40 servers comprise one dispatching server at HQ, three dispatching servers in regional administrative centers and 36 dispatching servers in regional centers. In the company, the Node HQ in the first layer comprises the dispatching server

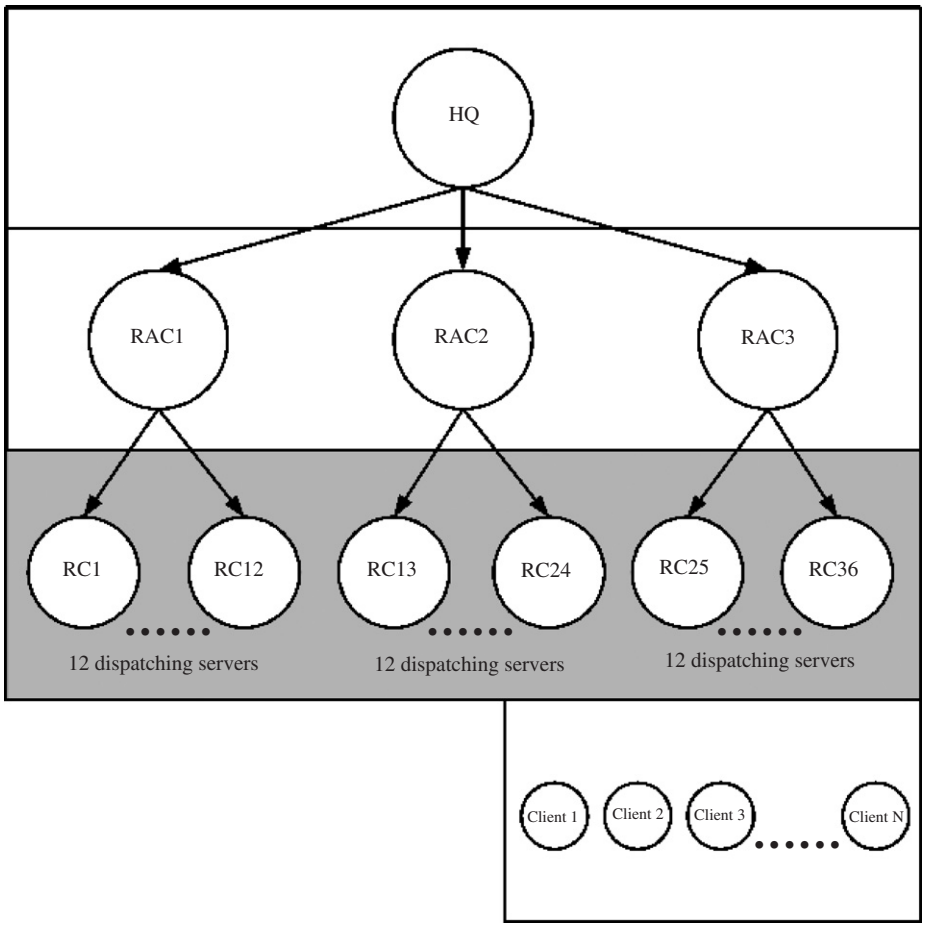

Fig. 3. The 4-tier architecture of FnFDS.

located at headquarters of the organization. RAC1, RAC2 and RAC3 in the second layer include dispatching servers in the three regional administrative centers located in the corresponding Business Groups. All of the other 36 regional centers in the third layer contain dispatching servers. The fourth layer comprises the client devices that are the final destinations of the updated contents [11].

6.2. Results

The reliability is calculated using Eq. (10) with real data and assigned variables. In this case, every malfunctioned situation has a specific staff member to repair it, and every server has links to connect other servers. The mean failure time $(1/\alpha_c)$ and the mean repair time $(1/\beta_c)$ of a server in the system employed by the company are 1094.4 and 6.4 h, respectively. The mean failure time $(1/\alpha_d)$ and the mean repair time $(1/\beta_d)$ of links are 358.6 and 0.8 h, respectively. The mean times between general omissions $(1/\alpha_G)$ and the mean time of restarting a retrieval process $(1/\beta_G)$ are 300 and 1.83 h, respectively.

The six-sigma capable process is originally used in manufactory industry to measure and improve the quality of goods. Recently, the methodology has been utilized in the service industry to gauge and improve service level. According to the standard of six-sigma, the FnFDS cannot be out of service for more than 3.4 times in a period of one million hours. The out of service rate is computed by expression (10), which can be reduced to $(\frac{1}{2})^s$, where s is the number of servers. This is because that the case company has a pre-assigned repair team to maintain each link (each link has its own specified repair team), the number of links is equal to the number of repair teams, the parameter n is equal to

r. In addition, since the number of links is equal to the number of disconnected links in the worse case, the parameter n is equal to x. In this case study, due to each link of being maintained by a pre-assigned repair team, the number of links is equal to the number of repair teams, $x = r$ when all links are broken down, implying the system is down. The out of service occurrences are 3.8149 times per million hours and 1.97035 times per million hours, for $s = 18$ and 19, respectively. Therefore, the ideal configuration should contain 19 servers since its rate is below 3.4 times per million hours.

However, before the analysis is completed, 40 servers have been deployed to serve customers. Obviously, the reliability of adopting 40 servers (9.0945×10^{-7} occurrences times per million hours) is far better than adopting 19 servers. However, the cost of the former approach is a lot higher than the latter one. Therefore, a plan for reducing the cost of providing service is launched currently [17].

7. Conclusions

Content distribution must be considered by service-oriented companies with fast-evolving intranet environments and many client installation bases. Restated, the same version of the updated contents needs to arrive at all on-line client devices before a required deadline to allow companies to upgrade corresponding client software and business processes. Hence, service-oriented companies without a content-delivering service with a disaster avoidance mechanism are likely to lose revenue, customers and even goodwill.

This study proposes a disaster avoidance mechanism designed for an effective content-delivering service implemented in a large-scaled company with multiple layers along a hierarchical organization. To analyze the error statuses and solutions, this study derives a reliability model and waiting-time distribution of disconnected links for such a content-delivering service. However, due to the lack of a utility function associated with the waiting-time of repairing failed links, the acceptable repair time for failed links will be discussed in a further study.

Disasters cause different levels of damage that can be characterized using a reliable six-sigma capable process. This investigation found that a company running at least 19 servers could ensure a working content-delivering service, with 1.90735 service stoppages per million hours. That is, the reliability of FnFDS using 19 servers is better than that of a six-sigma capable process. However, the number of deployed servers has to be along the hierarchical organization with a balanced tree. The practical result demonstrates that the frequency of service stoppages when using 40 servers of three layers, as in the case study, is 9.0945×10^{-7} times per million hours, which is better than the case of only employing 19 servers. This approach is clearly more reliable than a six-sigma capable process, but is also very costly. Hence, the company is scheduling a plan for cutting cost based on the results of this study, while using 19 servers.

References

[1] King D. Post disaster surveys: experience and methodology. Australian Journal of Emergency Management 2002;17(3):1–13.
[2] Pidgeon N, O'Leary M. Man-made disasters: why technology and organizations (sometimes) fail. Safety Science 2000;34:15–30.
[3] Hayes PE, Hammons A. Picking up the pieces: utilizing disaster recovery project management to improve readiness and response time. IEEE Industry Applications Magazine 2002;8(6):27–36.
[4] Bank DR. Telecomm disaster recovery planning for electric utilities. IEEE Conference Rural Electric Power 2005:D3/1–D3/10.
[5] Shao BBM. Optimal redundancy allocation for information technology disaster recovery in the network economy. IEEE Transactions on Dependable and Secure Computing 2005;2(3):262–7.
[6] Fallara P. Disaster recovery planning. IEEE Potentials 2003;22(3):42–4.
[7] Smith DR, Cybrowski WJ, Zawislan F, Amstein D, Dayton AD, Studwell TD. Contigency/Disaster recovery planning for transmission systems of the defense information system network. IEEE Selected Areas in Communications 1994;12(1):13–22.
[8] Hiles A. Surviving a computer disaster. Engineering Management 1992;3(3):271–4.
[9] Stuckenschmidt H, van Harmelen F. Generating and managing metadata for web-based information systems. Knowledge-Based System 2004;17(5–6):201–6.
[10] Taylor MJ, Mcwilliam J, England D, Akomode J. Skills required in developing electronic commerce for small and medium enterprises: case based generalization approach. Electronic Commerce and Research Applications 2004;3(3):253–65.
[11] Fry M, MacLarty G. Policy-based content delivery: an active network approach. Computer Communications 2001;24:241–8.
[12] Wang WM, Liang CC, Lu HZ, Chow WS, Chang KY. Research of testing process: the case of TOPS-system delivery process. TL Technology Journal 2004;34(1):7–34.
[13] Mills RJ, Paper D, Lawless KA, Kulikowich JM. Hypertext navigation—an intrinsic component of the corporate intranet. Journal of Computational Information Systems 2002;43(3):44–50.
[14] Conti M, Gregori E, Lapenna W. Client-side content delivery policies in replicated web services: parallel access versus single server approach. Performance Evaluation 2005;59(23):137–57.

[15] Ranganathan C, Ganpathy S. Key dimensions of business-to-customer web sites. Information & Management 2002;39:457–65.
[16] Jiang Y, Wu MY, Shu W. A hierarchical overlay multicast network. IEEE Conference Multimedia and Expo 2004; 1047–50.
[17] Goh TN. A strategic assessment of six sigma. Quality and Reliability Engineering International 2002;18(5):403–10.
[18] Liang CC, Hsu PY, Leu JD, Luh H. An effective approach for content delivery in an evolving intranet environment—a case study of the largest telecom company in Taiwan. Lecture Notes in Computer Science 2005;3806:740–9.
[19] Hong GY, Goh TN. Six sigma in software quality. TQM Magazine 2003;15(6):364–73.
[20] Pallis G, Vakali A. Insight and perspectives for content delivery networks. Communications of the ACM 2006;49(1):101–6.
[21] Défago X, Schiper A, Urbán P. Total order broadcast and multicast algorithms; taxonomy and survey. ACM Computing Surveys 2004;36(4): 372–421.
[22] Bhide M, Deolasee P, Katkar A, Panchbudhe A, Ramamritham K, Shenoy P. Adaptive push-pull: disseminating dynamic web data. IEEE Transactions on Computer 2002;51(6):652–68.
[23] Fich F, Rparentt E. Hundreds of impossibility results for distributed computing. Distributed Computing 2003;16(2–3):121–63.
[24] Davies DW. An historical study of the beginnings of packet switching. Computer Journal 2001;44(3):152–62.
[25] Androutsellis-Theotokis S, Spinellis D. A survey of peer-to-peer content distribution technologies. ACM Computing Surveys 2004;36(4): 335–71.
[26] Saxena N, Pinotti CM, Das SK. A probabilistic push-pull hybrid scheduling algorithm for asymmetric wireless environment. IEEE Conference GLOBALCOM 2004: 5–9.
[27] Rodeh O, Birman KP, Dolev D. Using AVL trees for fault-tolerant group key management. Journal of Information Security 2002;1:84–99.
[28] Lim H, Kim W, Lee B, Yim C. High-speed IP address lookup using balanced multi-way trees. Computer Communications 2006;29(11): 1927–35.
[29] Liang CC, Wang CH, Luh H, Hsu PY. A robust web-based approach for broadcasting down-ward messages in a large-scaled company. Lecture Notes in Computer Science 2005;4255:222–33.
[30] Morgan G, Shrivastava SK. Implementing flexible object group invocation in networked systems. International Conference on Dependable Systems and Networks 2000:439–48.
[31] Birman KP, Renesse R. Reliable distributed computing with the Isis toolkit. Wiley, IEEE Computer Society Press; 1994.
[32] Wagner LD, Ross JV, Possingham HP. Catastrophe management and inter-reserve distance for marine reserve network. eprint arXiv:math/0309255 2003;2:1–12.
[33] Gunther NJ. The dynamics of performance collapse in large-scaled networks and computers. Journal of High Performance Computing Applications 2000;14(4):367–72.
[34] Hsieh CC, Hsieh YC. Reliability and cost optimization in distributed computing systems. Computers of Operations Research 2003;30(8): 1103–19.
[35] Taylor HM, Karlin S. An introduction to stochastic modeling. 3rd ed, New York: Academic Press; 1998.
[36] Kleinrock L. Queueing systems volume I: theory. New York: Wiley; 1975.
[37] Little JD. A proof of the queueing formula $L = \lambda W$. Operation Research 1961;9:383–7.