Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Circle quorum system-based non-stop network service model

SONG Ping (宋 平)†, SUN Jian-ling (孙建伶), HE Zhi-jun (何志均)

(*Department of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*)

†E-mail: songping_zju@hotmail.com

Received Aug. 18, 2003; revision accepted Nov. 25, 2003

Abstract: Rapid developments in network systems of business service have resulted in more reliance on distributed computing, typified by "subscriber/push" architectures. Unfortunately, frequent and unexpectable network failures were routine, and downtime was not in hours, but in days. High availability has become the most important factor decreasing business risk and improving Quality of Service. Cluster technology has solved the non-stop problem on Local Area Network. However, most technologies including cluster today fail to ensure the non-stop Internet service based on Routers. With good performance on high availability and fault tolerance, quorum systems are very suitable for application to distributed business service networks. In this work, we modeled and developed a non-stop Internet service system based on a new quorum system, circle quorum system, for Boston Mutual Fund Broker, US. With five protocols, it provided highly available data services for clients on Internet.

Key words: Non-stop network, Quorum system, Distributed computation, Fault-tolerance
Document code: A **CLC number:** TP316.4

INTRODUCTION

Non-stop network, a special distributed network with feature of high availability, provides clients valid and consistent data in case of unexpectable failures. Based on the improvement of communication reliability, most business Internet service networks have adopted "Description/Push" architecture as the real-time Internet service framework which clients customize or use to describe their requirements to Internet servers who offer customized information to clients on time. Unfortunately, frequent and unexpectable network failures still happened routinely, and downtime was not in hours, but in days. Moreover, it is a great loss to Internet service companies, and decreases the Quality of Service and increases service costs. Technologies for fault tolerance study became an urgent issue. Fault tolerance technologies generally included RAID theory, backup online, mirror technology and fake IP. It just offered fault tolerant policies for computer hardware, but not for the entire non-stop network system.

Cluster technology proposed some non-stop polices on local area network (Lee *et al*., 1998; Mohan and Parmon, 1998). However, without including Routers, remote mainframes and processes takeover, it cannot solve non-stop problem on Internet. In real application, especially for financial transaction system on Internet, any of the above problems could not be accepted and computer network system was required to have extremely high stability and fault tolerance.

Quorum systems (Martin and Dahlin, 2002; Malkhi and Reiter, 2000; Malkhi, 2000) were recently introduced and studied to deal with the above problems. Generally, a quorum system was a set of sets called quorums; each pair of quorums intersects. All elements in a quorum should have the same features such as data and services consistency.

It had good performance on duplicate data (Ahamad and Ammar, 1980), fault tolerance (Yin *et al*., 2002; Malkhi *et al*., 1999) and load balance (Malkhi *et al*., 2001; Peris *et al*., 2001; Kumar, 2002).

By introducing quorums systems into distributed computation, a novel quorum system, circle quorum system, was designed to realize distributed fault tolerant policy with high availability. Moreover, a non-stop Internet service model with five protocols, duplication protocol, takeover protocol, Router protocol, read protocol and recovery protocol, was constructed that provided clients a highly available data service. The design of this model has successfully been implemented until now in international transaction system at Boston 24×7 Mutual Fund Broker, US.

CIRCLE QUORUM SYSTEM

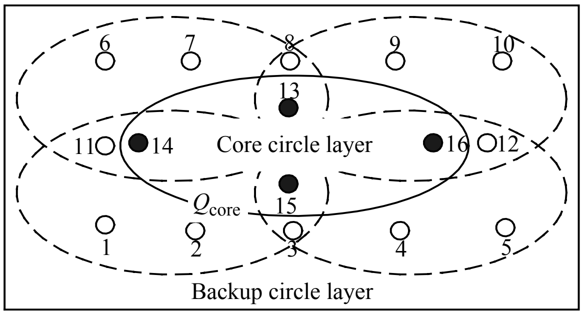
A set system $Q=\{Q_1, \ldots, Q_m\}$ is a collection of subsets $Q_i \subseteq U$ of a finite universe U. A quorum system is a set system Q that has intersection property: $P \cap R \neq \phi$ for all P, $R \in Q$. Alternatively, quorum systems are known as intersecting set systems or as intersecting hyper-graphs. The subsets of the set system are called quorums.

System definition

Traditional quorum systems are typically represented by Byzantine quorum system (Tsuchiya and Kikuno, 2002), Crumbling Wall quorum system (Peleg and Wool, 1997), Grid quorum system (Kumar, 2002), Tree quorum (Kafri and Janecek, 2002) and Diamond quorum system (Fu *et al*., 2002). Their topology covers regular grid, tree, diamond, and irregular grid.

Normally quorum systems simulate ROWA (Read One Write All) protocol for failure resistance, which means valid data are read from only one valid quorum and all data are written to all quorums. This failure tolerant mode requires huge disk capacity and loses big communication bandwidth to backup one data in all nodes of all quorums. Generally, unexpectable Internet failures easily split the whole quorum system into two or more independent and disconnected systems. It would make it impossible to write data to all quorums. In addition, the probability of three quorums' simultaneous failure is very small. Thus, in our design of circle quorum system, every three quorums are required to backup one part of the distributed data. Through two fault tolerant policies, the novel system can realize high availability.

A circle quorum system is defined as $Q=\{Q_1, \ldots, Q_m\}$, $Q_i=Q_{\text{out}}(i) \cup Q_{\text{core}}$, $Q_{\text{out}}(i)$ is a set of backup circle layer, Q_{core} is a set including all nodes of core circle layer, and $Q_{\text{out}}(i) \cap Q_{\text{out}}(i+1) \neq \phi$. Fig.1 shows an example of circle quorum system with 16 nodes.



$Q_{\text{core}}=\{13,14,15,16\}$ $Q_{\text{backup}}=\{1,2,3,4,5,6,7,8,9,10,11,12\}$
$Q_{\text{out}}(1)=\{1,2,3,11\}$ $Q_{\text{out}}(2)=\{3,4,5,12\}$
$Q_{\text{out}}(3)=\{12,8,9,10\}$ $Q_{\text{out}}(4)=\{6,7,9,11\}$
$Q(1)=\{Q_{\text{out}}(1) \cup Q_{\text{core}}\}$ $Q(2)=\{Q_{\text{out}}(2) \cup Q_{\text{core}}\}$
$Q(3)=\{Q_{\text{out}}(3) \cup Q_{\text{core}}\}$ $Q(4)=\{Q_{\text{out}}(4) \cup Q_{\text{core}}\}$

Fig.1 A circle quorum system with 16 nodes

Fault-tolerant policies

Unexpectable faults can happen in any place of the circle quorum system. Two basic policies were designed for nodes' failures of core circle layer and backup circle layer.

(1) Core circle layer: two neighbors' takeover policy. When one node of core circle layer failed, two valid neighbors next to it would take over its data services and generated data are sent to their nodes. In Fig.1, suppose node 15 failed, two valid neighbors node 14 and 16 would take over the data services of node 15 and send corresponding data to nodes, 1, 2, 3 and 11. Nodes, 1, 2, 3 and 11, would be merged into new quorums, $Q(2)$ and $Q(4)$. At the same time, node 14 and node 16 became neighbors, because the connectivity between node 14 and node 16 existed when node 15 failed.

(2) Backup circle layer: a voting policy with the newest timestamp. Only if under the newest timestamp, the summation of data number returned by valid nodes of backup circle layer was more than half of quorum size, the returned data were regarded as valid data. For instance, if node 1 failed and the quorum size was 6, the summation of data numbers of data sent by nodes 2, 3, 11 and 15 was 4. According to voting policy, the value of returned data was valid.

Circle quorum system is a fault tolerant system with high availability. With all connectivity of circle quorum system valid, circle quorum system failed only if all nodes of core circle layer failed.

NON-STOP INTERNET SERVICE MODEL

Logic framework definition

Non-stop Internet service model framework is defined with four layers, computation layer, Router layer, backup layer and client layer, which provide data services for clients on Internet.

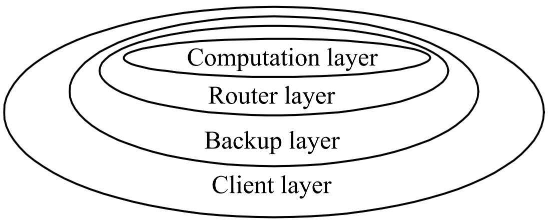


Fig.2 Four layers logic framework

Fig.2 shows the four-layered Non-stop Internet service model framework. Below is the definition:

(1) Computation layer: with strong computation capability, all computers work as mainframes which receive clients' customized information, generate original data and push them to client layer of clients' computers through computers on Router layer and backup layer;

(2) Router layer: with powerful transmission capability, Routers relay group broadcast information to computers on backup layer, screen failed computers between computer layer and backup layer and redirect valid computer address;

(3) Backup layer: computers of this layer have large disk or memory storage, save and rapidly push customized information to related clients who are online;

(4) Client layer: clients send customized information to mainframes and receive their data pushed by computers on backup layer.

With good features of strong data computation, powerful relay capability of group broadcast and rapid memory and transmission capability, Non-stop Internet service model can provide highly available data services for its clients as rapidly as possible.

Non-stop Internet service model definition

Definition 1 Computers in computation layer are defined as mainframes, and computers in backup layer are defined as servers. Routers on Router layer make a valid connection between computation layer and backup layer.

Definition 2 Mainframe set in computation layer is described by Eq.(1),

$$M = \{m_0, m_1, \ldots m_i, \ldots, m_{N-1}\} \tag{1}$$

$\forall i \in [0, N-1]$, mainframe $m_i \in M$, N means the number of mainframes.

Definition 3 Router set in Router layer is described by Eq.(2),

$$R_{\text{Router}} = \{R_0, R_1, \ldots R_i, \ldots, R_{N-1}\} \tag{2}$$

$\forall i \in [0, N-1]$, Router subset $|\mathrm{R}_i \cap \mathrm{R}_{(i+1) \bmod N}| \geq 2$.

Definition 4 Server set in backup layer is described by Eq.(3),

$$S_{\text{Backup}} = \{S_0, S_2, \ldots, S_i, \ldots, S_{N-1}\} \tag{3}$$

$\forall i \in [0, N-1]$, server subset $|S_i \cap S_{(i+1) \bmod N}| \geq 2$.

Definition 5 The format of distributed dataset in unit U_i is described by Eq.(4),

$$(DBL_i, (DCL_i, DCR_i), DBR_i) \tag{4}$$

$\forall i \in [0, N-1]$, DCL_i and DCR_i represent left and right computation dataset; $DBL_i = DCR_{(i-1+N) \bmod N}$ means left backup dataset, and $DBR_i=DCR_{(i+1)\ \mathrm{mod}\ N}$ the right backup dataset.

For quorum i, left backup dataset DBL_i equals right computation dataset $DCR_{(i-1+N)\mathrm{mod}\ N}$ of its left neighbor, quorum $(i-1+N)$ mod N; while right backup dataset DBR_i equals left computation dataset $DCL_{(i+1)\mathrm{mod}\ N}$ of its right neighbor, quorum $(i+1)$ mod N. We use mainframe m_i or server $s_j\rightarrow$dataset to denote mainframe m_i or server s_j operates on distributed dataset, such as $m_i\rightarrow DCL_i$ and $S_j\rightarrow DBR_j$ that indicate the mainframe m_i operates on DCL_i and servers of serve set S_i handle DBR_i.

Definition 6 D=(*timestamp*, *number*, *value*, *type*) is designed as basic data structure.

D(*timestamp*) and D(*number*) are used for clients to vote valid data D(*value*) under unexpected failures situation. Number D(*number*) is the times of the newest timestamp dataset by different quorums. Generally, D(*number*) of valid intersected nodes is set to 2; because their data are affirmed twice by two intersected quorums. For other non-intersected nodes, the number is set to 1. Data type D(*type*) is used for Routers to redirect valid mainframe. D(*type*) can be assigned to (000 & 001), (010 & 011), (100 & 101) and (110 & 111) to represent left backup data (Part one & two), left computation data (Part one & two), right computation data (Part one & two) and right backup data (Part one & two).

Based on the above definitions, the Non-stop Internet service model can be defined by Eqs.(5)−(7):

$$NSISM=\{U_0,U_1,...U_i,...,U_{N-1}\} \quad (5)$$

$$U_i=\{Q_{(i-1+N)\,\mathrm{mod}\,N},Q_i,Q_{(i+1)\,\mathrm{mod}\,N}\} \quad (6)$$

$$Q_i=\{m_i,R_i,S_i\} \quad (7)$$

where $\forall i\in[0,\ N-1]$, $m_i\in M$, $R_i\in R_{\mathrm{Router}}$ and $S_i\in S_{\mathrm{Backup}}$. Quorum Q_i mainly executes dataset (DCL_i, DCR_i) services such as computation, transmission and failure tolerance; Data service unit U_i, including quorum $Q_{(i-1+N)\mathrm{mod}\ N}$, Q_i and $Q_{(i+1)\mathrm{mod}\ N}$, principally monitors the availability of quorum Q_i and prepares to take over dataset (DCL_i, DCR_i) services. When quorum Q_i fails, quorum $Q_{(i-1+N)\mathrm{mod}\ N}$ takes over data DCL_i services and $Q_{(i+1)\mathrm{mod}\ N}$ takes over data DCR_i services. All computers of mainframe m_i, router set R_i and server set S_i collaborate with each other and push customized data to clients as rapidly and valid as possible.

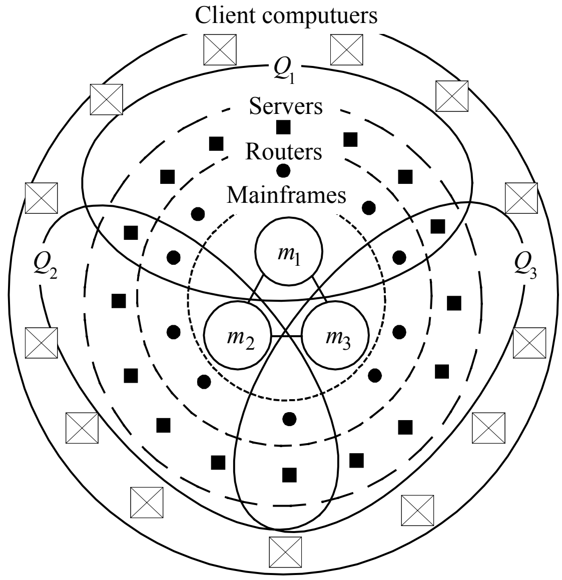


Fig.3 Least non-stop internet service model with 3 mainframes

Fig.3 shows an example of the least Non-stop Internet service model framework based on circle quorum system. Actually, the least model has only one data service unit U_1 composed of quorum Q_1, Q_2 and Q_3, that is because data service unit U_1 equals U_2 and U_3. If any of quorums fails, the other two quorums can take over without stop the data service of the failed quorum.

PROTOCOLS

Duplication protocol

Suppose mainframe m_i supplies data service for its clients and has generated datasets DCL_i and DCR_i in quorum Q_i. Mainframe m_i finishes the duplication of dataset DCL_i as shown below:

(1) $S_i\rightarrow DCL_i=m_i\rightarrow DCL_i$, servers of server set S_i receive dataset DCL_i directly from mainframe m_i through router set R_i;

(2) $m_{(i-1+N)\mathrm{mod}\ N}\rightarrow DBR_{(i-1+N)\mathrm{mod}\ N}=m_i\rightarrow DCL_i$, mainframe $m_{(i-1+N)\mathrm{mod}\ N}$ accepts dataset DCL_i and

keeps backup data $DBR_{(i-1+N)\bmod N}$ of quorum $Q_{(i-1+N)\bmod N}$ synchronized with computation data DCL_i of quorum Q_i;

(3) $S_{(i-1+N)\bmod N} \rightarrow DBR_{(i-1+N)\bmod N} = m_{(i-1+N)\bmod N} \rightarrow DBR_{(i-1+N)\bmod N}$, mainframe $m_{(i-1+N)\bmod N}$ stores and forwards dataset DCL_i to servers set $S_{(i-1+N)\bmod N}$ through router set $R_{(i-1+N)\bmod N}$.

Thus, quorum $Q_{(i-1+N)\bmod N}$ has synchronized its right backup dataset $DBR_{(i-1+N)\bmod N}$ with left computation dataset DCL_i of quorum Q_i, and prepared for taking over dataset DCL_i services at any moment. For dataset DCR_i, mainframe m_i also executes similar steps to keep left backup dataset $DBL_{(i-1+N)\bmod N}$ of quorum $Q_{(i+1)\bmod N}$ synchronized with right computation dataset DCR_i of quorum Q_i. Timestamp, value and type of dataset DCL_i or DCR_i are saved by all servers of server sets $S_{(i-1+N)\bmod N}$ and S_i or S_i and $S_{(i+1)\bmod N}$. The number of valid data of intersected servers of server sets $S_{(i-1+N)\bmod N}$ and S_i or S_i and $S_{(i+1)\bmod N}$ is set to 2, others' number is set to 1.

Takeover protocol

Assume that mainframe m_i crashes or fails, and that it cannot provide dataset DCL_i and dataset DCR_i services for its clients. Eq.(8) and Eq.(9) show that dataset $DBR_{(i-1+N)\bmod N}$ equals DCL_i, $DBL_{(i+1)\bmod N}$ equals DCR_i.

$$DBR_{(i-1+N)\bmod N} = \{DBR^1_{(i-1+N)\bmod N}, DBR^2_{(i-1+N)\bmod N}\} \quad (8)$$
$$DBR^1_{(i-1+N)\bmod N} \cap DBR^2_{(i-1+N)\bmod N} = \phi,$$
$$|DBR^1_{(i-1+N)\bmod N}| = |DBR^2_{(i-1+N)\bmod N}|$$
$$= |DBR_{(i-1+N)\bmod N}|/2.$$
$$DBL_{(i+1)\bmod N} = \{DBL^1_{(i+1)\bmod N}, DBL^2_{(i+1)\bmod N}\} \quad (9)$$
$$DBL^1_{(i+1)\bmod N} \cap DBL^2_{(i+1)\bmod N} = \phi,$$
$$|DBL^1_{(i+1)\bmod N}| = |DBL^2_{(i+1)\bmod N}| = |DBL_{(i+1)\bmod N}|/2.$$

Mainframe $m_{(i-1+N)\bmod N}$ performs takeover operation as follows:

(1) From $m_{(i+1)\bmod N}$ to $m_{(i-1+N)\bmod N}$, it searches the nearest valid mainframe to be its right neighbor. Suppose $m_{(i+1)\bmod N}$ is its valid right neighbor;

(2) $m_{(i-1+N)\bmod N} \rightarrow DCL_{(i-1+N)\bmod N}$
$= \{m_{(i-1+N)\bmod N} \rightarrow DCL_{(i-1+N)\bmod N},$
$m_{(i-1+N)\bmod N} \rightarrow DBR^1_{(i-1+N)\bmod N}\}$.

Mainframe $m_{(i-1+N)\bmod N}$ merges dataset $DBR^1_{(i-1+N)\bmod N}$ into its dataset $DCL_{(i-1+N)\bmod N}$, and it changes data type of $DBR^1_{(i-1+N)\bmod N}$ from 110 (right backup data) to 010 (left computation data) to take over dataset $DBR^1_{(i-1+N)\bmod N}$ services.

(3) $m_{(i-1+N)\bmod N} \rightarrow DCR_{(i-1+N)\bmod N}$
$= \{m_{(i-1+N)\bmod N} \rightarrow DCR_{(i-1+N)\bmod N},$
$m_{(i-1+N)\bmod N} \rightarrow DBR^2_{(i-1+N)\bmod N}\}$.

At one time, mainframe $m_{(i-1+N)\bmod N}$ also adds dataset $DBR^1_{(i-1+N)\bmod N}$ to its dataset $DCR_{(i-1+N)\bmod N}$ and changes data type of $DBR^2_{(i-1+N)\bmod N}$ from 111 (right backup data) to 100 (right computation data). Similarly, mainframe $m_{(i-1+N)\bmod N}$ also takes over backup dataset $DBR^2_{(i-1+N)\bmod N}$ and sends new data of new dataset $DCR_{(i-1+N)\bmod N}$ to its right neighbor $m_{(i+1)\bmod N}$;

(4) $m_{(i-1+N)\bmod N} \rightarrow DBR_{(i-1+N)\bmod N} = m_{(i+1)\bmod N} \rightarrow DCL_{(i+1)\bmod N}$. Mainframe $m_{(i-1+N)\bmod N}$ prepares to receive backup data of $DCL_{(i+1)\bmod N}$ from its right neighbor $m_{(i+1)\bmod N}$;

(5) $R_{(i-1+N)\bmod N} = \{R_{(i-1+N)\bmod N}, R_i\}$, Routers rebuild router set $R_{(i-1+N)\bmod N}$ to redirect data requests from failed mainframe m_i to $m_{(i-1+N)\bmod N}$ and screen failed mainframe m_i;

(6) $S_{(i-1+N)\bmod N} = \{S_{(i-1+N)\bmod N}, S_i\}$. S_i is merged into server set $S_{(i-1+N)\bmod N}$, which means mainframe $m_{(i-1+N)\bmod N}$ supplies non-stop service of data DCL_i for server set S_i.

Mainframe $m_{(i-1+N)\bmod N}$, the left neighbor of failed mainframe m_i, performs the takeover procedure of dataset DCL_i. By merging R_i and S_i into $R_{(i-1+N)\bmod N}$ and $S_{(i-1+N)\bmod N}$, mainframe $m_{(i-1+N)\bmod N}$ will take over dataset DCL_i services without stop when mainframe m_i or quorum Q_i fails. Similarly, mainframe $m_{(i+1)\bmod N}$ can take over dataset DCR_i services of the failed mainframe m_i or quorum Q_i.

Router protocol

Router protocol mainly performs detection of mainframe and other Routers' heartbeat; redirects valid mainframe & screens failed mainframes and Routers, and rebuilding Router list. Client computer's registration processes is initialized as follows:

(1) Client computer sends registration or customized information to mainframe m_i through Routers;

(2) Mainframe m_i transmits received information to servers of its Server set S_i;

(3) All servers of S_i will record IP address and customized information of client computer, receive data from mainframe m_i and push them to client computer.

After the above descriptions, in every cycle $T_{\text{cycle}}=T_{\max}/T_{\text{queue}}$, $0< T_{\text{cycle}} \le T_{\max}$; $T_{\text{Router}_i} = T_{\max}/T_{R_i}$; $0 < T_{\text{Router}_i} \le T_{\max}$, mainframe m_i sends heartbeat to all valid Routers of its Router set R_i; Router i sends heartbeat to other Routers of Router set R_i. T_{queue} is defined as the waiting time of data transmission between mainframe m_i and Routers of Router set R_i; T_{R_i} is the time between Routers of Router set R_i. The process of Router protocol is described as below:

(4) Within $3T_{\max}$, if Router does not receive any heartbeat from mainframe m_i, it requests mainframe m_i to send heartbeat and at the same time it collects the state mainframe m_i from other valid Routers of Router set R_i;

(5) If all Routers of Router set S_i find that the mainframe m_i is inactive, they search for two valid neighbors of failed mainframe m_i and notify left valid neighbor to take over left computation data service of failed mainframe m_i and right valid neighbor to take over right computation data service of failed mainframe m_i;

(6) If clients ask for $m_i \rightarrow D(type=010)$, Routers change it into $m_{(i+1)\bmod N} \rightarrow D(type=101)$ and mainframe $m_{(i+1)\bmod N}$ sends generated data $D(type=101)$ to server set $S_{(i+1)\bmod N}$ including S_i;

(7) Within $3T_{\max}$, if one Router does not get the heartbeat of Router j, it will send a message to other valid Routers to check that Router j activity. Similarly, if one router does not get response from Router j within $3T_{\max}$, it will send a message of Router j's failure to all valid Routers of Router set R_i to rebuild Router configuration;

(8) In each $3T_{\max}$, all valid Routers will send one message to failed Router j to check its recovery. Once one Router gets recovery information of Router j, it will notify other valid to reconstruct Router configuration.

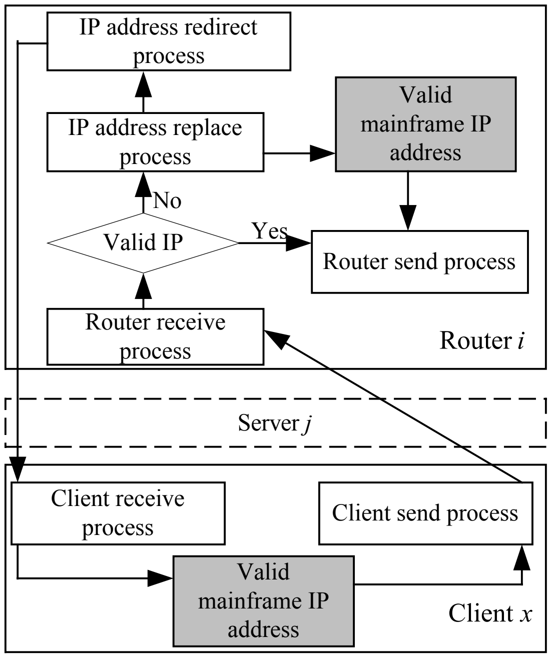


Fig.4 Router redirect process

In detail, Routers run redirection process to transmit and redirect clients' requirements. Fig.4 shows the flow chart of Router redirect process. First, client gets valid mainframe IP address and accesses related mainframe; then one Router checks whether the mainframe IP address is valid. If it is valid, Router will directly transmit client's requirements; or else if data type equals 0XX, Router will change failed mainframe IP address into its left valid neighbor IP address. If data type equals 1XX, Router will replace failed mainframe IP address by its right valid neighbor IP address, and then transmits them; and then, Router sends the client a redirection message, which changes client's failed mainframe IP address into the valid one. Finally, client uses the new valid mainframe IP address to access valid mainframe directly.

Read protocol

Presume a client wants to get its customized data d from data service unit U_i of quorum Q_i. It waits a fixed time T_{delay} to collect data d_i pushed by servers of server set S_i. All received data d_i form a dataset D and the loop variable *Loop* is initialized to be zero.

(1) Step 1: $T_{\text{newest}}=\max\{d_i(timestamp)|d_i\in D\}$, client gets the newest timestamp of dataset D;

(2) $D_{\text{newest}} = \{ d_i' | d_i' \in D, d_i'(timestamp) = T_{\text{newest}} \}$, a new dataset D_{newest} with newest timestamp is obtained;

(3) If $\exists i, j,\ d_i'(value) = d_j'(value)$ and *Loop* $\leq N$, a data failure message is sent to mainframe m_i to recompute data d based on local valid mainframe IP address, and *Loop*++;

(4) If *Loop*>N, Non-stop Internet service model fails, or else returns to Step 1;

(5) $N_{\text{newest}}= \sum d_i'(number)$, client calculates data number of valid data;

(6) If $N_{\text{newest}}< |Q_i|/2$, it triggers valid neighbors of quorum Q_i to send data, and returns to Step 1;

(7) or else, $d= d_i'(value)$, client gets valid data.

From the explanations above, a voting method with newest timestamp can ensure that clients get valid data.

Recovery protocol

Suppose that the failed or crashed mainframe m_i has been recovered and its nearest valid neighbors are $m_{(i-1+N)\bmod N}$ and $m_{(i+1)\bmod N}$. It will execute the followed steps to take its data services back:

(1) $m_i \to DBR_i = m_{(i-1+N)\bmod N} \to DCR_{(i-1+N)\bmod N}$

$$m_i \to DBL_i = m_{(i+1)\bmod N} \to DCL_{(i-1+N)\bmod N}.$$

Mainframe m_i firstly backups computation dataset $DCR_{(i-1+N)\bmod N}$ and dataset $DCL_{(i+1)\bmod N}$;

(2) $m_i \to DCR_i = m_i \to DBR_i$, and $m_i \to DCL_i = m_i \to DBL_i$, mainframe m_i changes data type and takes over computation dataset $DCR_{(i-1+N)\bmod N}$ and $DCL_{(i+1)\bmod N}$;

(3) $m_{(i-1+N)\bmod N}$ and $m_{(i+1)\bmod N}$ release dataset (DCL_i, DCR_i) services by changing data type and make m_i their neighbors by restoring current router set and server set to original ones.

Thus, five protocols are presented to construct the non-stop mechanism of Non-stop Internet service model based on circle quorum system. These protocols indicate all computers cooperate with each other to keep the whole system highly available.

CONCLUSION

The impact of network downtime, once relegated to either financial or specialized industrial applications, is becoming far more significant to a great number of businesses. State Street Company of USA, a world leader in financial services, reconstructed its Fund Broker System as 24×7 global international transaction service system on Internet. Original Fund Broker System, a service system on LAN, was required to be replanted to Internet. High availability and flexible upgrade were required to supply for Non-stop Internet service system. A novel quorum system named circle quorum system, combining with the high availability of quorum systems with distributed computation, was designed in this work. Non-stop Internet service model based circle quorum system was constructed and its five protocols were designed to provide highly available services for clients on Internet, even if some unexpectable failures happen. Currently, this model has been successfully implemented into international transaction system for Boston Mutual Fund Broker, US.

References

Ahamad, M., Ammar, M.H., 1980. Performance characterization of quorum-consensus algorithms for replicated data. *IEEE Trans. Software Eng.*, **15**(4):492-495.

Fu, A.W.C., Wong, M., Yat, S., 2002. Diamond Quorum Consensus for High Capacity and Efficient in a Replicated Data System. Distributed and Parallel Databases, p.1-25.

Kafri, N., Janecek, J., 2002. Dynamic Behavior of the Distributed Tree Quorum Algorithm. 22nd International Conference on Distributed Computing Systems (ICD-

CS'2002), Vienna, Austria.

Kumar, A., 2002. An Efficient Super Grid Protocol for High Availability and Load Balancing. IEEE Transactions On Computer, p.1126-1133.

Lee, C.M., Tam, A., Wang, C.L., 1998. Directed Point: An Efficient Communication Subsystem for Cluster Computing. Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Systems, Las Vegas, Nevada, USA, p.662-675.

Malkhi, D., 2000. Quorum Systems. *In*: The Encyclopedia of Distributed Computing, Joseph Urban and Partha Dasgupta editors. Kluwer Academic Publishers, Philip Drive Norwell, USA.

Malkhi, D., Reiter, M., 2000. An architecture for survivable coordination in large distributed systems. *IEEE Transactions on Knowledge and Data Engineering*, **12**(2):187-202.

Malkhi, D., Reiter, M.K., Alonso, G., Kemme, B., 2001. The load and availability of byzantine quorum systems. *SIAM J. Computer*, **29**(6):1889-1906.

Malkhi, L., Alvisi, P., Reiter, E.M., 1999. Fault Detection for Byzantine Quorum Systems. Proceedings of 7th International Working Conference on Dependable Computing for Critical Applications, California, USA, p.357-372.

Martin, J.P., Dahlin, M., 2002. Small Byzantine Quorum Systems. Proceedings of the International Conference on Dependable Systems and Networks (DSN 2002 and FTCS 32), DCC Track, Washington, DC, p.374-383.

Mohan, R.B., Parmon, B.G., 1998. PARMON: A Comprehensive Cluster Monitoring System. The Australian Users Group for UNIX and Open Systems Conference and Exhibition, AUUG'98−Open Systems: The Common Thread. Baulkham Hills, Australia.

Peleg, D., Wool, A., 1997. Crumbling walls: a class of practical and efficient quorum systems. *Distributed Computing*, **10**(2):87-98.

Peris, R.J., Martnez, P.M., Alonso, G., Kemme, B., 2001. How to Select A Replication Protocol According to Scalability, Availability, and Communication Overhead. Proc. of the Int. Symp. on Reliable Distributed Systems (SRDS), New Orleans, Louisiana.

Tsuchiya, T., Kikuno, T., 2002. Byzantine quorum systems with maximum availability. *Information Processing Letters*, **83**:71-77.

Yin, J., Martin, J.P., Alvisi, V., Dahlin, M., 2002. Fault-Tolerant Confidentiality (updated), International Workshop on Future Directions in Distributed Computing (IWFDDC'2002), Bertinoro, Italy.