A Large-Capacity Service Control Node Architecture Using Multicasting Access to Decentralized Databases in the Advanced Intelligent Network

Etsuo MASUDA\(^{1}\), Regular Member, Takeshi MISHIMA\(^{1}\), Nonmember, Naoki TAKAYA\(^{1}\), Kohei NAKAI\(^{1}\), and Masanori HIRANO\(^{1\dagger}\), Regular Members

SUMMARY Focusing on a distributed control service-control-node (SCP) that houses a database (DB) distributed across multiple modules, this paper proposes an autonomous distributed SCP architecture using multicasting access to the distributed DB, and highlights its application areas. We assume as a basic condition that neither the network nor the other modules in the system are aware of the DB configuration. Based on this condition, we propose two basic methods: a unicast approach in which the DB management module that is selected at random by the network routes the DB access request to the module where the target data resides (Method A), and a multicast method in which DB access requests are broadcast to all modules (Method B). A quantitative evaluation is made of the number of required modules and required communications performance between modules which is determined by the capacity of the main memory and processing capacity of the processors. Based on the results, we conclude that Method B better exploits the advantages of module autonomous distribution technology within the limits that the economy of inter-module communication overhead is not impaired. Furthermore, in the event a module fails in Method B, a scheme is proposed in which the defective module is cut out of the multicast group, and multicasting continues. This could be implemented most effectively using a separate route under hardware control that is independent of the on-line communications route between modules.

\textit{key words:} autonomous distribution, multicast, distributed database, SCP, intelligent network

1. Introduction

Personal communications services such as mobile and cellular and Personal Handy-phone System (PHS) services continue to grow at a phenomenal rate [1]. The number of mobile phone contracts had already exceeded 60 million by the end of the year 2000.

The Advanced Intelligent Network (AIN) [2] that maintains terminal location information, authentication information, and other data in databases (DBs) on the network is essential for the provisioning of personal communications services. Since location and authentication information must be maintained for every terminal, it is obvious that very-large-scale DBs that can accommodate more than 100 million (= \(10^8\)) entries will be required in the near-term future.

Meanwhile, autonomous distributed systems have also attracted enormous interest in recent years, for this approach can be used to efficiently implement systems using multiple equipment units that vary widely in scale and can operate independently of other units [3]. A notable feature of autonomous distributed systems is that such systems can be configured without the network or the constituent distributed modules themselves knowing how the system is configured, and this simplifies both the implementation and the maintenance of the system.

This paper investigates an architecture supporting efficient access to a distributed control service-control-node (SCP) in which the DB resides on multiple modules. More specifically, we propose an autonomous distributed architecture that employs multicasting access to the distributed DB, and highlight the areas where this approach could be applicable.

In analogous work done earlier, Kotera et al. [4] proposed a method of implementing a distributed database management system in a wide-area network from the standpoint of hiding the distributed configuration. These authors developed an original and highly reliable multicasting scheme for accessing the database. This study also referred to a mechanism for dynamically adjusting the multicast range and a way to ensure that the system continues to operate normally when a particular site fails. However, the study failed to consider how such a system might be actually implemented, and was short of quantitatively evaluating their proposed scheme in terms of performance.

In other related work, Nakao et al. [5] proposed a scheme in which the remote operator is unaware of the distributed configuration on the host side for application to an enterprise information network. In their approach, commands entered from a remote terminal are received by an operating host and passed on to multiple processing hosts by broadcasting. In the event that a processing host fails, delays in the command response time are avoided in this scheme by removing the defective host from the group receiving broadcasts. Here
again the authors failed to offer a quantitative evaluation of their proposed scheme, or consider its application areas based on the results of such a quantitative evaluation.

As a way of implementing wide-area distributed multiple SCPs in the advanced IN, Suzuki et al. [6] considered a novel approach to making the distributed topology invisible by having the switch side access any SCP, then having that SCP look up the desired SCP in a directory. This is of course very different from what we consider in this paper; namely, distribution invisibility when multiple independently configured SCPs are deployed.

Masuda et al. [7] compared two methods of implementing an Signaling System No. 7 (SS7) signal processing capability under the condition that adjacent nodes are unaware of the distribution configuration for application to distributed control switching nodes that interwork with the SS7 network.

Meanwhile, paralleling the rapid development of Internet-related technologies, attempts have been made to leverage the multicasting capabilities of the Internet Protocol (IP), and various schemes adopting this approach have been proposed [8]. Y.-I. Chang et al. [9] proposed a highly reliable token-passing broadcast communications protocol that circulates a token around a ring of interconnected nodes. In another study, Shiroshita et al. [10] proposed a reliable multicast protocol that adds transport layer delivery verification and redelivery features to the network layer multicast capability, and discussed the applicability of this approach to various network topologies. Finally, S.C. Liew [11] proposed a multicasting scheme that is so simple and generic in its conception that it can be applied virtually any interconnected network.

In contrast to these other approaches, this paper proposes an autonomous distributed SCP architecture that adopts an original multicast communications protocol for the AIN that can handle mobile communications and other advanced services. Although there are relatively few (around tens of) DBs, high reliability is required because the data stored in DBs is used for call control. Based on a quantitative evaluation of the proposed method, we clarify areas where the proposed method might be optimally applied.

The paper is organized into seven sections. Section 2 begins with an overview of the configuration of the AIN and the physical configuration of the distributed control SCP that is the focus of this study. In Sect. 3, we discuss a fundamental control condition of the distributed control SCP, the ability to access DBs without any awareness of the distribution configuration of the DBs. Based on this condition, we propose two basic methods: a unicast approach in which the DB management module that is selected at random by the network routes the DB access request (transaction) to the module where the target data resides (Method A), and a multicast method in which DB access requests are broadcast to all modules (Method B). We also introduce the inter-module communication scheme for the methods. In Sect. 4, we evaluate the number of required modules which is determined by the capacity of the main memory and processing capacity of the processors, and the required communication performance between modules for the two methods. Based on the results, we conclude that Method B better exploits the advantages of module autonomous distribution technology within the limits that the economy of inter-module communication overhead is not impaired. Focusing on Method B, in Sect. 5 we consider strategies for dealing with module failures. Here we propose a scheme in which the defective module is dynamically removed from the multicast group, and the multicasting continues without interruption to all the other modules. This could be most effectively implemented using a separate route under hardware control that is independent of the on-line communications route between modules. Section 6 presents an actual implementation of the proposed architecture, and Sect. 7 highlights the key points of the paper and suggests a number of issues that should be addressed in future work.

2. Configurations of the AIN and Distributed SCP

2.1 Configuration of the AIN

Figure 1 shows a schematic representation of the AIN structure. As shown in the figure, the AIN consists of Service Switching Points (SSPs) that accommodate terminals, SCPs that store DBs and perform advanced switching control, and SS7 network that transfers SS7 signals between SCPs and SSPs. Note that the SS7 network includes Signal Transfer Points (STPs). The layer including SSPs and the main data trunks is called
the *transport layer*, while the layer including SCPs is referred to as the *intelligent layer*. The AIN supports *Free Phone*, PHS, and other advanced services using conversion tables that calculate the actual physical location of terminals from logical terminal numbers that are stored in DBs in the SCPs.

In AIN services such as *Free Phone* and PHS, a logical number is used for the dial number to specify the called party. The SSP receiving the call calculates which SCP holds the information corresponding to the logical number that was dialed, and queries that SCP over the SS7 network. When it receives a query, the SCP searches its DB to find the corresponding location information, then sends the information back to the SSP that initiated the query. The SSP then sets up and executes the call based on the location information. It will be apparent that, up to now, AIN services have been implemented based on a unicast approach to a specific SCP.

### 2.2 Distributed SCP Configuration

In personal communications, the terminal location, authentication data, and other information used when dialing the number that has been assigned to a terminal are stored in a DB at an SCP. If the demand for personal communications services continues to grow as projected, this means that SCP will have to accommodate very-large-scale DBs.

As illustrated in Fig. 2, there are two basic strategies that SCPs could be made to accommodate large-scale DBs:

(A) Multiple SCPs each with its own DB: multiple SCPs are deployed as independent node on the network each with its own DB.

(B) Single distributed control SCP with multiple modules: a single large-scale distributed control SCP is deployed that has the ability to interconnect multiple modules that contain a distributed DB.

In the case of relatively small-scale DBs, Strategy A is clearly more cost effective since it does not require additional equipment to support inter-module communication, but Strategy B becomes preferable when larger scale DBs are needed and the rate of equipment cost increase diminishes.

In this work we will be concerned with the Strategy B approach—a single large-scale distributed control SCP with multiple modules—on the assumption that the demand for personal communications services will continue to grow in the years ahead. Figure 3 shows the physical configuration of the SCP assuming the Strategy B approach. As apparent from the figure, the SCP consists of multiple modules ($M_1$ to $M_n$) where databases are stored, and an Inter-Module Controller (IMC) that interconnects the modules. Considering the requirement for real-time performance in setting up and terminating calls, the DBs are implemented in main memory (MM). Although not shown in the figure, the DBs are backed up to file memory to enable recovery in the event that the module fails [12].
3. Distributed SCP Control Condition and Proposed Implementation

3.1 Control Condition

From the standpoint of both simple implementation and maintenance of AIN, it is desirable that SSPs can access DBs without being aware of the distributed DB topology. This means that, when an SSP sends a DB access request to an SCP via the SS7 network, it is not necessary for the switch to select and specify the address of the distributed DB. This can be achieved in one of two ways:

(ND 1) Database access request from the SS7 network is multicast to all the DB control modules in the SCP. Each module receiving the request checks to see if there is its entry or not.

(ND 2) Database access request from the network is sent via any SS7 link to a module in the SCP, and the request is sent to the target DB through further processing within the SCP.

In the Scheme ND1, nothing needs to be done to achieve distributions invisibility inside SCP, while in Scheme ND2, it will be apparent that an invisibility condition exists. That is, DB access requests received by the SCP from the SS7 network do not specify the target DB, so an additional condition is required in order to access the target DB.

3.2 Proposed Method

3.2.1 Selecting the Proposed Method

There are two ways to implement the internal SCP processing implicit in ND2: one way does not ensure distribution invisibility within the SCP (ND21), while the other way does ensure distribution invisibility in the SCP (ND22).

(ND21) Unicast DB access to a specific module: The module receiving the DB access request from the SS7 network determines the destination database, and relays the request to the right module. For this scheme to work, the module receiving the DB access request must know the whereabouts of all data in the SCP.

(ND22) Multicast DB access to all modules: Incoming DB access requests from SS7 are multicast to all other DB management modules. The module receiving the request checks to see if there is its entry or not.

Out of the three schemes ND1, ND21, and ND22, two of the methods (ND1 and ND22) achieve total distribution invisibility, and are referred to as fully autonomous distribution systems (FADS). The rest of the methods, ND21, achieves incomplete or partial distribution invisibility, and is therefore referred to as a partial autonomous distribution system (PADS).

Practical deployment of the FADS scheme ND1 would involve modifying the SS7 protocol itself, and since this would entail such a huge impact on implementation, this scheme has been excluded from further consideration. In the rest of the paper, we will concentrate on the two remaining methods: the PADS scheme ND21 and the FADS scheme ND22, that we will refer to as Method A and Method B, respectively. We will offer a quantitative evaluation of the two methods, then discuss their application areas based on the results.

3.2.2 Overview of the Proposed Methods

Figure 4 shows a schematic overview of the two methods under consideration. In Method A, DB access requests are received by a module, then unicast to a selected destination. In other words, any module (M_i in the figure) receiving a DB access request from the SS7 network (a) determines the correct destination module (M_n) by checking its internal control table, and unicasts the request to the selected destination (b). Module M_n then locates the requested data entry on its DB and returns the access results to the SS7 network (c).

Now turning to Method B, the module receiving the DB access request multicasts the request to all the

Fig. 4 Distributed databases access methods.
modules, and each module determines if it is the intended destination or not. In other words, any module (\(M_1\) in this case) receiving a DB access request from the SS7 network (a) multicasts the request to all the other modules (b) and each module attempts to locate the requested data entry on its DB as in the destination module \(M_n\) of the Method A. The module containing the requested data (\(M_n\) in the figure) locates the corresponding data entry and returns the access results to the SS7 network (c), while the other modules terminate in failing to locate the requested data entry.

We explain the processing flow within modules in more detail later at Sect. 4.1.2.

The basic difference between the two schemes is that in Method A each module must maintain an additional correspondence table showing the correspondence between all the data stored in the distributed DB and all the modules making up the SCP, while in Method B the modules not containing the requested data require extra processing of attempting to locate the data entry on their DBs. In comparing the two schemes, Method B with no correspondence table required has an advantage in terms of the amount of memory, but Method A with no extra data-entry-location processing and its less frequent inter-module communications has an advantage in terms of processing amount and required inter-module communications performance.

3.3 Inter-Module Communication Implementations

In a distributed control system like an SCP in the AIN, there are relatively few (around several tens of) modules and the packet size of DB access requests (transactions) sent between modules is relatively small (100 to 300 bytes). At the same time, however, the real-time performance (around several tens of milliseconds) and reliability of inter-module communications are extremely important, because the DB accesses are necessary for call setup.

Considering the points raised above, the inter-module communication implementation method adopted in this work and shown schematically in Fig. 5 will have the following features:

- The lower layer implements ATM based on AAL5, the upper layer implements an original transport layer. In Method B, the multicasting capability is implemented on this layer by firmware (\(\mu P\)).
- Frames received from the CPU (upper level software) are broken down into corresponding ATM cells on AAL5 and sent to their destinations. In Method B, several copies of frames are generated by the transport layer and sent to destinations. Confirmation of send acknowledgment and retransmission against NACK messages are done on a per-destination basis. When receipt acknowledgments are received from all destinations, a multicast end notification is sent from the transport layer to the upper layer.

4. Evaluation of the Two Proposed Methods

In this section we will evaluate the number of required modules and the inter-module communication performance of the two methods as a function of the number of users accommodated by the SCP, and discuss the optimal application areas of the two approaches.

4.1 Evaluation of the Required Number of Modules

In distributed control SCPS, the capacity of each module is generally determined by the capacity of main memory for storing the DB and the processor performance. In this section we will evaluate the required number of modules assuming limited main memory and processor processing capacity.

4.1.1 Limited Main Memory Capacity

As noted earlier in Sect. 2.2, the DB in an SCP resides in main memory. Here we will evaluate the required number of modules determined by limited main memory capacity.

(1) Evaluation Equations

In the following, the number of required modules as determined by main memory capacity will be denoted by \(N_A\) for Method A and by \(N_B\) for Method B, where \(x \times 10^6\) user/SCP is the number of users accommodated by each SCP, \(\alpha\) (bytes/user) is the data size per user, \(\beta\) (bytes/user) is the data entry size in the control table, and \(\gamma\) (GB) is the main memory capacity.
(a) Required Number of Modules $N_A$ for Method A
Since the total required capacity to store user data and control data is less than $\gamma \text{ [GB]}$, this leads to the equation:

$$\alpha \frac{x}{N_A} + \beta x \leq \gamma \times 10^9$$  \hspace{1cm} (1)

$$\therefore \quad N_A \geq \frac{\alpha x}{\gamma \times 10^9 - \beta x}$$  \hspace{1cm} (2)

Since the denominator of the right term in Eq. (2) must be larger than 0, then $x < \frac{\gamma \times 10^9}{\beta}$. In other words, this method can not accommodate more than $\frac{\gamma \times 10^9}{\beta}$ users.

(b) Number of Modules Required $N_B$ for Method B
In Method B, it is sufficient if the main memory of each module can accommodate the data of as many as the users accommodated by the module. There is no need to maintain a control table. The required number of modules for Method B is therefore given by

$$\alpha \frac{x}{N_B} \leq \gamma \times 10^9$$  \hspace{1cm} (3)

$$\therefore \quad N_B \geq \frac{\alpha x}{\gamma \times 10^9}$$  \hspace{1cm} (4)

(2) Evaluation Results and Considerations
Figure 6 shows the number of required modules versus the number of user entries evaluated from the standpoint of main memory capacity. Here it is assumed that the main memory capacity $\gamma$ is implemented with currently prevailing 32-bit microprocessors ($\mu P$) up to a maximum of 4 GB (= $2^{32}$), and either 2 GB or 3 GB can be allocated to the DBs. The data size per user $\alpha$ is given as 400 B assuming PHS service implementation where 30 B for basic connection, 120 B for optional connection services such as redirections, 50 B for charging, and 200 B for future extension. The data entry size in the control table $\beta$ is given as 20 B assuming 12 B for terminal number, 4 B for module identification number, and 4 B for additional temporary data.

In Method A, each module must include a control table. This puts pressure on the DB area of the main memory as the number of users per SCP escalates, and the number of modules observes quadratic curve growth. Now turning to Method B, each module manages the number of user entries on the DB, so the number of required modules increases linearly with increasing user entries. Thus from the standpoint of limited main memory capacity, it would be preferable to adopt Method B.

4.1.2 Limited Processor Processing Capacity
Next we will evaluate the required number of modules as determined by the processing capacity of processors.

(1) Processing Flow Within Modules
Figure 7 is a schematic showing the processing flow within modules. In Method A, when an SS7 signal ar-
rives from the SS7 network (Ds), the module containing the target information is determined (Dd) based on address information in the signal. The access request is sent to the appropriate module by a unicast message to a specified destination (P-P) (Dc), and the receiving module accesses the target user entry on the DB based on access request information in the received signal (Du, Da). The corresponding read data and write complete notifications are sent back to the SS7 network as SS7 signals (Ds).

Now turning to Method B, when an SS7 signal arrives from the SS7 network, the access request is sent by multicasting (P-MP) to all the other modules (Dc). Each module receiving the message then checks to see if the target user entry is in the module’s DB based on access request information in the received signal (Du), and if it is, the entry is accessed (Da). Again, the corresponding read data and write complete notification is sent back to the SS7 network as an SS7 signal (Ds).

One can observe in Fig. 7 that Method B does not require the “Destination decision” block (Dd) needed in Method A, but the “User entry retrieval?” block (Du) is required for every transaction.

In searching for entries in the “Destination decision” (Dd) and “User entry retrieval” (Du) processing, a binary search method is used for maximum efficiency.

### (2) Evaluation Equations for Required Number of Modules

The rate of incoming DB access requests from the SS7 network $\lambda_x$ (transactions per second) is given by

$$\lambda_x = \frac{x \times 10^6 \times 0.005}{100} \times v$$

where the busy hour call rate is 0.005, the average holding per call is 100 seconds per call, $v$ is the number of transactions per call, and $x$ ($\times 10^6$ users/SCP) is the number of users per SCP. In the following, note that $\rho$ ($\times 10^6$ instructions per second) is the processing capability of the processor in MIPS.

#### (a) Required Number of Modules in SCP for Method A $n_A$

By referring to processing flow model shown in Fig. 7, the following equation holds.

$$\frac{\lambda_x}{n_A(x)} \{D_s + D_d + D_c + D_a + D_a + D_s\} \leq \rho$$

where

$$\frac{\lambda(x)}{n_A(x)} \left\{2D_s + 2D_a + D_0 \log_2 x + n_a(x) + D_a\right\} \leq \rho$$

#### (b) Required Number of Modules in SCP for Method B $n_B$

$$\frac{\lambda(x)}{n_B(x)} \{D_s + D_c + n_B(x)D_c + n_B(x)D_a + D_a\} \leq \rho$$

where

$$\frac{\lambda(x)}{n_B(x)} \left\{2D_s + \{n_B(x) + 1\}D_c + n_B(x)D_0 \log_2 x \right\} \leq \rho$$

This is providing that $\xi(x)$ is the minimum value of $n_A$ that satisfies Eq. (6) with respect to the given value of $x$.

### (3) Evaluation Results and Considerations

Typical evaluation results are presented in Fig. 8. In terms of number of required modules determined by processor capacity, it is apparent that Method A is advantageous across all areas. In Method B, the number of required modules observes quadratic curve growth, but this is because inter-module communication processing increases due to multicasting, and other processing ceases.

#### 4.1.3 Summary

Assuming a 32-bit architecture, the number of modules increases sharply due to the systematic bottleneck
in main memory capacity, and Method A is limited to about 100 \((\times 10^6)\) users. In Method B, on the other hand, the processors can be speeded up without running into this systematic restraint, and promises to accommodate much greater numbers than Method A by exploiting the recent enhancements in processor performance. Figure 9 illustrates the application areas of both methods. The application area is determined according to which method is more advantageous in the number of modules required considering both MM capacity and processor processing capacity. The graph in Fig. 9 is the boundary of application areas of both methods and shows the relationship between processor capacity \(\rho\) MIPS and MM capacity \(\gamma\) MB. The graph has been drawn so that the equation \(\gamma = N_A^{-1}(n_B\text{ for }\rho)\) may be satisfied.

Our evaluation is that Method B offers the better overall performance in this regard.

4.2 Evaluation of Required Inter-Module Communication Performance

4.2.1 Required Inter-Module Throughput

(1) Evaluation Conditions
In this section we will evaluate the required throughput (transactions per second) of the IMC interface and the communications bearer rate (bytes per seconds) assuming the multicast communications scheme presented in Sect. 3.3. To avoid complicating the evaluation we will assume that there is no imbalance in the frequency of accesses to particular DBs; in other words, we assume a uniform distribution.

(2) Evaluation Equations
Let \(x\) be the number of users accommodated by the SCP and \(\lambda(x)\) be the number of transactions for each unit of time from \(x\) users, and \(n_A(x)\) and \(n_B(x)\) be the number of required modules for Method A and B, respectively. In the following, we show equations for evaluating the required throughputs \(\tau_A(x)\) and \(\tau_B(x)\) per IMC interface for the two methods.

(a) Unicast to a Specific Module SCP (Method A)
The transmission throughput \(\tau_A^t(x)\) must support
\[
\tau_A^t(x) = \frac{1}{n_A(x)} \cdot \frac{\lambda(x)}{n_A(x)} \quad \text{(transactions/second)}
\]
to all other modules except the originating module \(n_A(x) - 1\) and is therefore given by
\[
\tau_A^t(x) = \left\{ n_A(x) - 1 \right\} \times \frac{1}{n_A(x)} \cdot \frac{\lambda(x)}{n_A(x)}
\]
\[
= \frac{n_A(x) - 1}{n_A(x)^2} \cdot \lambda(x) \quad (10)
\]

On the other hand, since each module must be capable of receiving
\[
\frac{1}{n_A(x)} \cdot \frac{\lambda(x)}{n_A(x)} \quad \text{(transactions/second)}
\]
from all other modules except the module in question \(n_A(x) - 1\), the receiving throughput \(\tau_A^r(x)\) is given by
\[
\tau_A^r(x) = \left\{ n_A(x) - 1 \right\} \cdot \lambda(x) \quad (11)
\]

Thus, the total transmission and reception throughput \(\tau_A(x)\) is derived by combining Eqs. (10) and (11) to yield
\[
\tau_A(x) = \frac{2\left\{ n_A(x) - 1 \right\}}{n_A(x)^2} \cdot \lambda(x) \quad (12)
\]

(b) Multicast to all Modules SCP (Method B)
The transmission throughput \(\tau_B^t(x)\) must support \(\frac{\lambda(x)}{n_B(x)}\) (transactions/second) to all other modules except the originating module \(n_B(x) - 1\) and is therefore given by
\[
\tau_B^t(x) = \left\{ n_B(x) - 1 \right\} \times \frac{\lambda(x)}{n_B(x)}
\]
\[
= \frac{n_B(x) - 1}{n_B(x)} \cdot \lambda(x) \quad (13)
\]

Just as on the transmit side, the receive side throughput \(\tau_B^r(x)\) must be \(\frac{\lambda(x)}{n_B(x)}\) (transactions/second) from all other modules except the self module \(n_B(x) - 1\) and is given by
\[
\tau_B^r(x) = \left\{ n_B(x) - 1 \right\} \cdot \lambda(x) \quad (14)
\]

Thus, total transmission and reception \(\tau_B(x)\) is given by
\[
\tau_B(x) = \frac{2\left\{ n_B(x) - 1 \right\}}{n_B(x)} \cdot \lambda(x) \quad (15)
\]
(3) Evaluation Results and Considerations

Typical evaluation results for the required inter-module communications throughput are shown in Fig. 10. In light of our considerations in the previous section, here we will determine the number of required modules for 100-MIPS capacity processors. Method B requires slightly less than one order of magnitude greater throughput than Method A. As an example, in order to accommodate $100 \times 10^6$ users per SCP, throughput more than $2.8 \times 10^4$ tr/sec is required. If the throughput of IMCI is $1.6 \times 10^4$ tr/sec to be described later in Sect. 6, two IMCIs for each module must be equipped for this requirement.

4.2.2 Required Bearer Rate between Modules

(1) Evaluation Conditions

Let $m$ (bytes per transaction) be the length of a packet of DB access request. Based on the required throughput outlined in the previous section, we give in the following sections the required bearer rate between modules $\omega_A(x)$ MBytes/sec and $\omega_B(x)$ MBytes/sec, respectively. Here we will assume an upper limit duty factor of 0.7 for each bearer rate to avoid considering deteriorating quality when traffic volumes are high.

(2) Evaluation Equations

(a) Unicast to a Specific Module SCP (Method A)

Using Eq. (12), we have

$$\frac{2(n_A(x) - 1)}{n_A(x)^2} \cdot \lambda(x) \times m \leq 0.7 \omega_A(x) \times 10^6$$

$$\therefore \omega_A(x) \geq \frac{2m(n_A(x) - 1) \cdot \lambda(x)}{0.7 \times 10^6 \times n_A(x)^2}$$

(16)

(b) Multicast to all Modules SCP (Method B)

Similarly, using Eq. (15), we have

$$\frac{2(n_B(x) - 1)}{n_B(x)^2} \cdot \lambda(x) \times m \leq 0.7 \omega_B(x) \times 10^6$$

$$\therefore \omega_B(x) \geq \frac{2m(n_B(x) - 1) \cdot \lambda(x)}{0.7 \times 10^6 \times n_B(x)}$$

(17)

(3) Evaluation Results and Considerations

Figure 11 shows examples of required bearer rates between modules. The profiles exhibit a similar tendency to those we saw in Fig. 10 for the transaction packet length. For example, Method B requires a bearer rate of 10 MBytes/sec for a packet length of 128 Bytes. With the technology that is available today, this should not present any major difficulties.

4.3 Evaluation Summary

In the case of Method B, the throughput and bearer rate can be increased by simply increasing the number of IMC interfaces. Method B that requires less equipment costs and takes full advantage of autonomous distribution thus has an advantage over Method A that requires additional modules due to limited memory.

5. Dealing with Module Failures

In this section we will consider countermeasures for dealing with the failure of a module in a multicast environment.

5.1 Proposed Method for Dealing with Failures

In DB access methods based on multicasting, the module first receiving the DB access request from the network does not know where the target data resides. It only becomes apparent which module has the data after each module checks to see if it contains the data that is requested. This means that, even if a particular module fails and is taken off-line, you cannot selectively discard
access requests for data that happens to be stored on that module.

In this section we will consider two basic strategies to deal with module failures, and compare the two in terms of connection losses of DB access requests.

5.1.1 Strategies for Dealing with Module Failures

Figure 12 illustrates two strategies for dealing with module failures from the time they occur until service is restored, in one approach multicasting is suspended and in the other it is continued.

**Scheme FA**: Multicasting is suspended: When a module fails, multicasting is suspended until the module is restored, and DB access requests generated during that interval are lost.

**Scheme FB**: Multicasting is continued: When a module fails, multicasting is continued to all of the other modules in the group except the module that is down. When no delivery acknowledgment (DACK) is received in response to a message from any module, that message is considered to be lost.

5.1.2 Evaluation Examples and Considerations

Solving state transition equations with $P_i$ ($i = 0, 1, 2$), we have

\[
P_0 = \frac{1}{1 + n\lambda T + n(n-1)\lambda^2T^2}
\]  

\[
P_1 = n\lambda T
\]  

\[
P_2 = n(n-1)\lambda^2T^2
\]

where $n$ is the number of modules, $\lambda$ is the failure rate of each module, $T$ is the repair time, and the probabilities of 0, 1, or 2 out of $n$ modules being in a down state are represented by $P_0, P_1, P_2$, respectively. The probability of 3 or more modules failing at the same time is ignored.

Because the loss rate ($L$) per DB access request is given by $\sum_i \{\text{number of down modules} (i) \times \text{DB access request module hold density} (\delta) \times \text{module down state probability} (P_i)\}$ ($i = 1, 2$), loss rates $L_A, L_B$ for the two schemes are given by

\[
L_A = n(x) \cdot \delta (P_1 + P_2)
\]  

\[
L_B = 1 \cdot \delta P_1 + 2 \cdot \delta P_2 = \delta (P_1 + 2P_2)
\]

where $n(x)$ is the number of modules when the SCP is accommodating $x$ number of users.

Typical evaluation results for the two schemes are presented in Fig.13. In the evaluation, we used $\lambda^{-1} = 5000$ hours and $T = 2$ hours based on the performance of analogous systems in the past and we used the value of $\delta$ derived from the dynamic instruction steps for processing DB access request and processor performance (100 MIPS). Since processor performance is more likely to produce a bottleneck than main memory capacity in multicast systems, $n(x)$ is calculated based on the assumption that the number of modules is determined from the processor performance. Figure 13
shows that \( n(x) \) increases in Scheme FA as the number of users that are accommodated increases, which means that the loss rate increases sharply according to Eq. (13). In Scheme FB, the amount of increase of state probabilities (11), (12) is relatively minor, so the amount of increase is small. The loss rate of Scheme FB is close to the same as a method in which multicasting is not used (omitted from the figure) and accessing failed modules is avoided.

5.2 Method of Modifying the Multicast Group

In distributed DB management systems, failed modules are removed from the multicast group. It is therefore critically important to detect a failed module quickly, to notify the other modules, and to disconnect links between all the operating modules and the failed module. Here we will consider two schemes for implementing these steps.

5.2.1 Proposed Methods

In this work we investigated two approaches—Method GA and Method GB—shown schematically in Fig. 14. When a module fails in Method GA, the other modules are notified via a Remote Maintenance Center (RMC). When a module fails in Method GB, the other modules are notified via separate dedicated routes (that are only under hardware control) that are deployed alongside the inter-module communication routes.

In Method GA, when a hardware control device detects a failure of its own module, then it notifies the failure to the Remote Maintenance Center via a hardware control route. RMC issues a command blocking access to the failed module via a software control route. Both hardware and software control routes are usually connected with RMC for maintenance operations, so no new equipment needs to be deployed with this scheme. In Method GB, blocking instructions disconnecting links with a failed module are sent over a dedicated route, so new equipment and cabling does need to be deployed.

5.2.2 Evaluation Examples and Considerations

From the time a module fails until all links with the failed module have been blocked (represented as \( T_a \) in Method GA and as \( T_b \) in Method GB), multicasting continues, so the DB accesses that take place during this interval are obviously affected. Here we will take a closer look at the difference between \( T_a \) and \( T_b \) regarding the second approach outlined above, and evaluate the number of DB access calls that would be affected. The results are presented in Fig. 15.

In the highly reliable multicasting scheme introduced in Sect. 3, Method GB that notifies modules immediately when a module fails is preferable, because the next multicasting message is not sent out until all destinations have acknowledged receipt of the previous multicasting message. Implementation of Method GB uses communications equipment installed for communicating between each module and the Remote Maintenance Center. Using the equipment’s IP protocol control capability, other modules can notified that a particular module is down very efficiently by broadcasting that module’s IP address (a pattern of all 1s). The dedicated route is not just used for this one purpose, but rather is used to convey a wide range of information.

![Fig. 14](image-url) Multicasting group modifying methods.

![Fig. 15](image-url) No. of uncompleted DB accesses.
pertaining to equipment failures, and thus should enhance the reliability of the system as a whole.

6. Implementation Example

A distributed control system was implemented based on our systematic evaluation of different system options. In order to avoid the problem of a system bus bottleneck, the IMCI is housed in the CPU and is not routed over the system bus. The CPU is an Ultra-SPARC running at 400 MHz, and a separate µP (called MPC8260) is equipped on IMCI for communication processing firmware. The IMC is configured using an ATM switch, capable of being expanded up to 32 ports. Communications performance between processors was evaluated by a full-scale simulator implemented based on an actual system, and performance of $1.6 \times 10^4$ transactions per second was demonstrated for a 1-to-16 multicasting topology.

7. Conclusions

In this work we have investigated an architecture supporting efficient access to a DB that is distributed across multiple modules in a distributed control SCP under the condition that the modules are unaware of the distribution configuration. Here we will highlight the principle results of the study.

(1) Assuming as a basic condition that neither the network nor the other modules in the system are aware of the DB configuration, we proposed two basic schemes: a unicast approach in which the DB management module that is selected at random by the network routes the DB access request to the module where the target data resides (Method A), and a multicast method in which DB access requests are broadcast to all modules (Method B).

(2) Regarding the two methods, a quantitative evaluation was made of the number of required modules and required communications performance between modules which is determined by the capacity of the main memory and processing capacity of the processors. Based on this evaluation, it is found that Method B better exploits the advantages of module autonomous distribution technology within the limits that the economy of inter-module communication overhead is not impaired.

(3) In the event a module fails in Method B, a scheme is proposed in which the defective module is cut out of the multicast group, and multicasting continues. This could be implemented most effectively using a separate route under hardware control that is independent of the on-line communications route between modules.

In contrast to unicasting, when multicasting is used for communications between modules, the demands placed on inter-module performance increase proportionately as the number of modules increases, and this causes systematic limitations to emerge as the number of modules increases. In order to implement distributed systems that are made up of an even greater number of modules, we would like to see communication performance expanded even more in the years ahead. System-level comparisons must also be made with a direct multicasting schemes in which data is multicast from the network to DB management modules in each node.

We would also note that the approach advocated here could also be applied in the same way to CAs (call agents) when distributed control CAs are integrated with GKs (gate keepers) in existing telephone networks (PSTN) and IP integrated networks.

Acknowledgements

The authors would like to express their appreciation to project manager Kouji Kogure and to research group leader Naoaki Yamanaka at NTT Network Service Systems Laboratories. They are also grateful to Yoshihito Kawamura and all colleagues concerned in this work for their valuable discussions and suggestions.

References

[12] M. Hirano, M. Yamane, M. Yamazaki, Y. Kinouchi, and

Etsuo Masuda received the B.E., M.E. and Ph.D. degrees from the University of Electro-Communications in 1975, 1977, and 2000 respectively. Since joining NTT in 1977, he has been active in the research and development of common channel signaling equipment, traffic design of switching systems, the control systems hardware of the next generation switching node. Currently, he is a Senior Research Engineer at the NTT Network Service Systems Laboratories, where he is engaged in the development of the Media Gateway Controller (MGC) in the integrated PSTN-IP network as well as service control node (SCP) in the Advanced Intelligent Network.

Takeshi Mishima received the B.E. and M.E. degrees in information science from the University of Tsukuba in 1994 and 1996, respectively. He joined NTT in 1996 and has been involved in the research and development of control system architecture for the switching system. Currently, he is engaged in the research and development of the network service system architecture at NTT Network Service Systems Laboratories. He is a member of the Information Processing Society of Japan (IPSJ).

Naoki Takaya graduated from Keio University, Japan, with the M.E. degree in engineering in 1995. He joined Nippon Telegraph and Telephone (NTT) Corporation, Tokyo, Japan, in 1995. He has been involved in the research and development of xDSL systems and ATM switching systems. Currently, he is engaged in the research and development of the Media Gateway Controller (MGC) in the integrated PSTN-IP network at NTT Network Service Systems Laboratories.

Kohei Nakai received the B.E. and M.E. degrees in electronic engineering from the University of Tokyo, Japan, in 1995 and 1997, respectively. In 1997, he joined Nippon Telegraph and Telephone Corporation’s (NTT) Network Service Systems Laboratories, Tokyo, Japan. He has been involved in the research of high-speed ATM switching system. Currently, he is engaged in the research and development of the Media Gateway Controller (MGC) in the integrated PSTN-IP network at NTT Network Service Systems Laboratories. He is a member of IEEE Communication Society.

Masanori Hirano received the B.E. and M.E. degrees in engineering and electronics from Kumamoto University in 1973 and 1975, respectively. He entered the Telecommunications Laboratories of NTT in 1975. He has been involved in research and development of processor architecture, hardware systems and service control nodes for the Intelligent Network. He became a professor at Tokyo University of Information Sciences in 2001. He holds a doctorate in engineering, and is a member of the Information Processing Society of Japan (IPSJ) and IEEE.