Fault Recovery Performance Analysis of Functionally Distributed Transport Networking System

Kentaro Ogawa, Kenichi Higuchi, and Shinichiro Chaki

Abstract. We propose a fault recovery method in functionally distributed transport networking that separates the control-plane processing part (control element, CE) from the forwarding-plane processing part (forwarding element, FE) of the router. In this architecture, one path-control process in the CE consolidates and processes the path computations and the path settings for multiple FEs. This leads to reduction in the path-control complexity and efficient operation of large scale networks. On the other hand, it is absolutely critical to ensure the high reliability of the CE. We analyze the performance of the proposed fault recovery method by using software implementation.

Keywords: fault recovery; performance analysis; functionally distributed transport networking; redundant configuration; router.

1 Introduction

Traffic in IP networks has increased and services have diversified over the last few years. IP networks need to have scalability and high-quality data transmissions, and they must be able to incorporate new functions. The progress of large-scale IP networks has increased the complexity of the path control and has required the use of increasingly sophisticated routers. For instance, a high-performance processor and a huge memory are now required for the edge router

Kentaro Ogawa · Kenichi Higuchi · Shinichiro Chaki
Network Service Systems Laboratories,
NTT Corporation,
3-9-11, Midori-cho, Musashino-shi,
Tokyo, 180-8585 Japan
e-mail:{ogawa.kentaro,higuchi.kenichi,
chaki.shinichiro}@lab.ntt.co.jp
http://www.ntt.co.jp/islab/e/org/ns.html

S. Caballé, F. Xhafa & A. Abraham (Eds.): INCoS 2009, SCI 329, pp. 281–295.
springerlink.com

that supports VPNs and multicasts because routing protocols, such as open shortest path first (OSPF), border gateway protocol (BGP), and protocol independent multicast sparse mode (PIM-SM), operate at the same time. The interior gateway protocol (IGP) used for path control manages the topology of the network according to the link state database (LSDB) and computes the paths. An increase in the number of routers in a network enlarges the LSDB and lengthens the path computational time, which limits the scale of a network. In large scale network composed of hundreds of thousands of routers, the routing information is consolidated based on network domains within a defined area, and coordination between exterior gateway protocols (EGP) and IGPs for path control is needed. The IGP controls the path in the network domains, and the EGP controls the path between the network domains. As a result, routers must be set according to the network design, which increases maintenance operations. Moreover, the addition of functions to the transport stratum entails the addition of functions to an individual router. Because large scale networks have many and various routers, it takes a long time to provide new network services.

To overcome these problems, the softrouter architecture, with a control-plane processing server for high processing performance and ease of adding functions to the transport stratum, has been proposed [1]. This architecture separates the control-plane processing part (control element, CE) from the forwarding-plane processing part (forwarding element, FE) of the router. In this architecture, a CE is mounted on a general-purpose server platform, which becomes a control-plane-processing server, and one path-control process in the CE consolidates and processes the path computation and path setting for one or more routers, which act as FEs. The FEs are bound to an arbitrary server via an IP session. An FE collects topology information from adjacent FEs and sends it to a CE. The CE computes the path and sets up a routing table for the connected FEs. The performance and functions of the CE can be easily upgraded in this architecture. However, the softrouter cannot solve the complexity of the path control due to the coordination between IGPs and EGPs.

To reduce the complexity of the path control in large scale networks, we have proposed the functionally distributed transport networking architecture [2, 3] by expanding the softrouter concept. This architecture configures a functionally distributed router (FDR) with one CE and multiple FEs as well as the softrouter, and recursively configures the entire network with a higher CE and all of the FDRs. The higher CE treats each FDR as one router and computes the path for the entire network by using the same path-control method as one for the FDR. This leads to reduction in the path-control complexity and efficient operation of large scale networks.

As for a related work, OpenFlow [4] adopts the concept of the separation between the control-plane and the forwarding-plane. OpenFlow can also control multiple forwarding elements, called OpenFlow Switch, by a controller. However, OpenFlow needs the specific hardware for an OpenFlow Switch and the specific OpenFlow protocol for the communication between a controller and OpenFlow Switches.

The above studies have not considered the high reliability of the system from the perspective of actual operation. Under the architecture in which one CE consolidates all of connected FEs, if faults occur in a CE and the CE become unable to serve a routing function, all of the FEs controlled by the CE will be affected. FEs can continue to serve the data packet transmission during the fault of their CE because the FEs are physically-separated from the CE in this architecture. However, the FEs cannot receive the path computation and path setting from their CE. That is, FEs can no longer respond to changing situations of the network topology such as link-up/down of any ports, links, and FEs. Therefore, it is absolutely critical to ensure the high reliability of the CE in this architecture.

In this paper, we propose a fault recovery method in functionally distributed transport networking. This method responds to CE fault by taking the redundant configuration of N+m CEs. This means that network operators provide m standby CEs for the control-plane consisting of N active CEs. If faults occur in any of the active CEs, the operation of the failed CE will be continued in any of the standby CEs.

The paper is structured as follows: Section 2 gives an overview of functionally distributed transport networking. Section 3 proposes a fault recovery method of CEs. Section 4 explains how to implement a functionally distributed transport networking system and apply the proposed fault recovery method. Section 5 analyzes the fault recovery performance by using the implemented system in a simulation environment and evaluates its feasibility. Section 6 concludes the paper.

2 Overview of Functionally Distributed Transport Networking

2.1 Routing Framework

This section explains the basic framework of the functionally distributed transport networking architecture. Fig. 1 shows the configuration of an FDR in which a CE is physically-separated from an FE. In this model, one or more FEs connect to one CE. The CE controls all connected FEs via control sessions. The control sessions between the CE and the FEs are established through the management channels. There is no limit on the number of hops of the control session.

The router functions are distributed among the CE and the FEs.

1. The CE performs functions that:

- collect and manage the topology information of the network between the FEs and construct the LSDB.
- exchange the LSDB with other CEs that control other FE groups via the higher CE.
- compute the routing information base (RIB).
- generate the forwarding information base (FIB) of each connected FE from the RIB.
- send each FIB to the corresponding FE via the control session.

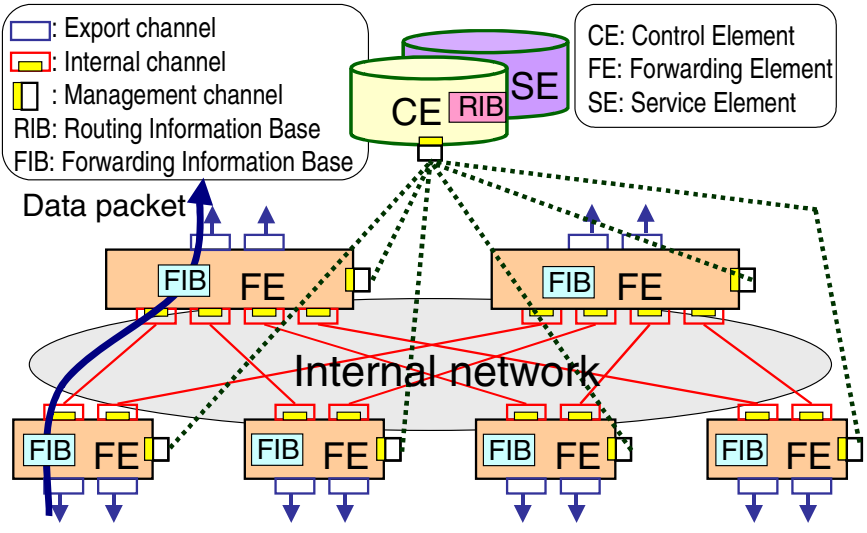


Fig. 1 Configuration of functionally distributed router

2. The FE performs functions that:

- collect topology information about the network between adjacent FEs.
- send topology information to the CE.
- forward the data packets based on the FIB.

2.2 Expansion of Performance and Functions in Transport Stratum

In functionally distributed transport networking architecture, the control-plane and the forwarding-plane are physically-separated, and the interface between those planes is specified. Therefore, performance and functions of each plane can be developed independently.

This architecture can also include a service element (SE) (see Fig. 1). This is an element in which the service processing (for example, security, monitoring, and access control) is executed on demand from the CE. The SE may be mounted on a control-plane processing server running any of CEs.

The number and capacities of CEs and SEs in the network can be increased according to the demand for processing performance. In addition, CEs can dynamically change the correspondence with the FEs in accordance with the load balancing policy.

3 Fault Recovery of CEs

It is important to ensure the high reliability of the CE in the architecture in which one CE consolidates and processes the path computations and the path settings for multiple FEs. Generally, current networks provide standby routers to respond to faults of active routers. In many cases, Virtual Router Redundancy Protocol (VRRP) [5] is adopted into the routers as a standardized protocol to switch from a failed router to a standby router. The standby router may be active for other VRRP group with a different VRID. Although VRRP switches automatically once a fault is detected on an active router, an active router and its standby routers have to be connected at the same LAN. This means that there is a restriction in which an active router connects to standby routers via one hop.

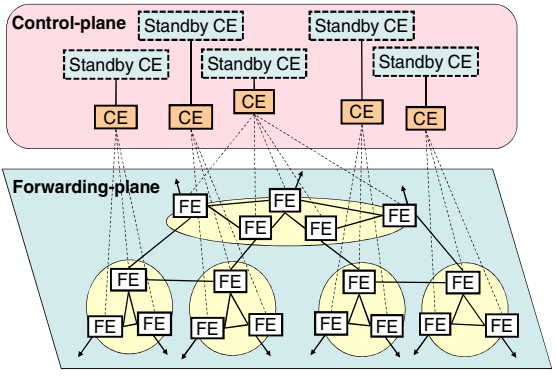
To respond to CE failover in functionally distributed transport networking architecture, we propose a CE failover method under the CE redundant configuration like VRRP. However, from the aspect of widely distributed elements in this architecture, the restriction on the number of hops between an active CE and a standby CE must be lifted.

3.1 CE Redundant Configuration

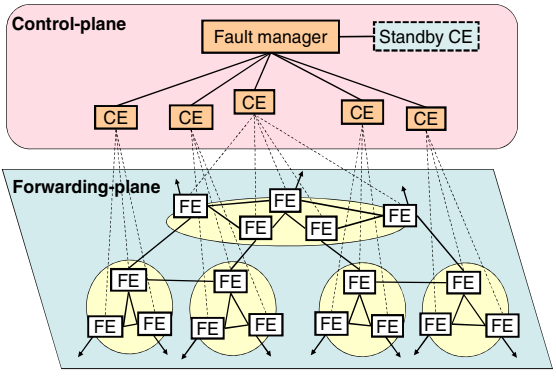
When network operators configure the control-plane with redundant CEs in their networks, they have to decide how many redundant CEs they need to provide and how to make a correlation between active CEs and standby CEs. There are several possible solutions depending on the requirements from the perspective of system reliability, configuration flexibility, and economical efficiency.

3.1.1 1+1 Redundant Configuration

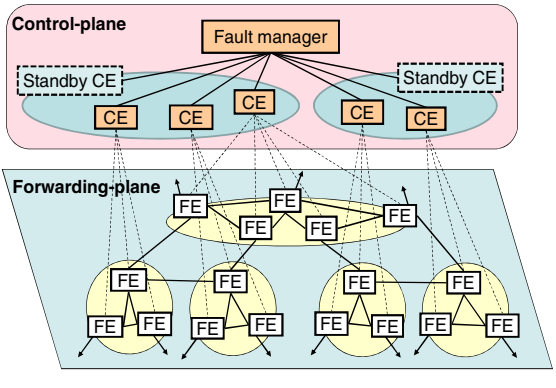
1+1 redundant configuration is the simplest solution whereby each active CE has its own standby CE to ensure that CE functionality continues in the event of CE fault (see Fig. 2-(a)). Each active CE sends heartbeat signals to its own standby CE. When the standby CE no longer receives the heartbeat signals from the active CE, the standby CE judges the active CE to be in fault and takes over the operation as a new active CE. This configuration can achieve very high reliability. Because each subset of an active CE and a standby CE can operate failover process independently, the overall system integrity will not be impacted even when multiple active CEs break down at the same time. However, network operators have to prepare twice the number of CEs needed for primary operation. This means that this configuration is not economically efficient.



(a) 1+1 redundant configuration.



(b) N+1 redundant configuration.



(c) N+m redundant configuration.

Fig. 2 Redundant configurations

3.1.2 N+1 Redundant Configuration

N+1 redundant configuration is a solution in which multiple active CEs (N) have one independent standby CE to ensure that CE functionality continues in the event of CE fault (see Fig. 2-(b)). In this configuration, the system has a fault manager in the control-plane. The fault manager is connected with all of the active CEs and the standby CE. Each active CE sends heartbeat signals to the fault manager. When the fault manager no longer receives the heartbeat signals from any of the active CEs, the fault manager judges that active CE to be in fault and activates the standby CE as a new active CE. When the failed CE recovers, it connects to the fault manager as a new standby CE. Because only one standby CE is shared by all of the active CEs, network operators can keep extra expenses down. On the other hand, system reliability lowers. There is no standby CE during the recovery period of a failed CE. That is, the overall system integrity will be impacted when multiple active CEs break down at the same time.

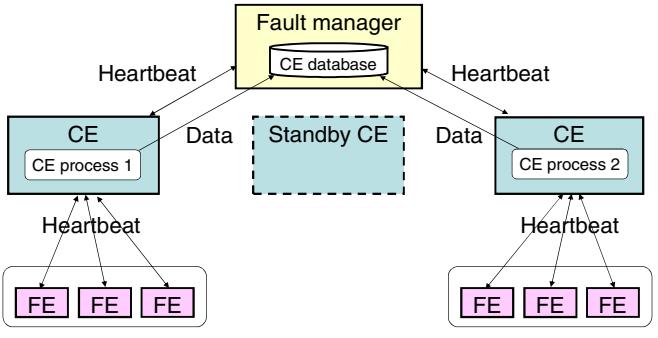
3.1.3 N+M Redundant Configuration

N+m redundant configuration is a solution in which multiple active CEs (N) have multiple independent standby CEs (m) to ensure that CE functionality continues in the event of CE fault (see Fig. 2-(c)). All active CEs are divided into several groups (m), and each CE group is related to its own standby CE. In this configuration, the system has a fault manager in the control-plane. The fault manager is connected with all of the active CEs and the standby CEs. Each active CE sends heartbeat signals to the fault manager. When the fault manager no longer receives the heartbeat signals from any of the active CEs, the fault manager judges that active CE to be in fault and activates the standby CE related to the group with the failed CE as a new active CE. When the failed CE recovers, it connects to the fault manager as a new standby CE for its group. This configuration results in a highly flexible redundant configuration. Network operators are free to decide the number of standby CEs in the control-plane depending on such requirements as CE fault rate and budget for deployment. If m is set to N, the 1+1 redundant configuration can be realized. If m is set to 1, the N+1 redundant configuration can be realized. Because of its flexibility, we propose to adopt this configuration into our CE failover method.

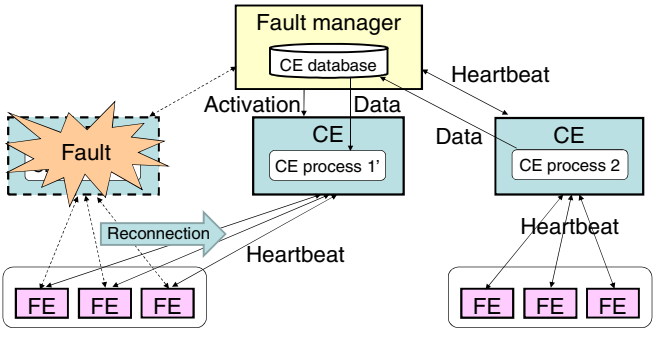
In this case, since there is a single fault manager that deals with all CEs, the fault manager is itself a single point of fault. This method requires full duplication of the fault manager for the robust system. A fault manager has an active body and a standby body and always runs status mirroring and data mirroring between two bodies. Also, links between a fault manager and CEs are duplicated. Because there is only one fault manager in whole system, its full duplication has little influence on economical efficiency. This paper supposes that a fault manager is guaranteed its correct operation, and focuses on CE faults.

3.2 Failover Operation

This section describes detailed failover operation under the N+m redundant configuration. Failover operation is attained with a combination of steady-state (see Fig. 3-(a)) and fault-state phases (see Fig. 3-(b)).



(a) Steady-state phase.



(b) Fault-state phase.

Fig. 3 Failover operation

In each phase, each component of the network, such as CE, FE and fault manager, plays an individual role. Unlike the popular distributed system such as the blade server which has all elements in one chassis, CEs are separated from FEs via a network in the proposed architecture. Therefore, separately from monitoring and switching CEs by a fault manager, FEs have to monitor the connected CE and reconnect to the new active CE. Since FEs move independently from a fault manager, the FE operation and the fault manager operation have to be integrated into one failover operation.

3.2.1 Steady-State Phase

The steady-state phase is a phase in which the system runs normally for routing and forwarding. In this phase, an active CE sends heartbeat signals to the fault manager and all of the FEs connected to that CE and uploads CE's control data including a list of connected FEs, LSDB and RIB to the CE database on the fault manager. The data is uploaded repeatedly in conjunction with data updates by the CEs to keep the CE database up-to-date. The fault manager monitors the health of the CEs by receiving heartbeat signals and stores the data from the CEs in the CE database. The fault manager also stores a table, which shows the correspondences between a group of several active CEs and a standby CE. The table is set to the fault manager by network operators. An FE monitors the health of the connected CE by receiving heartbeat signals as well as the fault manager. All the FEs are set to an address of the standby CE, to which they should reconnect in case they detect CE fault, by network operators.

3.2.2 Fault-State Phase

The fault-state phase is a phase in which the system recovers its incompleteness due to the CE fault by switching from the failed CE to the corresponding standby CE. When faults occur in any of the active CEs, the system goes into this phase. Due to breakdown of the heartbeat signals from the failed CE, the fault manager and FEs connected to that CE detect the CE fault. Then, the fault manager activates the standby CE related to the group with that failed CE as a new active CE and sets the data of the failed CE stored in the CE database to the new active CE. This enables the new active CE to quickly take over the operation. All of the FEs connected to the failed CE try to reconnect to the new active CE. As it may take some time before the new active CE completes preparations for the reconnection, FEs try to reconnect periodically. When the reconnections from all of the FEs are over, the new active CE collects the topology information of the network between the reconnected FEs and recomputes the LSDB, RIB and FIB of each reconnected FE. This enables the system to respond to changing situations in the network topology during a recovery period of the failed CE.

This operation assumes a basic "fail-stop" model for CE fault. In case that only the link between a fault manager and a CE has failed, the communication can recover quickly by switching to a standby link because links between a fault manager and CEs are duplicated. The same handling can be used for the case that only the link between a CE and an FE has failed.

4 System Implementation

To analyze the fault recovery performance based on the proposed method, we implemented a functionally distributed transport networking system and applied the proposed fault recovery method to the system. In this section, we explain our ways of implementing the system and applying the proposed method to the system.

4.1 Functionally Distributed Transport Networking System Implementation

4.1.1 XORP Retrofit

We implemented a functionally distributed transport networking system based on XORP [6, 7], which is a modular, extensible, open source IP routing software. XORP provides a fully featured platform that implements IPv4 and IPv6 routing protocols and a unified platform to configure them. Due to its modularity, we can easily add new functions. We retrofitted XORPv1.3 as follows.

1. To allow physical separation between a CE and an FE, we divided XORP functional modules into CE side and FE side, and packaged each of the module groups as a CE software and an FE software.
2. We used the ForCES [8, 9] protocol, which is specified by the Internet Engineering Task Force (IETF), as the communication interface between management channels of a CE and FEs. ForCES is a protocol by which CEs can configure and manage one or more separate FEs within a network element. There is no limit on the number of hops of the ForCES session between a CE and an FE. To apply the ForCES protocol, we added a ForCES communication management function to the CE and FE software.
3. Original XORP can control only one forwarding engine. We modified the CE software to control multiple FEs at a time. This enabled a CE to establish and maintain multiple ForCES sessions with each of FEs, manage and configure status of the connected FEs, and compute RIBs and FIBs of the connected FEs.
4. We added a higher CE mode to the CE software. The higher CE mode is used on a higher CE which treats each FDR composed of one CE and multiple FEs as one FE and computes optimal paths among the FDRs. A higher CE communicates with all of the CEs using XORP's extensible IPC mechanism called XORP Resource Locators (XRL).
5. We made a path computation module using the Shortest Path First (SPF) algorithm. This module can compute RIBs of all FEs based on an LSDB and generate FIBs of all FEs from the RIBs. Both CE and high CE recursively use the same path computation module to reduce the complexity of the path control.

4.1.2 Functional Architecture

Fig. 4 illustrates the functional architecture of our implementation. Higher CE, CE and FE have some function modules as follows.

1. The Higher CE has:

- CE management module which manages the information of the connected CEs, such as a CEID and an IP address.
- path computation module which receives the path information among all of the FDRs, maintains the LSDB, computes RIBs of all of the FDRs, and distributes the computed optimal path results to the corresponding FDRs.

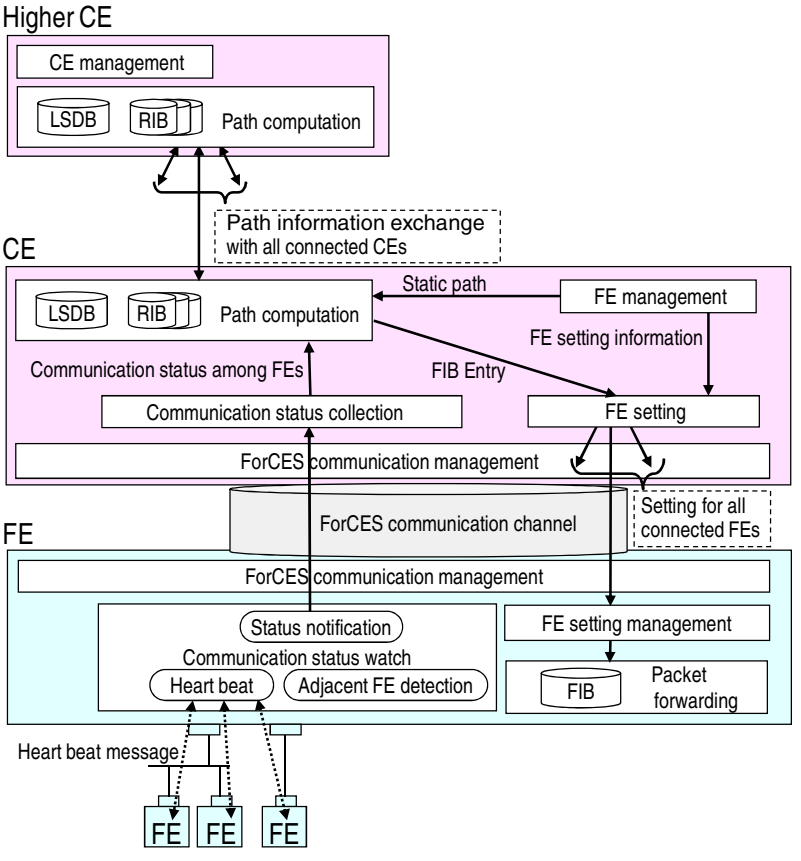


Fig. 4 Functional architecture of the implementation

- LSDB which is composed of the link state information among all of the FDRs, such as a network address and a metric.
- RIB of each FDR which is computed from the LSDB using the SPF algorithm.

2. The CE has:

- FE management module which manages the information of the connected FEs, such as an FEID and an IP address, passes the FE setting information to the FE setting module, and if any, passes the static path information among the FEs to the path computation module.
- FE setting module which receives the FE setting information from the FE management module and the FIB entries from the path computation module, and passes the information to the FE setting management module on the corresponding FEs via ForCES communication channels.
- path computation module which receives the path information among all of the connected FEs from the FE management module and the communication status

collection module, maintains the LSDB, computes RIBs of all of the connected FEs, generates FIBs of all of the connected FEs, and passes the generated FIB entries to the FE setting module.
- LSDB which is composed of the link state information among all of the connected FEs, such as a network address and a metric.
- RIB of each connected FE which is computed from the LSDB using the SPF algorithm.
- communication status collection module which receives the communication status among FEs including the link state information from the communication status watch module on each FE via ForCES communication channel, and passes the information to the path computation module.
- ForCES communication management module which establishes and maintains the ForCES communication channels between a CE and each connected FE, encapsulates a message between a CE and each connected FE with the ForCES massage format, and exchange the encapsulated massage via the ForCES communication channel.

3. The FE has:

- FE setting management module which receives the FE setting information and the FIB entries from the FE setting module on the connected CE, reflects the FE setting information to the FE operation, and passes the FIB entries to the packet forwarding module.
- packet forwarding module which receives the FIB entries from the FE setting management module, stores the FIB entries on the FIB, and forwards the data packets based on the FIB.
- FIB which is composed of the FIB entries set by the FE setting module on the connected CE.
- communication status watch module which detects adjacent FEs, monitors the communication status with the adjacent FEs by exchanging heartbeat massages, and notifies the communication status collection module on the connected CE of the latest communication status.
- ForCES communication management module which is same as that of a CE.

4.2 Application of Fault Recovery Method

We applied the proposed fault recover method to the implemented system described above. To allow the implemented system to operate as shown in fig.3, we added the functions for the failover operation as follows.

1. We added the fault manager module to the higher CE. The fault manager exchanges heartbeat signals with all of the active CE, maintains the CE database, manages the correspondences between a group of active CEs and a standby CE, activates the appropriate standby CE when faults occur in any of the active CEs, and sets the data of the failed CE stored in the CE database to the new active CE.

2. We added the CE high availability module to the CE. The CE high availability module exchanges heartbeat signals with the higher CE and all of the connected FEs, and sends CE's control data to the fault manager module on the high CE.
3. We added the FE high availability module to the FE. The FE high availability module exchanges heartbeat signals with the connected CE, stores an address of the standby CE set by network operators, and reconnects to the new active CE when faults occur in the connected CE.

5 Fault Recovery Performance Analysis

Under the architecture in which one CE consolidates all connected FEs, if faults occur in a CE and it becomes unable to serve a routing function, all of the FEs controlled by that CE cannot receive the path computation and path setting from their CE. That is, FEs can no longer respond to changing situations in the network topology such as link-up/down of any ports, links, and FEs. Therefore, CE failover operation must be completed as quickly as possible.

To evaluate the feasibility of the proposed fault recovery method, we determined the time of CE failover operation by using our implementation in a simulation environment. In the simulation environment, we used two servers with two dual-core Xeon (2.13 GHz) processors for an active and a standby CEs. We prepared two more servers for a higher CE and FEs. The FE server can run multiple FE processes at the same time, and each FE process can establish an individual association to the CE process. All of the servers in the simulation environment run on the CentOS 5 [10] operating system, which is a Linux distribution derived from sources provided free to the public. The bandwidth of all links between the servers is 1.0 Gbps. Since the bandwidth is sufficiently-broad to exchange the massages between servers, there is no bottleneck link.

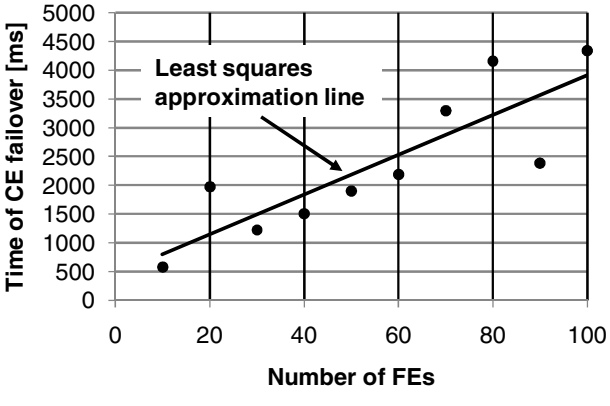


Fig. 5 CE-failover time

The simulation is supposed to start when the fault manager module on the higher CE detects the fault of the active CE. This detection induces the activation of the standby CE and setting the data of the failed CE stored in the CE database to the new active CE. Then, all of the FEs reconnect to the new active CE. The new active CE processes the ForCES association setup for all of the FEs. As for time of CE failover, we determined the time from CE fault detection by the fault manager module to the completion of the reconnection by all the FEs. Additionally, the simulation was carried out on various network scales. There were ten network topologies wherein the number of FEs changed from 10 to 100 in steps of 10. As each FE has 10 links, the number of links in whole system changed from 100 to 1000 in steps of 100.

Fig. 5 shows the results from the CE failover. This scatter diagram plots CE failover on various network scales, and an approximation line was made using the least squares method. Because the new active CE has to process the requests of the reconnection equaling the number of FEs, the total time increases dramatically with the number of FEs. Additionally, an increase in the number of FEs means increased possibility of FE reconnection fault and retry. Several reconnection requests at once ran out of time. This leads to an increase in the total processing time. However, the graph is not in perfect incremental shape. This is attributed to the timers set in the algorithm. We set the interval parameters of the CE fault detection and the reconnection retry by an FE to 1.0 s. It would appear that those timers caused the variation in CE-failover time. Even if the number of FEs was 100, CE-failover time would be less than 5 s. Given that the initial value of RouterDeadInterval in OSPFv2 [11] is 40 s, the CE-failover time is short enough not to be recognized as a fault by neighbor routers. Therefore, we can say that the proposed method has adequate feasibility.

For a further study, an experimental evaluation of the feasibility and scalability of the proposed method in real environments is important issue.

6 Conclusion

We proposed a fault recovery method in functionally distributed transport networking, which leads to reduction in the path-control complexity and efficient operation of large scale networks. Under the architecture in which one CE consolidates all connected FEs, if faults occur in a CE and the CE become unable to serve a routing function, all the FEs controlled by that CE cannot receive the path computation and path setting from their CE. Therefore, it is absolutely critical to ensure the high reliability of the CE in this architecture. Our proposed method responds to CE faults by taking the redundant configuration of N+m CEs. This means that network operators provide m standby CEs for the control-plane consisting of N active CEs. If faults occur in any of the active CEs, the failed CE function will be continued in any of the standby CEs. Additionally, we described the operation of each component in the proposed method, explained how to implement a functionally distributed transport networking system and apply the

proposed fault recovery method, and analyzed the fault recovery performance by using the implemented system in a simulation environment. We confirmed that the time of CE failover is short enough and the proposed fault recovery method has adequate feasibility.

Acknowledgments. This work was supported in part by the National Institute of Information and Communications Technology (NICT) of Japan.

References

1. Lakshman, T.V., Nandagopal, T., Ramjee, R., Sabnani, K., Woo, T.: The Sof-trouter Architecture. In: ACM HotNets-III Workshop (2004)
2. Aoki, M., Ogawa, K., Hamano, T., Chaki, S.: Functionally Distributed Trans-port Networking on Next-Generation Network. In: IEEE-WS NGN-EC2, pp. 593–600 (2007)
3. Ogawa, K., Aoki, M., Chaki, S.: Verification of Path Computational Perform-ance in Functionally Distributed Transport Networking on Next-Generation Network. In: IEICE APSITT 2008, pp. A-1-2 (2008)
4. OpenFlow, `http://www.openflowswitch.org/wp/documents/`
5. Hinden, R.: Virtual Router Redundancy Protocol (VRRP). Internet Engineering Task Force, RFC3768, `http://ftp.ietf.org/rfc/rfc3768.txt`
6. Handley, M., Hodson, O., Kohler, E.: XORP: An Open Platform for Network Research. ACM SIGCOMM, Computer Communication Review 33(1), 53–57 (2003)
7. XORP Project, `http://www.xorp.org`
8. Khosravi, H., Anderson, T.: Requirements for Separation of IP Control and Forwarding. Internet Engineering Task Force, RFC3654, `ftp://ftp.ietf.org/rfc/rfc3654.txt`
9. Yang, L., Dantu, R., Anderson, T.: Forwarding and Control Element Separation (ForCES) Framework. Internet Engineering Task Force, RFC3764, `ftp://ftp.ietf.org/rfc/rfc3746.txt`
10. The Community ENTerprise Operating System, `http://centos.org`
11. Moy, J.: OSPF Version 2. Internet Engineering Task Force, RFC2328, `ftp://ftp.ietf.org/rfc/rfc2328.txt`