International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:5, No:11, 2011

CNet Module Design of IMCS

Youkyung Park, SeungYup Kang, SungHo Kim and SimKyun Yook

***Abstract*—**IMCS is Integrated Monitoring and Control System for thermal power plant. This system consists of mainly two parts; controllers and OIS (Operator Interface System). These two parts are connected by Ethernet-based communication. The controller side of communication is managed by CNet module and OIS side is managed by data server of OIS. CNet module sends the data of controller to data server and receives commend data from data server. To minimizes or balance the load of data server, this module buffers data created by controller at every cycle and send buffered data to data server on request of data server. For multiple data server, this module manages the connection line with each data server and response for each request from multiple data server. CNet module is included in each controller of redundant system. When controller fail-over happens on redundant system, this module can provide data of controller to data sever without loss. This paper presents three main features – separation of get task, usage of ring buffer and monitoring communication status – of CNet module to carry out these functions.

***Keywords*—**Ethernet communication, DCS, power plant, ring buffer, data integrity

I. Introduction

THIS paper describes Ethernet communication method for IMCS(Integrated Monitoring and Control System for thermal power plant). CNet module (Ethernet communication module of IMCS) is designed to exchange data between controller of IMCS and OIS. CNet module gathers and stores data result of logic execution. When data server of OIS sends data request message, CNet module sends stored data. Also, CNet module receives control command from OIS for the controller.

To exchange data, the most important thing is data integrity. To guarantee data integrity on any situation, CNet module separates Get-task that buffers and manages data from the other tasks. Get-task synchronizes with logic performing period and grabs every data. CNet module uses and manages ring buffers to store every data without any loss. Also, CNet module monitors every opened communication line and socket.

In this paper, design concepts of CNet module and details for ensuring efficient data exchange are described.

II. Ethernet Communication Concept of IMCS

Ethernet communication method is used in data exchange between controllers and OIS [1]. On the side of a controller, CNet module is responsible for data communication. On the other side – OIS – data server is responsible for data communication with CNet module. A network between controllers and data servers refers to CNet (Control Network) and another network between data servers and HMI (Human Man Interface) refers to OIS Net. All Ethernet networks are duplicated in a way of cold sparing standby [3]. So if there is any problem in a primary network, other normal network takes primary-ship.

Each controller and data server is redundant by duplicated or TMR (Triple Modular Redundancy) depending on situations [2]. The method of redundant is a way of hot sparing standby. If there is any problem on a master controller then a fail-safe happens.

A data server stores acquired data from CNet module, sends a control command from HMI to CNet module, shows real-time trend on HMI screen and preserves data to historian data server without redundancy. A communication module of data server refers to Gateway. In other words, Ethernet communication of CNet on IMCS is performed between CNet modules and Gateway. The HMI real-time trend is represented by data from data server. When user wants to check past data, data server will access the historian data server.

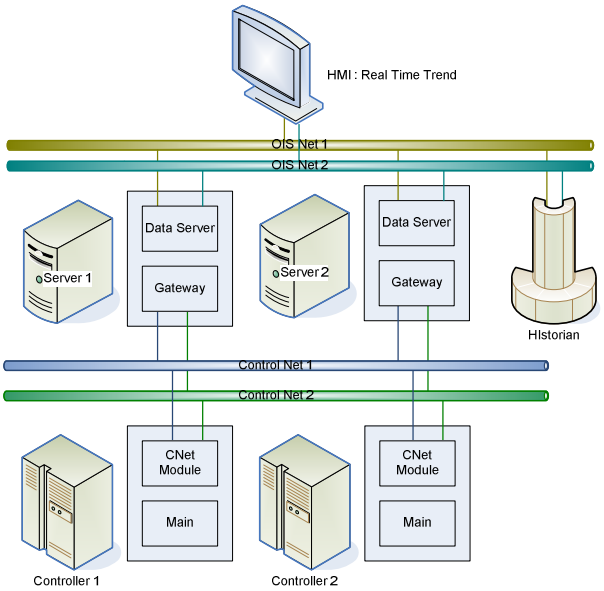

Fig. 1 Ethernet communication concept of IMCS

III. Characteristics of CNet

CNet module is designed to meet these conditions; does not miss any important data, makes minimum load of each module and uses network efficiently.

A. Request & Reply

To exchange data within Ethernet communication, a method of Request and Reply is accepted. The method is that a controller stores every created data from result of logic execution and sends the data when Gateway requests data.

Youkyung Park is with Doosan Heavy Industries and Construction, Deajeon, Republic of KOREA (phone: +82-42-712-2211; fax: +82-42-712--2230; e-mail: youkyung.park@ doosan.com).

SeungYup Kang, SungHo Kim and SimKyun Yook are with Doosan Heavy Industries and Construction, Daejeon, Republic of KOREA

Manuscript received October 31, 2011.

If the controller sends data to Gateway unilaterally, the load of Gateway will be increase. However, request and reply algorithm does not increase unnecessarily loads on the data server, so the result has the advantage of facilitating data communications.

Gateway of every data server sends data request messages to every controller which is connected with the Gateway. When the data request messages reach to CNet module, CNet module once judge whether the controller is master or slave. If the controller is master, CNet module sends all stored data to Gateway. Or if the controller is slave, CNet module does not send stored data but sends an 'IsNotMaster' message as response. Gateway just ignores the IsNotMaster message, without taking additional actions cause of a status of controllers – a master or a slave. In this case, data exchange process is able to be standardized that Gateway performs same task, regardless of the controller switching.

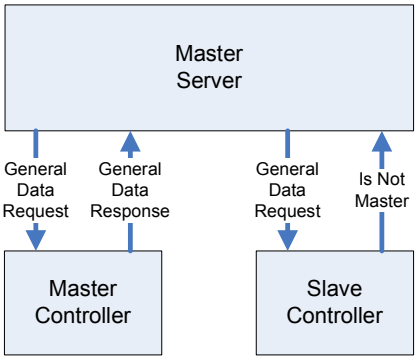

Fig. 2 Structure of Request and Reply

B. Support for Hot Standby of Gateway

On a data communication between CNet module and Gateway, the initially design sets for only a master Gateway to send data request messages to the CNet module. In this case, the Gateway which is switching to a new master has no previous data. As a result, a real-time trend showing is impossible as well as there is a possibility of losing data when switching of Gateway. To fix this problem, change standby sparing way of Gateway from cold standby sparing (one of Gateway is active and the other is idle) to hot standby sparing (every Gateway are active and one of them send data to HMI)[3].

Duplicated Gateway send data request messages to CNet modules which are connected to the Gateway, regardless of the state which is a master or a slave. CNet module receives the data request message from all Gateways and transmits data in the predefined form, depending on the state of a controller – master or slave. Each server that receives data stores the data in the internal DB and master servers communicate with the upper OIS. As such, regardless of the status of Gateway and controllers, Gateway performs the same role and is standardized.

C. Master/ Slave Change of controllers

In general, general data is generated from controller by the unit of kB (kByte) and saved changed at every 100msec (milli – second). Therefore, to prevent the loss of these general data, data buffer set of the GB (giga byte) unit in size. Modern technology, there is no problem to support these buffer capacity. Just a data server communicates with multiple controllers and handles many data in a short time. So when the data server receives large data in a short time, the processing to store DB (data base) can experience an increase in load. In addition, when fail-over of controller happens, Gateway receives from new-master controller same data that had previously received from past-master controller and eventually it leads to unnecessary increased load on the network.

In order to efficient utilization of servers and network, it is designed to change a buffer size depending on the state of controller – master or slave. When a controller is master mode, CNet module sets data buffer size normally, and when the controller is slave mode, CNet module sets data buffer size smaller than buffer size of the master-mode. As like this, even if the fail-over of controller happens, it does not happen for Gateway to receive suddenly huge and ruinous data. The buffer size of the slave-mode controller is set as well as it can hold created data during fail-over of controllers, so a possibility of any data loss is avoided.

D. Kinds of CNet Data

There are 3 kinds of data in CNet module: general data, SOE (Sequence Of Event) data and diagnostic data. General data is associated with a logic execution of controller, saved in general data ring buffer of CNet module and sent to Gateway when Gateway requests the general data. SOE data is pre-set to 1ms resolution to save on EWS. A number of SOE is smaller than a number of general data. The SOE data is gathered from each IO board and saved in a ring buffer and sent to Gateway when Gateway requests SOE data. The diagnostic data is a collection of status and diagnostic information for each board of controller. For the diagnostic data, CNet module sends the data when Gateway requests diagnostic data without buffering. CNet module classified data into three categories depending on characteristics of data so increases an efficiency data communication.

TABLE I
KIND OF DATA FOR CNET MODULE

Name of data	PERIOD OF BUFFERING	Description
General data	multiple of logic period (decision on configuration)	Input and output data related to control logic Digital data and analog data Input data and output data Classified according to the size of the data
SOE data	Basic logic time	Specified and classified by EWS
Diagnostic data	N/A	Specified and classified data by EWS to diagnostic data

General data and SOE data is gathered and stored in Get-task. General data is handled in a digital and an analog type. General data is obtained according to the data acquisition cycle in the EWS configuration and SOE data is obtained depending on the period of basic logic cycle.

In general data, the data sent from OIS typically referred to as control commands. When CNet module receives a control command, CNet module applies the control command immediately in OIS memory to recognize for logic. After applying the control command, CNet module sends an ACK message to Gateway in order to verify whether the applying process of a control command is well or not. Because CNet module sends all data of buffer to Gateway, user can check changed data by applying the control command. Therefore other check message according to apply the control command are designed not to generate.

IV. Separate Get-Task

In CNet module, on the initial design, data gathering and communication with Gateway are treated in one task. However in the result of this design approach, if there is any problem in the task then the task can't handle communication too. Therefore, there is a possibility of data loss – during that fail-over time, the data may be missing. To avoid this situation, Get-task is separated from the task of CNet module and performed the end of logic execution.

Tasks of CNet module are separated in 4 kinds in order to manage effective CNet module. As a result, any other problem in one task does not affect other task. Fig. 3 shows to overview of CNet module action and describes Get-task and Comm-task. Data which are created and managed from controller are saved to an OIS memory. Get-task is called after logic executing, gathers data from OIS memory and saves data to data ring buffer of CNet module. Comm-task charges of communication with Gateway. Comm-task is designed to receive messages from every Gateway and analyze the messages in turn. If the analyzed message is data request message then Comm-task accesses to the data ring buffer and sends every data to Gateway which requests the data.

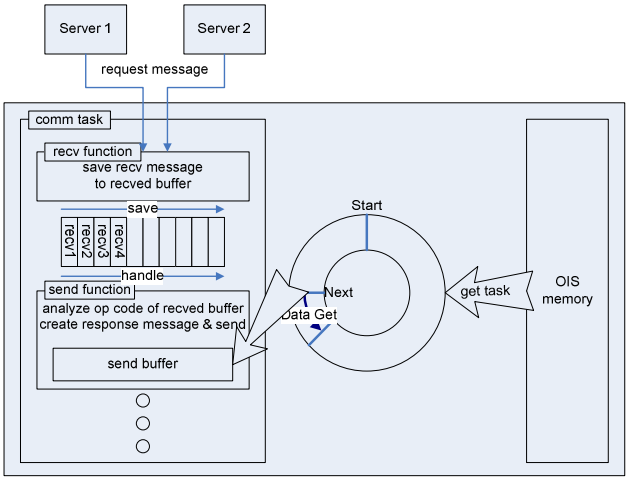

Fig. 3 Outline of managing data of CNet

Because Comm-task and Get-task can access to the data ring buffer at the same time, a binary semaphore is set for the ring buffer. As a result of using the semaphore, during one task access to the ring buffer, the other task wait that the semaphore free as a result of ending access by the task. It is blocked that different task access the same ring buffer at the same time, so the possibility of errors for data is blocked.

V. Ring Buffer

The basic communication concept of CNet module is the request and reply way. The communication way has a problem that it is possible to miss data when data buffer is not enough large or Gateway doesn't request message during some time by any reason. And also, insertions and extractions of data are repeated frequently. If the data buffer is overflow unavoidably, the most recent occurred data have a higher priority than existing data. In order to use limited resource and grab every created data without loss, a concept of ring buffer is accepted.

A. Concept of ring buffer

A ring buffer uses a continuous memory capacity physically but works in a donut-shaped circle conceptually. On the data characteristics of CNet module, in order to avoid infringing the original algorithm that copies the memory by data header, a 'null pointer' is devised to point a memory area which does not use in especial case.

Fig.4 is shows a concept of ring buffer managing. There is a ring buffer manager that saves pointers which point specific locations of the ring buffer. As like the Fig.4, for an example, there is a memory buffer that can save 20 data. And at one time, 7 data are created, gathered and saved. At the first, number 1 shows that 7 data is saved in the memory buffer. When Gateway sends data request message, CNet module sends to Gateway all data in the ring buffer. And then, 7 data is created again. At that time, the ring buffer manager saved a last address which is sent to Gateway last time, and saves data from the next address which is saved. As a result, a status of the ring buffer is as like as number 3 of Fig.4. And then the ring buffer manager sends data to Gateway and gets data again, the ring buffer is structured as like number 5 of Fig.4.

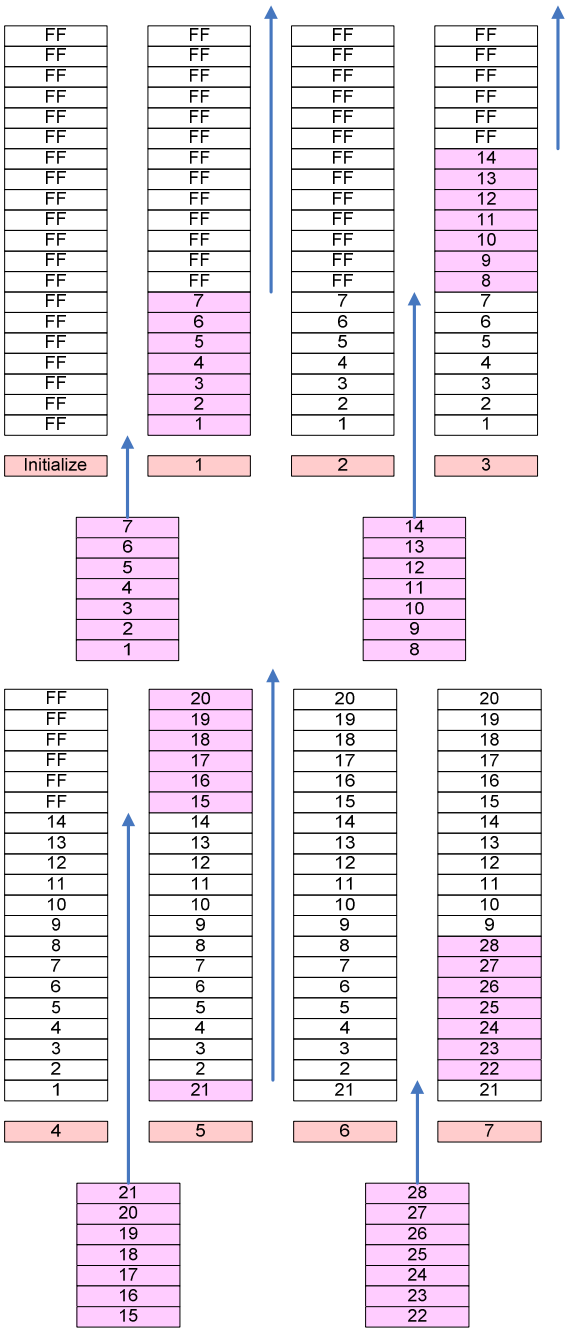

Fig. 4 Example of ring buffer

As like this, the ring buffer uses a continuous buffer memory, while conceptually it is consists of a circular buffer configuration as like data rotate. By using the ring buffer, there are following advantages. It can make efficient use of limited memory capacity. It also can overwrite when a ring buffer is full and support hot standby sparing structure of Gateway.

The ring buffer has a shortcoming. The ring buffer manager cannot determine whether the ring buffer is empty or full by using current pointers.

To compensate for this disadvantage, a ring buffer operation of CNet avoids a full state of the ring buffer its own. In other words, before inserting the data into the ring buffer, CNet module calculates a buffer size required in advance that newly be created and stored in the ring buffer by controllers.

For instance, a ring buffer of Fig.5 is conceptually drawn an operation of Fig.4. Number 2 of Fig.5 is showing that 14 data is in the ring buffer whose capacity is 20 data. Now, when 7 data is gathered, the ring buffer manager is aware that the ring buffer will be overflow so makes room of the ring buffer to be entered 8 data which is added one to 7 data to be stored as be shown in number 3 of Fig.5. After that, as like as number 4 of Fig.5, 7 data is inserted in the ring buffer.

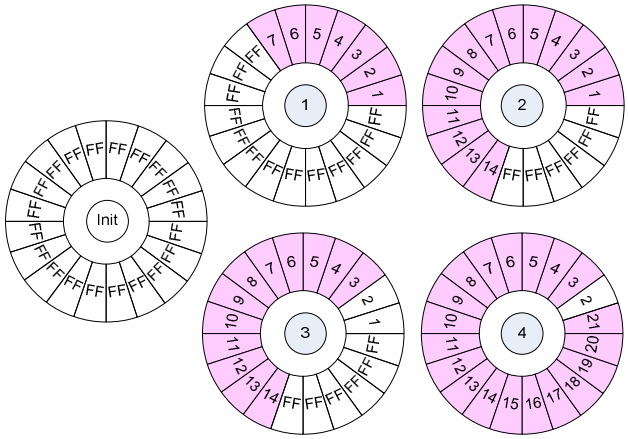

Fig. 5 concept of ring buffer

For efficient and smooth operation of the ring buffer, a ring buffer manager has five kinds of pointers. First pointer refers to the first memory address of the actual memory buffer. Second pointer refers to the last memory address of the buffer. Third pointer refers to the start address of data which is saved in the ring buffer and Fourth pointer refers to the end address of data which is saved in the buffer. And the last pointer refers to an unused portion address on managing the ring buffer. When data are sent to Gateway, the ring buffer manager determines a structure of the ring buffer and copies data to a send buffer.

By the management of the ring buffer, the newly generated data is added to the ring buffer and the data is removed from the ring buffer as many as transmitted data. It can use limited memory of controllers effectively and prevent data loss.

B. Ring buffer in CNet

The initial design of a ring buffer is implemented to accommodate only one Gateway. In order to support hot standby sparing of the data servers, a ring buffer manager has been fixed to act as buffers more than 2, conceptually, as shown in Fig.6.

Let's be assumed that a ring buffer is set at some point. It behaved as if it were two ring buffers, but is actually part of the same area of memory. The outside of the ring buffer corresponds to a Gateway-1 and the inside of the ring buffer corresponds to a Gateway-2. A ring buffer which refers to

Gateway-1 has data from 3 to 17 and a ring buffer which refers to Gateway-2 has data from 7 to 17. When data request message receives from Gateway-1, the ring buffer manager sends all data which corresponds to Gateway-1. Therefore after it sends data, the ring buffer for Gateway-1 is empty. Now, 7 data is created and get, so the ring buffer consists like number 3 of Fig.6. And at that time, if Gatewya-2 sends data request message to CNet module then the ring buffer manager sends all data from the ring buffer which refers to Gateway-2 as like as the case of Gateway-1.

As such, by operation of a ring buffer manager, a ring buffer manager can respond to multiple data servers on the same memory and it is guaranteed that data transmitted to each data server is identical.

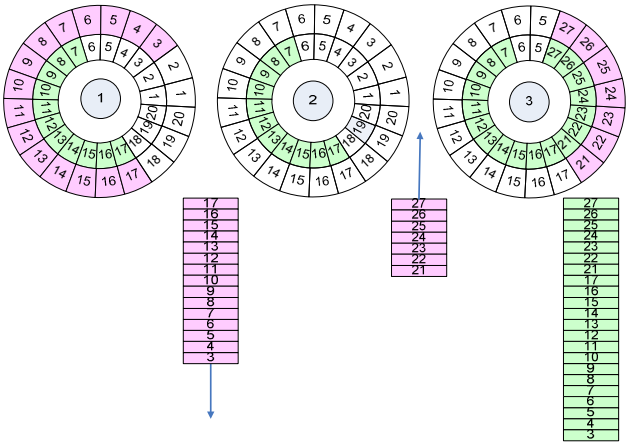

Fig. 6 ring buffer on hot standby sparing for data server

C. SOE data

SOE data is used to judge reasons of a failure or review a series of operations of controllers. Therefore, it should be secured that the integrity of SOE data has higher priority than general data. It is designed that SOE data is saved in temporary buffer for each data server before CNet module sends SOE data to Gateway, and the temporary buffer is cleaned after receiving response message that sent SOE data are received well by Gateway. If after sending SOE data, acknowledge message does not arrive and SOE data request message from same Gateway is received, CNet module sends SOE data previously sent to the Gateway. It can prevent from occurring loss of SOE data in advance.

A number of data to be transferred at one time is calculated by considering a number of occurred data. Because if huge SOE data are generated in a short time then it can give excessive load on the network, the number of data that can be transmitted at one time is limited by limiting the temporary buffer size.

In general data, recent data has a higher priority and overwrites buffer when buffer is full, by contrast, in case of SOE data, previous-generation data has a high priority than recent data, as a result, recent data is deleted inevitably when the SOE ring buffer is full. However, if the controller is slave then SOE data is also overwritten because the SOE ring buffer of the slave mode controller has a role not sending but saving.

The buffer location of storing General data or SOE data is different but a way of management for the ring buffer and sending to Gateway is basically same. While a master-mode controller responds to a request message from Gateway, a slave-mode controller does not respond to the request message but just get and store data.

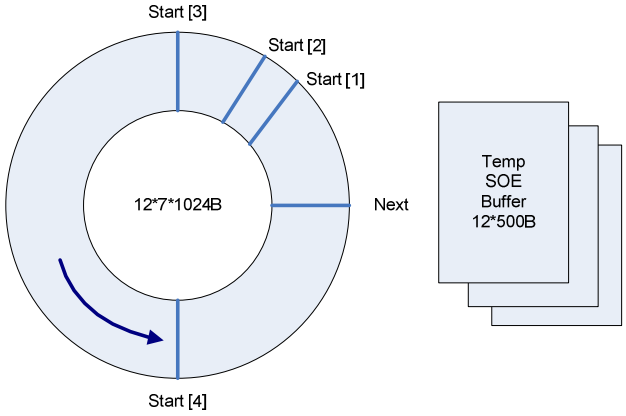

Fig. 7 Ring buffer for SOE data

VI. Monitoring the Status of Communication

CNet module monitors whether ring buffers overflow, whether Gateway normally sends request messages in given period and status of duplicated communication network. And during this diagnostic information, CNet module generates an error message or a fail message in order of previously set importance.

In addition, an ability to monitor a status of current communication is required to perform normal data communication without any loss of data. For this, CNet raises Heart-beat signals and performs surveillance on every established socket.

If CNet module determines that an error occurs in a controller and the controller can't do normal function, CNet module sets the fail-over signal to the controller is switching.

A. Diagnostic data

Unlike general data, in case of diagnostic data, all controllers send diagnostic data regardless of the state – master or slave – because each controller has different kinds of diagnostic data which is collected from those belonging boards to the controller and there are necessary diagnostic data even if the controller is slave.

Because main objective of diagnostic data is monitoring a current health at this point, CNet module does not buffer as like general data or SOE data but just collect data from OIS memory and send the data when diagnostic data request message is received.

B. Heart-beat signal

In the basic design of CNet module, Heart-beat-task is operated by independent task in order not to be affected by

other tasks. Heart-beat-task creates sockets as much as a number of possible networks and broadcasts heart-beat signals through the sockets for each network

Gateway creates sockets for broadcasting and receives the heart-beat signals from controllers which are connected with Gateway. Gateway analyzes the heat-beat signal, extracts CtrlID (ID of the controller) from the signal and attempts to connect to the controller. If Gateway does not receive any heart-beat signals over the network, then Gateway determines that the network has a failure and switches the communication network. And if duplicated communication networks are not normal then data server is switched.

CNet module creates a broadcasting message receiving socket and analyzes received heart-beat signals to judge whether communication lines are healthy or not. CNet module ignores the heart-beat signals generated itself. When CNet module receives no heart-beat signal from other controllers, then determines that the network has some error.

Through these operations, Heart-beat-task judges a soundness of each communication network and provide a starting point for Gateway to communicate with the controller through a sound network. Also, it offers a clue for Diagnostic-task to detect error of the controller and be switched to a healthy controller.

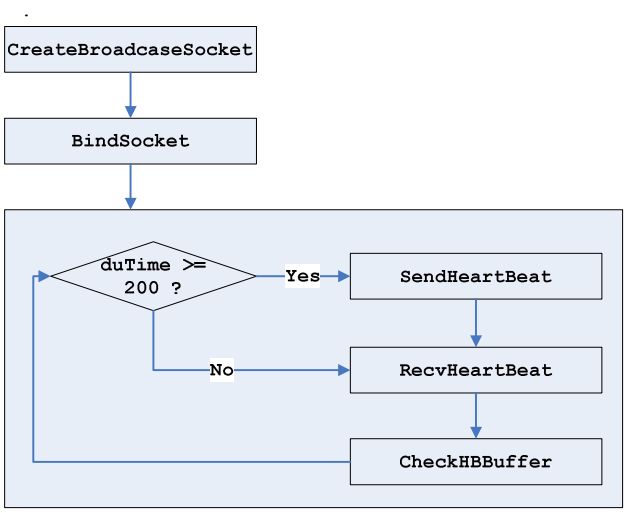

Fig. 8 Follow chart of Heart-beat-task

C. Checking the opened sockets

In CNet communication, the subject of connecting to controllers, maintaining the connections and reconnecting when a connection is expired is Gateway. In past, even if the network is fine, cause of other, for example error on Gateway, it is possible that communication is not done normally. To solve this problem, one task is needed to monitor created and connected sockets – Socket-check-task.

In Comm-task, last step of Comm-task is sending 'socket check message' to every established socket. Gateway which manages these sockets responds immediately after receives this message. Socket-check-task receives this response message and analyzes. If any response messages are not arrived during the amount of time, Socket-check-task determines that the socket is abnormal and closes the socket. If a controller closes a socket then Gateway which is connected the controller through the socket recognizes the socket has been closed. And if network and Gateway is normal, Gateway attempts a reconnection to the controller.

Thus, Socket-check-task determines the health of each established socket and controls it. It provides for Gateway to determine and for Diag-task to recognize and set a fail-over signal when socket has an error.

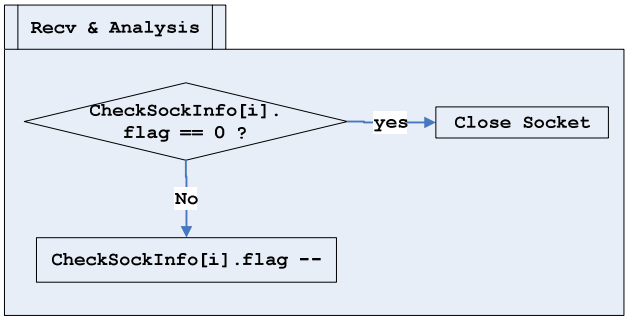

Fig. 9 follow chart of socket check task

VII. Conclusion

CNet module of IMCS operates separated 4 kinds of task by multi-tasking and manages buffers which contains 2 ring buffers, efficiently and effectively. There is no data loss because Get-task synchronizes with period of logic execution. Of course, even if the controller fail-over happens or network is abnormal, CNet module grabs every data from the controller and sends to Gateway. In addition, CNet module minimizes the load of data server and communication lines.

It is designed for the user to control a plant reliably and efficiently, and if it is necessary, analyze the data stably and systematically.

Acknowledgment

This work has supported by the Power Generation & Electricity Delivery of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Knowledge Economy. (No.R-2007-1-004-02)

References

[1] Standard of IEEE802.3
[2] Youkyung Park, "CNet protocol", Doosan heavy industries and constructions co., Ltd, March 2008.
[3] Bary W. Johnson, "Design and Analysis of Fault-Tolerant Digital System", Addison-Wesley Publishing Company, 1989.

Youkyung Park was born in Republic of KOREA at 1981. Ms. Park was graduated of KOREA University (Seoul, Republic of KOREA) at 2005 and took the master degree of electric engineering from KOREA University (Seoul, Republic of KOREA) at 2007. Ms. Park published a paper – "A proposal of Voltage Stability Analysis Using Database System" at 2006 and got the best paper award in CICS 2010 by a paper – "Ethernet communication module design". Ms. Park has been with "Doosan Heavy Industries and Constructions Company" on Daejeon, Republic of KOREA, since 2007. Now Ms. Park is ASSISTANT MANAGER of "I&C Research Team" and researches communication design of DCS.

SeungYup Kang was graduated of HONGIK University and took the master degree of computer science from HONGIK University (Seoul, Republic of KOREA). Mr.Kang has been with "Doosan Heavy Industries and Constructions Company" on Daejeon , Republic of KOREA since 2004. Mr.Kang is STAFF R&PD ENGINEER of "I&C Research Team".

SungHo Kim was graduated of KYUNGPOOK National University (Daegu, Republic of KOREA). Mr.Kim has been with "Doosan Heavy Industries and Constructions Company" on Daejeon , Republic of KOREA since 1994. Mr.Kim is STAFF R&PD ENGINEER of "I&C Research Team".

SimKyun Yook received Ph.D. in mechanical engineering from KYUNGPOOK National University (Daegu, Republic of KOREA). Mr.Yook has been with "Doosan Heavy Industries and Constructions Company" on Daejeon , Republic of KOREA since 1993. Mr.Yook is TEAM-LEADER of "I&C Research Team".