Group Communication Protocol for Autonomic Computing

Tomoya Enokido
Faculty of Business Administration,
Rissho University
4-2-16, Osaki, Shinagawa, Tokyo,
141-8602, Japan
eno@ris.ac.jp

Makoto Takizawa
Dept. of Computers and Systems Engineering,
Tokyo Denki University
Ishizaka, Hatoyama, Hiki, Saitama,
350-0394, Japan
taki@takilab.k.dendai.ac.jp

Abstract

We discuss a group communication protocol which supports applications in change of QoS supported by networks and required by applications. An autonomic group protocol is realized by cooperation of multiple autonomous agents. Each agent autonomously takes a class of each protocol function. Classes taken by an agent are required to be consistent with but might be different from the others. We make clear what combination of classes can be autonomously taken by agents. We also present how to change retransmission ways.

1. Introduction

Peer-to-Peer (P2P) systems [1] are getting widely available like autonomic computing [3]. Multiple peer processes first establish a *group* and then messages are exchanged among the processes. Group communication protocol [2, 4, 5, 6, 7, 8, 9] supports basic communication mechanism like the causally ordered and atomic delivery of messages. A group protocol is implemented in protocol modules which is composed of protocol functions; multicast/broadcast, receipt confirmation, detection and retransmission of messages lost, ordering of messages received, and membership management. There are various ways to realize each of these functions.

The complexity and efficiency of implementation of a group protocol depends on what type and quality of service (QoS) are supported by the underlying network. Since messages may be lost and unexpectedly delayed due to congestions and faults in the network, QoS parameters are dynamically changed. The higher level of communication function is supported, the larger computation and communication overheads are implied. Hence, the system has to take classes of functions necessary and sufficient to support service required by application.

The paper [8] discusses a communication architecture which satisfies application requirements in change of network service. However, a group protocol cannot be dynamically changed each time QoS supported by the underlying network is changed. In ISIS [2], protocol modules which support service required can be constructed. However, a protocol module of each process cannot be changed and every process has to take the same module. It is not easy to change protocol functions in all the processes since a large number of processes are cooperating.

In this paper, we discuss an *autonomic* group protocol which can support QoS required by applications even if QoS supported by the underlying network is changed. Each protocol module is realized in an autonomous agent. An agent autonomously changes classes, i.e. implementations of each protocol function depending on network QoS monitored. Here, an agent might take different classes of protocol functions from other agents but *consistent* with the other agents. We make clear what combination of classes can be autonomously taken by agents. We also present how to change retransmission ways.

In section 2, we show a system model. In section 3, we discuss classes of protocol functions. In section 4, we present an agent-based group protocol. In section 5, we discuss how to change retransmission functions.

2. System Model

2.1. Autonomic group agent

A group of multiple *application processes* $A_1, ..., A_n$ ($n \geq 2$) are cooperating by taking usage of group communication service. The group communication service is supported by cooperation of multiple *autonomous group* (AG) agents $p_1, ..., p_n$ [Figure 1]. Here, a term "*agent*" means an AG agent in this paper. The underlying network supports a pair of agents with basic communication service which is characterized by QoS parameters; delay time [msec], message loss ratio [%], and bandwidth [bps].

A group protocol is realized in a collection of protocol functions; coordination, transmission, confirmation, retransmission, ordering of message, detection of message lost, and membership management. There are multiple ways to implement each protocol function. A $class$ shows a way to implement a protocol function. The classes for each function are stored in a protocol class base (CB) of each agent. Each agent p_i autonomously takes one class for each protocol function from CB, which can support an application with necessary and sufficient QoS by taking usage of basic communication service with given QoS supported by the underlying network. Each agent p_i stores QoS information of the underlying network in a QoS base (QB) of p_i. If enough QoS cannot be supported or too much QoS is supported for the application, the agent p_i reconstructs a combination of protocol function classes by autonomously selecting a class for each protocol function in CB. Each agent negotiates with other agents to make a consensus on which class to take for each protocol function if some pair of agents are inconsistent.

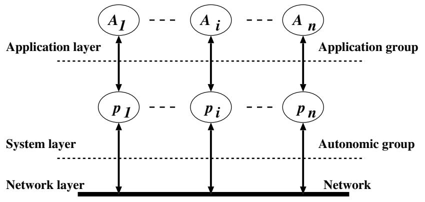


Figure 1. System model.

2.2. Views

A $group$ G is composed of multiple agents p_1, ..., p_n ($n > 1$). In a group G including larger number of agents, it is not easy for each agent to maintain membership information of the group. Each agent p_i has a view $V(p_i)$ which is a subset of agents to which the agent p_i can deliver messages directly or indirectly via agents. Thus, a view is a subgroup of the group G. For every pair of agents p_i and p_j, p_i in $V(p_j)$ if p_j in $V(p_i)$. A pair of different views V_1 and V_2 may include a common $gateway$ agent p_k. A collection of gateway agents is also a view V_3.

An agent p_i which takes a message m from an application process A_i and sends the message m is an original $sender$ agent of the message m. If an agent p_j delivers a message m to an application process, the agent p_j is an original $destination$ agent of the message m. If an agent p_k forwards a message m to another agent in a same view V, p_k is a $routing$ agent. A $local$ sender and destination of a message m are agents which send and receive m in a view, respectively.

3. Functions of Group Protocol

In this paper, we consider protocol functions, coordination, transmission, confirmation, and retransmission functions, which are the most significant to design and implement a group protocol. There are $centralized$ and $distributed$ classes to coordinate the cooperation of agents. In the centralized control, there is one centralized controller in a view. In the distributed control, each agent makes a decision on correct receipt, delivery order of messages received by itself.

There are $centralized$, $direct$, and $indirect$ classes to multicast a message in a view. In the $centralized$ transmission, an agent first sends a message to a controller agent. Then, the controller agent forwards the message to all the destination agents in a view. In the $direct$ transmission, each agent directly sends a message to each destination agent in a view V. In the $indirect$ transmission, a message is first sent to some agent in a view V. The agent forwards the message to another agent and finally delivers the message to the destination agents in the view V.

There are $centralized$, $direct$, $indirect$, and $distributed$ classes to confirm receipt of a message in a view V. In the centralized confirmation, every agent sends a receipt confirmation message to a controller agent in a view V. After receiving confirmation messages from all the destination agents, the controller agent sends a receipt confirmation to the local sender agent. In the $direct$ confirmation, each destination agent p_i in the view V sends a receipt confirmation of a message m to the local sender agent p_i which first sends the message m in the view V. In the $indirect$ confirmation, a receipt confirmation of a message m is sent back to a local sender agent p_i in a view V by each agent p_j which has received the message m from the local sender agent p_i. In the $distributed$ confirmation, each agent which has received a message m sends a receipt confirmation of the message m to all the other agents in the same view [7].

There are $sender$ and $destination$ retransmission classes [Figure 2]. In the $sender$ retransmission, the local sender agent p_j which has first sent a message m in a view V retransmits m to p_i which fails to receive m. In the $destination$ retransmission, one or more than one destination agent in a view V which have safely received the message m forwards m to p_i which fails to receive m. In the $distributed$ confirmation, each agent can know if every other destination agent safely receives a message m. If a destination agent p_j receives no confirmation from an agent p_i, p_j detects p_i to lose m.

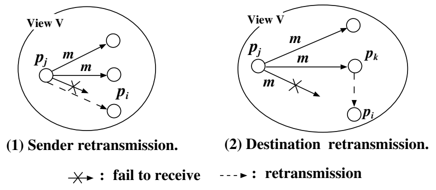


Figure 2. Retransmission.

4. Autonomic Group Protocol

4.1. Local protocol instance

Let **F** be a set of the significant protocol functions $\{C$(coordination), T(transmission), CF(confirmation), R(retransmission)$\}$. Let $Cl(F)$ be a set of classes to implement a protocol function F in **F**. Table 1 shows possible classes for the protocol functions, C, T, CF, and R. We rewrite **F** to be a set $\{F_1, F_2, F_3, F_4\}$ of protocol functions where each element F_i shows a protocol function, i.e. $\langle F_1, F_2, F_3, F_4\rangle = \langle C, T, CF, R\rangle$. A tuple $\langle c_1, c_2, c_3, c_4\rangle \in Cl(F_1)\times Cl(F_2)\times Cl(F_3)\times Cl(F_4)$ is referred to as $protocol\ instance$. Each agent takes a protocol instance $C = \langle c_1, c_2, c_3, c_4\rangle$, i.e. a class c_i is taken for each protocol function F_i (i = 1, 2, 3, 4). Here, some protocol instances out of possible 48 ones cannot work in an agent.

[De nition] A protocol instance $\langle c_1, c_2, c_3, c_4\rangle$ is $consistent$ iff an agent taking the protocol instance can work with other agents which take the same protocol instance to support group communication service.

A protocol $profile$ is a consistent protocol instance. In Table 2, seven possible protocol profiles are summarized. A protocol profile $signature$ "$c_1c_2c_3c_4$" denotes a protocol profile $\langle c_1, c_2, c_3, c_4\rangle$. For example, $DDDirS$ shows a protocol profile $\langle D, D, Dir, S\rangle$ which is composed of distributed control (D), direct transmission (D), direct confirmation (Dir), and sender retransmission (S) classes. Let $PF(1)$, $PF(2)$, $PF(3)$, $PF(4)$, $PF(5)$, $PF(6)$, and $PF(7)$ show the protocol profiles $CCCenS$, $DDDirS$, $DDDisS$, $DDDisD$, $DIIndS$, $DIDisS$, and $DIDisD$, respectively, which are shown in Table 2. Let **P** be a set $\{PF(i) \mid i = 1, ..., 7\}$ of all the protocol profiles. In $PF(i)$, i is referred to as the protocol profile number.

Table 1. Protocol classes $Cl(F)$.

Function F	Protocol classes $Cl(F)$
C	{*C*(*centralized*), *D*(*distributed*)}
T	{*C*(*centralized*), *D*(*direct*), *I*(*indirect*)}
CF	{*Cen*(*centralized*), *Dir*(*direct*), *Ind*(*indirect*), *Dis*(*distributed*)}
R	{*S*(*sender*), *D*(*destination*)}

Table 2. Protocol pro les.

Control	Transmission	Confirmation	Retransmission	Signature
Centralized	Centralized	Centralized	Sender	CCCenS
Distributed	Direct	Direct	Sender	DDDirS
		Distributed	Sender	DDDisS
			Destination	DDDisD
	Indirect	Indirect	Sender	DIIndS
		Distributed	Sender	DIDisS
			Destination	DIDisD

4.2. Global protocol instance

Let C_i be a consistent protocol instance $\langle c_{i1}, ..., c_{i4}\rangle \in$ **P**, i.e. protocol profile taken by an agent p_i (i = 1, ..., n). A $global$ protocol instance C for a view $V = \{p_1, ..., p_n\}$ is a tuple $\langle C_1, ..., C_n\rangle$ where each element C_i is a protocol profile which an agent p_i takes. Each C_i is referred to as $local$ protocol instance of an agent p_i. In traditional protocols, every agent has to take a same local protocol profile. A global protocol instance $C = \langle C_1, ..., C_n\rangle$ is $complete$ if $C_1 = \cdots = C_n$. A global protocol instance $C = \langle C_1, ..., C_n\rangle$ is $incomplete$ if $C_i \neq C_j$ for some pair of agents p_i and p_j. A global protocol instance C is $globally\ consistent$ if a collection of agents where each agent p_i takes a protocol profile C_i (i = 1, ..., n) can cooperate. A $global\ protocol\ profile$ is a consistent global protocol instance. In this paper, we discuss a group protocol where a view V of agents $p_1, ..., p_n$ can take an incomplete but consistent global protocol instance $C = \langle\ C_1, ..., C_n\ \rangle$. First, following types of protocol instances are globally inconsistent:

[Property] A global protocol instance $C = \langle\ C_1, ..., C_n\ \rangle$ of a view V is not consistent if V is composed of more than three agents and the global protocol instance C satisfies one of the following conditions:

1. At least one agent in V takes the protocol profile $CCCenS$ and the global protocol instance C is not complete.
2. At least one agent takes an $indirect$ transmission class in V and at least one other agent takes a $direct$ confirmation class in V.

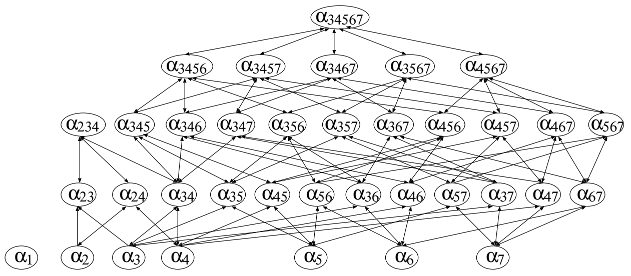


Figure 3. Hasse diagram.

We introduce a notation α_I where $I \in 2^{\{1,\cdots,7\}}$ as follows:

1. α_i indicates a global protocol profile where all the agents take the same local protocol profile $PF(i)$ (i = 1, ..., 7).
2. Let I be a sequence of protocol profile numbers $i_1 \cdots i_l$ ($l \leq 7$), α_I shows a global protocol instance where each agent takes one of the local protocol profiles $PF(i_1)$, ..., $PF(i_l)$ and each protocol profile $PF(i_k)$ is taken by at least one agent (k = 1, ..., l).

For example, α_{23} means a global protocol instance where every agent takes $PF(2) = DDDirS$ or $PF(3) = DDDisS$.

[De nition] A global protocol instance α_I can be transited to another global protocol instance α_J ($\alpha_I \rightarrow \alpha_J$) iff

1. if J = Ii, the agents can support group communication service even if an agent taking $PF(k)$ where $k \in I$ autonomously takes $PF(i)$ $(i \neq k)$.
2. if I = Jj, the agents can support group communication service even if an agent taking $PF(j)$ takes $PF(k)$ where $k \in I$.
3. For some global protocol instance α_K, $\alpha_I \rightarrow \alpha_K$ and $\alpha_K \rightarrow \alpha_J$.

Figure 3 shows a Hasse diagram where a node shows a global protocol profile and a directed edge from α_I to α_J indicates a transition relation "$\alpha_I \rightarrow \alpha_J$". For example, $\alpha_2 \rightarrow \alpha_{24}$ since an agent can autonomously change a local protocol profile $PF(2)$ = $DDDirS$ to $PF(4)$ = $DDDisD$.

5. Retransmission

5.1. Cost model

We discuss how an agent can autonomously change the retransmission classes. Suppose there are three agents p_s, p_t, and p_u in a view V. An agent p_s sends a message m to a pair of agents p_t and p_u. Then, the agent p_t receives m while p_u fails to receive m. Let d_{st} be delay time between agents p_s and p_t [msec], f_{st} show probability that a message is lost, and b_{st} indicate bandwidth [bps].

First, let us consider the sender retransmission. Let $|m|$ show the size of a message m [bit]. It takes ($2d_{su}$+ $|m|$ / b_{su}) [msec] to detect message loss after p_s sends a message m. Then, p_s retransmits m to p_u. Here, the message m may be lost again. The expected time ST_{su} and number SN_{su} of messages to deliver a message m to p_u are given as ST_{su} = ($2d_{su}$ + $|m|$ / b_{su}) / $(1 - f_{su})$ and SN_{su} = 1 / $(1 - f_{su})$.

In the destination retransmission, some destination agent p_t forwards the message m to p_u. The expected time DT_{su} and number DN_{su} of messages to deliver a message m to p_u are given as DT_{su} = (d_{su} + $|m|$ / b_{su} + d_{ut}) + ($2d_{ut}$ + $|m|$ / b_{ut}) / $(1 - f_{ut})$ if $d_{st} \leq d_{su}$ + d_{ut}, (d_{st} + $|m|$ / b_{st}) + ($2d_{ut}$ + $|m|$ / b_{ut}) / $(1 - f_{ut})$ otherwise. DN_{su} = 1 + 1 / $(1 - f_{ut})$. If $ST_{su} > DT_{su}$, the destination agent p_t can forward the message m to the faulty agent p_u because the message m can be delivered earlier.

Each agent p_t monitors delay time d_{ut}, bandwidth b_{ut}, and message loss probability f_{ut} for each agent p_u which are received in the QoS base (QB). For example, p_t obtains the QoS information by periodically sending QoS information messages to all the agents in a view. The agent p_t maintains QoS information in a variable Q of QB where Q_{ut} = $\langle b_{ut}, d_{ut}, f_{ut} \rangle$ for u = 1, ..., n. If p_t receives QoS information from p_s, Q_{su} = $\langle b_{su}, d_{su}, f_{su} \rangle$ for u = 1, ..., n.

5.2. Change of retransmission class

Suppose an agent p_s sends a message m and every agent p_t take the sender retransmission class, C_t = $\langle \cdots, S \rangle$. An agent p_u fails to receive m. According to the change of QoS supported by the underlying network, the sender agent p_s makes a decision to change the retransmission class with the destination one, say an agent p_t forwards m to p_u. However, p_t still takes the sender retransmission. Here, no agent forwards m to p_u. In order to prevent these silent situations, we take a following protocol:

1. A sender agent p_s sends a message m to all the destination agents. Every destination agent sends receipt confirmation not only to the sender agent p_s but also to the other destination agents.
2. If an agent p_t detects that a destination p_u has not received m, p_t selects a retransmission class which p_t considers to be optimal based on the QoS information.
   - 2.1 If p_t is a destination agent and changes a retransmission class, p_t forwards m to p_u and sends $Retx$ message to the sender p_s.
   - 2.2 If p_t is a sender of a message m and takes the sender retransmission class, p_t retransmits m to p_u. If p_t takes a destination retransmission class, p_t waits for $Retx$ message from a destination. If p_t does not receive $Retx$, p_t retransmits m to p_u.

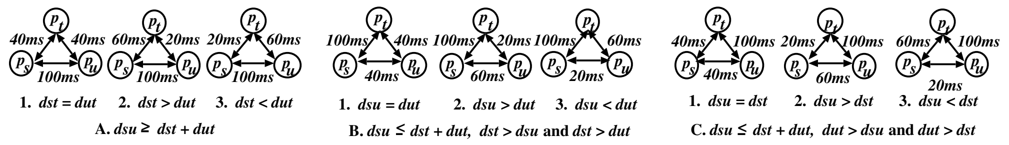
[**Theorem**] At least one agent forwards a message m to an agent which fails to receive the message m.

5.3. Evaluation

We evaluate the autonomic group protocol (AGP) in terms of delivery time of a lost message. We make the following three assumptions on this evaluation. 1) d_{st} = d_{ts} for every pair of p_s and p_t. 2) The protocol processing time of every process is same. 3) No confirmation message is lost.

Let us consider a view V = $\{p_s, p_t, p_u\}$ where every agent takes a profile $DDDisS$. Here, suppose that an agent p_s sends a message m to a pair of agents p_t and p_u in a view V. Then, the agent p_t receives m while another agent p_u fails to receive m. After the sender p_s and destination p_t detect the destination agent p_u fails to receive m, p_s and p_t autonomously select the retransmission class based on the QoS information. Here, we evaluate time to deliver a message m to a faulty agent p_u. In the view V, we assume that bandwidth between every pair of agents is same (b_{st} = b_{su} = b_{ut} = 10Mbps) and f_{st} = f_{su} and f_{ut} = 0 %. Figure 4 shows an agent graph for V where each node denotes an agent and each edge shows a communication channel between agents. A label of the edge indicates delay time.

First, we consider a case $d_{su} \geq d_{st}$ + d_{ut} [Figure 4 A]. There are further cases: d_{st} = d_{ut} [A.1], $d_{st} > d_{ut}$ [A.2], and $d_{st} < d_{ut}$ [A.3]. Figure 5 shows the expected time DT_{su} for three cases. In Figure 5, horizontal axis shows a message loss probability of f_{su} and f_{ut}. For case of Figure 4 A.2, $DT_{su} < ST_{su}$. For case of Figure 4 A.1, $DT_{su} < ST_{su}$ if $f_{su} > 15\%$ and $f_{ut} > 15\%$. For case of Figure 4 A.3, DT_{su} $< ST_{su}$ if $f_{su} > 50\%$ and $f_{ut} > 50\%$.



40ms 40ms 100ms
60ms 20ms 100ms
20ms 60ms 100ms
1. dst = dut 2. dst > dut 3. dst < dut
A. dsu ≥ dst + dut
100ms 40ms 40ms
100ms 20ms 60ms
100ms 60ms 20ms
1. dsu = dut 2. dsu > dut 3. dsu < dut
B. dsu ≤ dst + dut, dst > dsu and dst > dut
40ms 100ms 40ms
20ms 100ms 60ms
60ms 100ms 20ms
1. dsu = dst 2. dsu > dst 3. dsu < dst
C. dsu ≤ dst + dut, dut > dsu and dut > dst

Figure 4. AG agent graph.

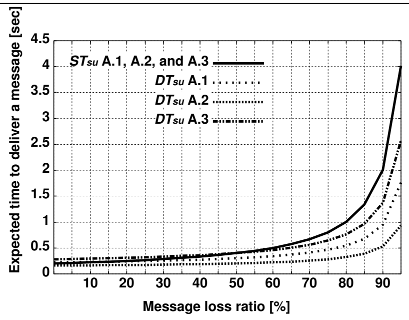


Figure 5. $d_{su} \geq d_{st}$ + d_{ut}.

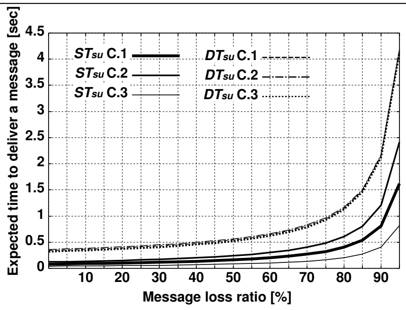


Figure 7. $d_{su} \leq d_{st}$ + d_{ut}, $d_{ut} > d_{su}$, **and** $d_{ut} > d_{st}$.

Next, there are further following cases for $d_{su} \leq d_{st} + d_{ut}$ [Figure 4]:

a. $d_{st} > d_{su}$ and $d_{st} > d_{ut}$: $d_{su} = d_{ut}$[B.1], $d_{su} > d_{ut}$[B.2], and $d_{su} < d_{ut}$[B.3].

b. $d_{ut} > d_{su}$ and $d_{ut} > d_{st}$: $d_{su} = d_{st}$[C.1], $d_{su} > d_{st}$[C.2], and $d_{su} < d_{st}$[C.3].

The expected time DT_{su} [Figure 4 B and 4 C] is shown for these six cases in Figures 6 and 7. For Figure 4 B.1 and B.3, $DT_{su} > ST_{su}$. For Figure 4 B.2, $DT_{su} < ST_{su}$ if f_{su} $>$ 20% and $f_{ut} >$ 20%. For Figure 4 C, $DT_{su} > ST_{su}$.

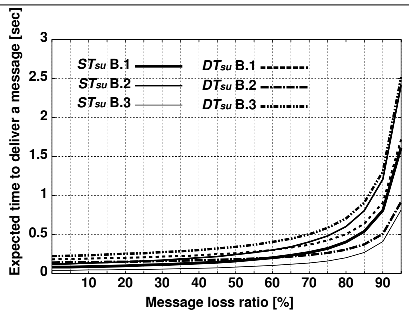


Figure 6. $d_{su}{\leq}d_{st}$+d_{ut}, $d_{st}{>}d_{su}$, **and** $d_{st}{>}d_{ut}$.

6. Concluding Remarks

In this paper, we discussed an agent-based architecture to support applications with autonomic group service in change of network and application QoS. We made clear what classes of functions to be realized in group communication protocols. Every agent autonomously changes a class of each protocol function which may not be the same as but are consistent with the other agents. We discussed how to support applications with the autonomic group service by changing retransmission classes as an example. We showed which retransmission class can be adopted for types of network configuration in the evaluation.

References

[1] P. E. Agre. P2P and the Promise of Internet Equality. *Communication of the ACM*, 46(2):39–42, 2003.

[2] K. Birman, A. Schiper, and P. Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Trans. on Computer Systems*, 9(3):272–290, 1991.

[3] IBM Corporation. Autonomic Computing Architecture : A Blueprint for Managing Complex Computing Environments. 2002. http://www-3.ibm.com/autonomic/pdfs/ACwhitepaper 1022.pdf.

[4] M. F. Kaashoek and A. S. Tanenbaum. An evaluation of the amoeba group communication system. *Proc. of IEEE ICDCS-16*, pages 436–447, 1996.

[5] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *CACM*, 21(7):558–565, 1978.

[6] F. Mattern. Virtual Time and Global States of Distributed Systems. *Parallel and Distributed Algorithms*, pages 215–226, 1989.

[7] A. Nakamura and M. Takizawa. Reliable Broadcast Protocol for Selectively Ordering PDUs. *Proc. of IEEE the 11th International Conference on Distributed Computing Systems (ICDCS-11)*, pages 239–246, 1991.

[8] v. Renesse, K. P. Birman, and S. Maffeis. Horus: A Flexible Group Communication System. *CACM*, 39(4):76–83, 1996.

[9] C. Steketee, W. P. Zhu, and P. Moseley. Implementation of Process Migration in Amoeba. *Proc. of IEEE ICDCS-14*, pages 194–201, 1994.