Autonomic Group Protocol for Distributed Systems

Tomoya Enokido and Makoto Takizawa
Department of Computers and Systems Engineering
Tokyo Denki University
{*eno, taki*}*@takilab.k.dendai.ac.jp*

Abstract

An autonomic group protocol supports applications with enough quality of service (QoS) in change of QoS supported by networks and applications. An autonomic group service is supported for applications by cooperation of multiple autonomous agents. Each agent autonomously takes a class of each protocol function like retransmission. Classes taken by an agent are required to be consistent with but might be different from the others. A group is composed of views each of which is a subset of agents and in each of which agents autonomously take protocol classes consistent with each other. We discuss a model of autonomic group protocol. We also present how to autonomously change retransmission ways in a group as an example.

1 Introduction

Peer-to-Peer (P2P) systems [1] are getting widely available like grid computing [4] and autonomic computing [5]. Group communication supports basic communication mechanism to realize cooperation of multiple peer processes. Multiple peer processes first establish a $group$ and then messages are exchanged among the processes [2, 7, 8, 10, 12]. There are group protocols which support the ordered and atomic delivery of messages [2, 7, 8, 10, 12]. A group protocol is realized by 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 like selective and go-back-n retransmissions [6].

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

The paper [12] discusses a communication architecture which supports a group of multiple processes which satisfies application requirements in change of network service. However, a protocol cannot be dynamically changed each time QoS supported by the underlying network is changed. In addition, each process in a group has to use the same protocol functions. It is not easy to change protocol functions in all the processes since a large number of processes are cooperating and some computers like personal computers and mobile computers are not always working well.

In this paper, we discuss an $autonomic$ group protocol which can support types and quality (QoS) of service 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 implementation of each group protocol function depending on network QoS monitored. Here, an agent might take different types of protocol functions from other agents but $consistent$ with the other agents. We discuss what combination of protocol functions are consistent. Each agent has a $view$ which is a subset of agents to which the agent can directly send messages. If a group is too large for each agent to perceive QoS supported by other agents and manage the group membership, the group is decomposed into views. In each view, messages are exchanged by using its own consistent protocol functions. A pair of different views might take different protocols.

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 architecture to support the autonomic group service. 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 peer $autonomous\ group$ (AG) agents p_1, ..., p_n through exchanging messages by taking usage of underlying network service [Figure 1]. For simplicity, a term "$agent$" means an AG agent in this paper. The underlying network supports a pair of agents with communication service which is characterized by quality of service (QoS) parameters; delay time [msec], message loss ratio [%], and bandwidth [bps].

The cooperation of multiple AG agents is coordinated by a group protocol. A group protocol is realized in a collection of protocol functions, transmission, confirmation, retransmission, ordering of message, detection of message lost, coordination schemes, and membership management. There are multiple ways to implement each protocol function. A protocol function $class$ means a way of implementation of protocol function. The classes are stored in a protocol class base (CB). Each application process A_i takes group communication service through an agent p_i. Each agent p_i autono
takes one class for each group protocol function from t
tocol class base CB, which can support an applicatic

necessary and sufficient QoS by taking usage of basic communication service with given QoS supported by the underlying network. Each agent p_i monitors QoS supported by the underlying network. The network QoS information monitored is stored 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 group protocol function classes which are consistent with the other agents by selecting a class for each protocol function in the CB. Here, each agent negotiates with other agents to make a consensus on which class to take for each protocol function. In the paper, we discuss how each agent autonomously change protocol function classes in change of QoS monitored.

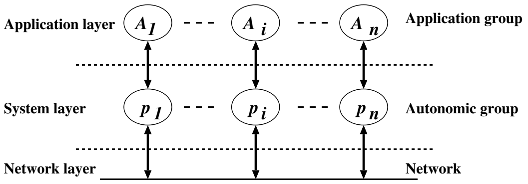


Figure 1. System model.

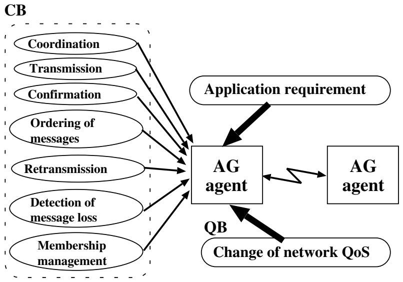


Figure 2. Autonomic group protocol.

2.2 Views

A *group* G is composed of multiple autonomous group (AG) agents p_1, ..., p_n ($n > 1$). An agent is an autonomous peer process which supports application process with group communication service by exchanging messages with other agents. The cooperation of agents is coordinated in a distributed way. In a group G including larger number of agents, it is not easy for each agent to deliver messages to all the agents and maintain membership information. 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. We assume that for every pair of agents p_i and p_j, p_i in $V(p_j)$ iff p_j in $V(p_i)$. Each agent p_i maintains membership of its view $V(p_i)$. Current information systems are composed of local networks which are interconnected with each other in a trunk network. Here, a view can be a collection of agents interconnected in a local network. A pair of different views V_1 and V_2 may include a common agent p_k. The agent p_k is a *gateway* agent between agents in V_1 and V_2. A collection of gateway agents which are interconnected in a trunk network is also a view V_3. Here, the views V_1, V_2, and V_3 are hierarchically structured. If an agent p_i belongs to only one view, p_i is a *leaf* agent. 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 A_j, 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. Let $src(m)$ be an original source agent and $dst(m)$ be a set of original destination agents. A *local* sender and destination of a message m are agents which send and receive m in a view, respectively.

A view V which includes all the agents in a group G is referred to as *complete*. A *global* view is a complete view in a group G. If $V \subset G$, V is *partial*. A partial view V is changed if an agent joins and leaves the view V. If a view $V(p_i)$ is changed, $V(p_i)$ is *dynamic*. For example, an agent p_i sends each message to different agents. If $V(p_i)$ is invariant, $V(p_i)$ is *static*.

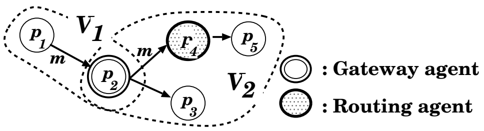
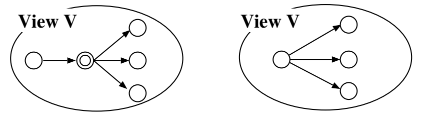


Figure 3. Group views.

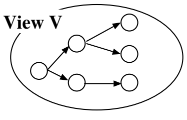
3 Functions of Group Protocol

A group protocol is realized in a collection of following protocol functions: coordination of the agents, message transmission, receipt confirmation, retransmission, detection of message loss, ordering of messages, and membership management. There are multiple ways to realize each of these functions. A *class* of a protocol function shows one way to implement the protocol function. One protocol module for an autonomous group (AG) agent is a collection of protocol classes, each of which is for one protocol function. We discuss what classes exist for each protocol function in this section and what combination of classes are *consistent* in the succeeding section.

There are *centralized* and *distributed* approaches to coordinating the cooperation of agents in a view. In the centralized control, there is one centralized controller in a view V. On the other hand, there is no centralized controller in the distributed control scheme. Each agent makes a decision on correct receipt, delivery order of messages received, and group membership by itself.



(1) Centralized transmission (2) Direct transmission



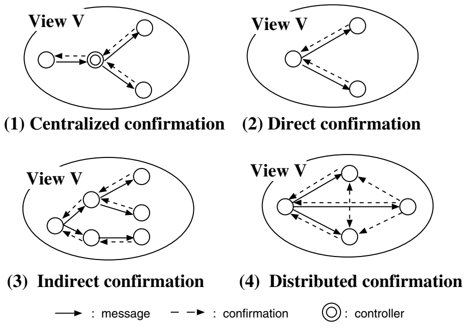
(3) Indirect transmission

Figure 4. Transmission schemes.

There are *centralized*, *direct*, and *indirect* approaches to multicasting a message to multiple agents in a view [Figure 4]. In the *centralized* transmission, an agent first sends a message to a *forwarder* agent and then the forwarder agent forwards the message to all the destination agents in a view [Figure 4 (1)]. The forwarder agent plays a role of a centralized controller. It takes at least two rounds to deliver messages since every message is forwarded by the controller. In the *direct* transmission, each agent directly not only sends a message to each destination agent but also receives messages from other sender agents in a view V [Figure 4 (2)]. Thus, a message can be delivered to every destination by one

In the *indirect* transmission, a message is first sent t

agent in a view V. The agent forwards the message to agent and finally delivers the message to the destination agents in the view V [Figure 4 (3)]. Tree routing [3] is an example. It takes a longer time than one round to deliver a message in the indirect transmission scheme.


(1) Centralized confirmation (2) Direct confirmation

(3) Indirect confirmation (4) Distributed confirmation

Figure 5. Confirmation schemes.

There are *centralized*, *direct*, *indirect*, and *distributed* schemes to confirm receipt of a message in a view V. In the centralized confirmation, every agent sends a receipt confirmation message to one *confirmation* agent in a view V. After receiving confirmation messages from all the destination agents, the destination agent sends a receipt confirmation to the local sender agent [Figure 5 (1)]. 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 [Figure 5 (2)]. 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 [Figure 5 (3)]. 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 [10] [Figure 5 (4)]. Each agent in a same view V can know whether or not all the other agents in V have received a same message m by using the distributed confirmation scheme.

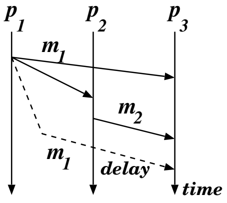


Figure 6. Causally ordered delivery.

A group of multiple agents are exchanging messages in the network. A message m_1 *causally precedes* another message m_2 ($m_1 \rightarrow m_2$) if and only if (iff) a sending event of m_1 *happens before* a sending event of m_2 [7]. A message m_1 is *causally concurrent* with another message m_2 ($m_1 \parallel m_2$) if neither $m_1 \rightarrow m_2$ nor $m_2 \rightarrow m_1$. For example, suppose there are three agents p_1, p_2, and p_3 in a group G [Figure 6]. An agent p_1 sends a message m_1 to a pair of agents p_2 and p_3. The agent p_2 sends a message m_2 to p_3 after receiving another message m_1. Here, m_1 causally precedes m_2 ($m_1 \rightarrow m_2$). Due to communication delay, m_1 may arrive at p_3 after m_2. The agent p_3 is required to deliver m_1 before m_2 because $m_1 \rightarrow m_2$. Messages received are ordered by each agent in the distributed approach. In order to causally deliver messages, realtime clock with NTP (network time protocol) [9], linear clock [7], and vector clock [8] are used.

There are *sender* and *destination* retransmission schemes with respect to which agent retransmits a message m lost [Figure 7]. Suppose an agent p_j sends a message m to agents and one destination agent p_i fails to receive m. In the *sender retransmission*, the local sender agent p_j which first sent the message m in the view V retransmits the message m to p_i. In the *destination retransmission*, one or more than one destination agent in the view V which have safely received the message m forwards m to the agent p_i which fails to receive m [Figure 7 (2)]. In the *distributed* confirmation, each agent can know if every other destination agent safely receives a message m.

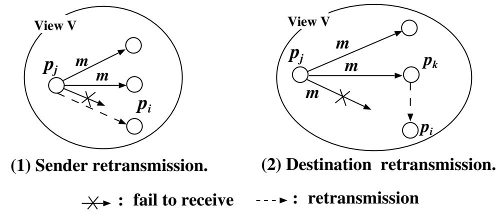

(1) Sender retransmission. (2) Destination retransmission.

Figure 7. Retransmission scheme.

There are *centralized* and *distributed* ways for managing the membership. In the centralized way, one membership manager communicates with all the member agents to obtain their states. In the distributed way, each agent obtains the states of the other agents by communicating with other agents.

A *centralized* system is one with centralized coordination, transmission, and confirmation. There is one controller which forwards messages to destination agents and confirms receipt of messages. Most traditional distributed systems like teleconference systems and Amoeba [11] take the centralized approach. A system with distributed coordination, transmission, and centralized confirmation system is classified to be *decentralized*. ISIS [2] takes the decentralized approach. A sender agent coordinates transmission and receipt of a message. Destination agents send the receipt confirmation to the sender agent. Takizawa *et al.* [10] take the *distributed* approach which coordination, transmission, and confirmation are distributed. Here, every destination agent sends the receipt confirmation to not only the sender agent but also all the other destination agents.

4 Autonomic Group Protocol

4.1 Consistent combination of classes

Each autonomous group (AG) agent takes a collection of classes for protocol functions to communicate with the other agents. In this paper, we consider significant protocol functions, coordination, transmission, confirmation, and retransmission functions. Let **F** be a set of the significant protocol functions $\{C(\text{coordination}), T(\text{transmission}), CF(\text{confirmation}), R(\text{retransmission})\}$. For each protocol function F in **F**, $Cl(F)$ shows a set of classes each of which shows one way of implementation of the protocol function F. For example, $Cl(C) = \{C(centralized), D(distributed)\}$. Table 1 shows classes for protocol functions.

We rewrite **F** to be a set $\{F_1, F_2, F_3, F_4\}$ of protocol functions where $\langle\ F_1, F_2, F_3, F_4\ \rangle = \langle\ C, T, CF, R\ \rangle$. 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 a *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). We discuss what p instance each agent can take to communicate with th agents.

Table 1. Protocol classes.

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

As discussed in the preceding section, the destination retransmission scheme can be taken in the distributed confirmation scheme but not in the centralized one. A protocol instance $\langle\, c_1, c_2, c_3, c_4 \,\rangle$ is referred to as *consistent* iff an agent taking the instance can operate. If an agent takes an inconsistent protocol instance, the agent cannot work. Thus, only some protocol instances of function classes are consistent. An agent can take only a consistent protocol instance. Table 2 summarizes possible protocol profiles. A protocol $profile$ is a consistent protocol instance which each agent can take. Protocol profiles are shown in Table 2. A 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, direct transmission, direct confirmation, and sender retransmission. Let **P** be a set of the protocol profiles which are show in Table 2.

4.2 Consistent set of profiles

Suppose autonomous group (AG) agents p_1, ..., p_n are in a view V of a group G. Let C_i show a consistent protocol instance, i.e. protocol profile taken by an agent p_i, $C_i = \langle\, c_{i1}, ..., c_{i4} \,\rangle \in$ **P**. A $global$ protocol instance C for a view $V = \{p_1, ..., p_n\}$ is a tuple $\langle\, C_1, ..., C_n \,\rangle$ where each C_i is a protocol profile which an agent p_i takes. Here, each C_i is referred to as $local$ protocol instance of an agent p_i (i = 1, ..., n). In traditional protocols, every agent has to take a same local protocol instance, i.e. $C_1 = \cdots = C_n$. Hence, if some agent p_i would like to change a class c_{ik} of a protocol function F_k with another one c_{ik}', all the agents have to be synchronized to make consensus on a new protocol instance. A global protocol instance $C = \langle\, C_1, ..., C_n \,\rangle$ is referred to as $complete$ if $C_1 = \cdots = C_n$. If $C_i \neq C_j$ for some pair of agents p_i and p_j, a global protocol instance $C = \langle\, C_1, ..., C_n \,\rangle$ is $incomplete$. A global protocol instance $C = \langle\, C_1, ..., C_n \rangle$ is $consistent$ if a collection of agents where each agent p_i takes C_i can be cooperating. A global protocol profile is a consistent global protocol instance. It is trivial a complete global protocol instance is consistent. In this paper, we discuss a group protocol where a view of agents p_1, ..., p_n can take an $incomplete$ global protocol instance $C = \langle\, C_1, ..., C_n \,\rangle$. First, suppose that a global protocol instance $C = \langle\, C_1, ..., C_m \,\rangle$ is complete and some agent p_i changes a local protocol instance C_i with another one C_i'. We discuss whether or not a global protocol instance $\langle\, C_1, ..., C_{i-1}, C_i', C_{i+1}, ..., C_n \,\rangle$ is consistent, i.e. all the agents p_1, ..., p_n can cooperate even if $C_i' \neq C_j$ for some agent p_j.

According to change of network QoS and application requirement, each agent autonomously changes the protocol profile. For example, suppose an agent p_3 belongs to a pair of views V_1 and V_2 [Figure 8]. In the view V_1 where all of the agents take DDDirS, an agent p_1 sends a message m to all the other agents. On receipt of the message m, an agent p_3 with DDDirS forwards the message m to the other agents p_5 and p_6 which belong to another view V_2 with DDDisD. Here, the agent p_3 can receive the receipt confirmation of the message m from a pair of agents p_5 and p_6 in the view V_2. In addition, the agent p_3 sends back the receipt confirmation of the message m to the original sender agent p_1. Here, the original sender agent p_1 can receive the receipt confirmation from all the destination agents in the view V_1. Therefore, the agent p_3 does not need to change the profile since the agent p_3 can forward the message m to another agent in the view V_2.

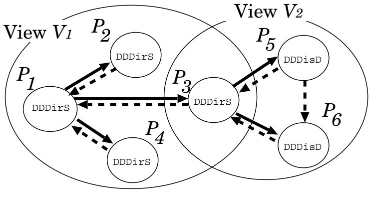


Figure 8. Change of profiles.

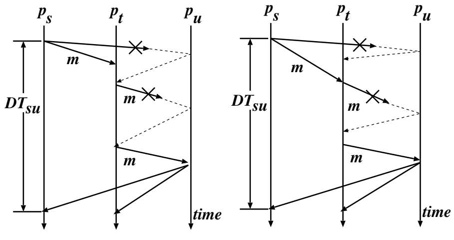
5 Retransmission

We discuss how an autonomous group (AG) agent can autonomously change the retransmission classes in a group as an example.

5.1 Cost model

Suppose there are three autonomic group (AG) 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 the message m while another agent p_u fails to receive m. Here, p_u is referred to as $faulty$. The following notations are used to discuss a cost model for a pair of agents p_s and p_t:

1. d_{st} = delay time between agents p_s and p_t [msec].
2. f_{st} = probability that a message is lost.
3. b_{st} = bandwidth [bps].



A. $d_{st} \le d_{su} + d_{tu}$ **B.** $d_{st} \ge d_{su} + d_{tu}$

Figure 9. Destination retransmission.

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 the agent p_s sends a message m. Then, the agent 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 be transmitted to deliver a message m to a faulty destination p_u are given as follows:

1. ST_{su} = ($2d_{su}$ + $|m|$ / b_{su}) / ($1 - f_{su}$).
2. SN_{su} = 1 / ($1 - f_{su}$).

In the destination retransmission, some destination p_t forwards the message m to the agent p_u [Figure 9

Table 2. Protocol profiles.

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

expected time DT_{su} and number DN_{su} of messages to deliver a message m to p_u are given as follows:

1. $DT_{su} = (d_{su} + |m| / b_{su} + d_{ut}) + (2d_{ut} + |m| / b_{ut}) / (1 - f_{ut})$ if $d_{st} \leq d_{su} + d_{ut}$.

   $DT_{su} = (d_{st} + |m| / b_{st}) + (2d_{ut} + |m| / b_{ut}) / (1 - f_{ut})$ otherwise.
2. $DN_{su} = 1 + 1 / (1 - f_{ut})$.

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 lost 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, the agent p_t obtains the QoS information by periodically sending QoS information messages to all the agents in a view. The agent p_t maintains the quality of service (QoS) information in a variable Q of QB where $Q_{ut} = \langle b_{ut}, d_{ut}, f_{ut} \rangle$ for u = 1, ..., n. If the agent p_t receives QoS information from another agent p_s, $Q_{su} = \langle b_{su}, d_{su}, f_{su} \rangle$ for u = 1, ..., n.

5.2 Change of retransmission scheme

Suppose an agent p_s sends a message m and every agent p_t take the sender retransmission scheme, $C_t = \langle \cdots, S \rangle$. As shown in Figure 10, an agent p_u fails to receive the message m. According to the change of QoS supported by the underlying network, the sender agent p_s makes a decision to change the retransmission scheme with the destination one, say an agent p_t forwards the message m to the agent p_u. However, the agent p_t still takes the sender retransmission. Here, no agent forwards the message m to p_u.

Next, suppose all the agents is taking the destination retransmission scheme. Here, QoS supported by the network is changed and the agent p_t decides to take the sender retransmission scheme. However, no agent forwards the message m to the agent p_u since the sender agent p_s still takes the destination retransmission scheme. 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 [Figure 10].
2. If an agent p_t detects that a destination agent p_u has not received the message m, p_t selects a retransmission scheme which p_t considers to be optimal based on the QoS information Q.
   - 2.1 If p_t is a destination agent and changes a retransmission scheme, p_t forwards m to p_u and sends $Retx$ message to the sender agent p_s.
   - 2.2 If p_t is a sender of a message m and takes a sender retransmission scheme, p_t retransmits m to p_u. If p_t takes a destination retransmission scheme, p_t waits for $Retx$ message from a destination. If p_t does not receive $Retx$, p_t retransmits m to p_u.

It is straightforward for the following theorem to hold from the definition.

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

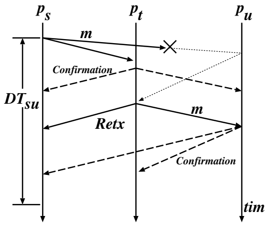


Figure 10. Retransmission.

6 Evaluation

We evaluate the autonomic group protocol (AGP) in terms of delivery time of a lost message. We make the following assumptions on this evaluation.

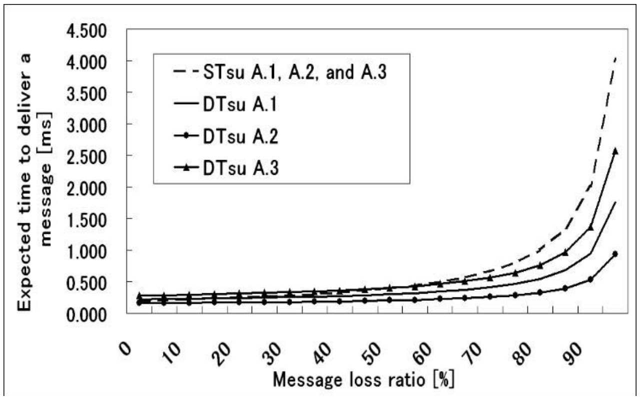
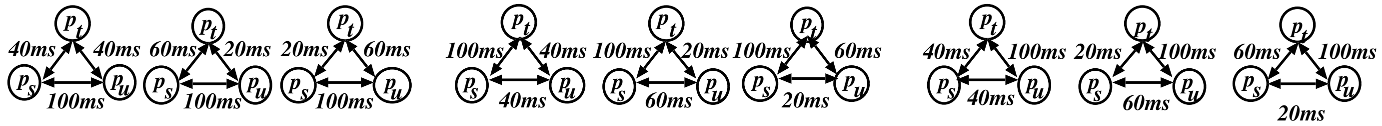


Figure 11. $d_{su} \geq d_{st} + d_{ut}$.

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 although messages may be lost.

Let us consider a view $V = \{p_s, p_t, p_u\}$ where every agent takes a profile DDDisS, distributed control, direct transmission, distributed confirmation, and sender retransmission. Here, suppose that an agent p_s sends a message m to a agents p_t and p_u in a view V. Then, the agent p_t recei message m while another agent p_u fails to receive m.



1. $dst = dtu$ 2. $dst > dtu$ 3. $dst < dtu$

A. $dsu \geq dst + dtu$

1. $dsu = dtu$ 2. $dsu > dtu$ 3. $dsu < dtu$

B. $dsu \leq dst + dtu$, $dst > dsu$ and $dst > dtu$

1. $dsu = dst$ 2. $dsu > dst$ 3. $dsu < dst$

C. $dsu \leq dst + dtu$, $dtu > dsu$ and $dtu > dst$

Figure 12. AG agent graph.

the sender p_s and destination p_t detect the destination agent p_u fails to receive the message m, the agents p_s and p_t autonomously select a retransmission scheme 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 12 shows an AG agent graph for the view 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 $d_{su} \geq d_{st} + d_{ut}$. There are further cases: $d_{st} = d_{ut}$ [Figure 12 A.1], $d_{st} > d_{ut}$ [Figure 12 A.2], and $d_{st} < d_{ut}$ [Figure 12 A.3]. Figure 11 shows the expected time DT_{su} for three cases. In Figure 11, horizontal axis shows a message loss probability of f_{su} and f_{ut}. For case of Figure 12 A.2, $DT_{su} < ST_{su}$. For case of Figure 12 A.1, $DT_{su} < ST_{su}$ if $f_{su} > 15\%$ and $f_{ut} > 15\%$. For case of Figure 12 A.3, $DT_{su} < ST_{su}$ if $f_{su} > 50\%$ and $f_{ut} > 50\%$.

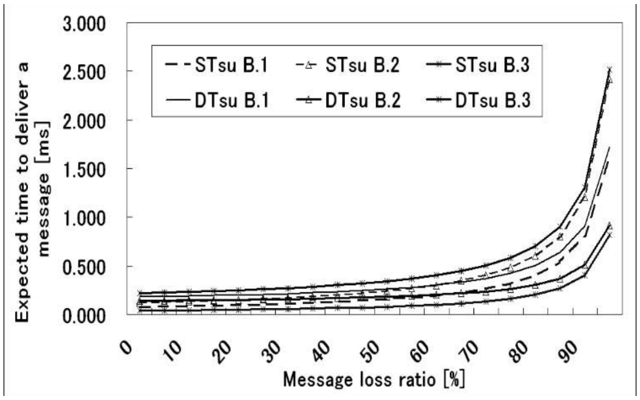


Figure 13. $d_{su} \leq d_{st} + d_{ut}$, $d_{st} > d_{su}$, and $d_{st} > d_{ut}$.

Next, we consider a case $d_{su} \leq d_{st} + d_{ut}$. There are further following cases [Figure 12]:

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 12 B and 12 C] is shown for these six cases in Figures 13 and 14. For cases of Figure 12 B.1 and B.3, $DT_{su} > ST_{su}$. For case of Figure 12 B.2, $DT_{su} < ST_{su}$ if $f_{su} > 20\%$ and $f_{ut} > 20\%$. For case of Figure 12 C, $DT_{su} > ST_{su}$.

7 Concluding Remarks

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

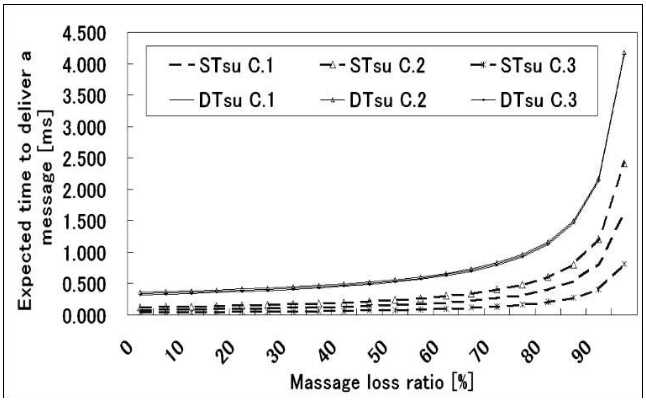


Figure 14. $d_{su} \leq d_{st} + d_{ut}$, $d_{ut} > d_{su}$, and $d_{ut} > d_{st}$.

References

[1] P. E. Agre. P2p and the promise of internet equality. *Communication of the ACM*, 46(2):39–42, 2003.

[2] S. A. Birman, K. and S. P. Lightweight causal and atomic group multicast. *ACM Trans. on Computer Systems*, 9(3):272–290, 1991.

[3] S. Deering. Host groups: A multicast extension to the internet protocol. *RFC 966*, 1985.

[4] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[5] IBM Corporation. Autonomic computing architecture : A blueprint for managing complex computing environments. 2002. http://www-3.ibm.com/autonomic/pdfs/ACwhitepaper 1022.pdf.

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

[7] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *CACM*, 21(7):558–565, 1978.

[8] F. Mattern. Virtual time and global states of distributed systems. *Parallel and Distributed Algorithms*, pages 215–226, 1989.

[9] D. L. Mills. Network time protocol. *RFC 1305*, 1992.

[10] A. Nakamura and M. Takizawa. Reliable broadcast protocol for selectively ordering pdus. *Proc. of IEEE ICDCS-11*, pages 239–246, 1991.

[11] C. Steketee, W. P. Zhu, and P. Moseley. Implementation of process migration in amoeba. *Proc. of IEEE ICDCS-14*, pages 194–201, 1994.

[12] R. van Renesse, K. P. Birman, and S. Maffeis. Horus: A group communication system. *CACM*, 39(4):76–83, 19