

Byzantine Agreement Protocol using Hierarchical Groups

Hiroyuki Yoshino, Satoshi Kawanami, Tomoya Enokido, and Makoto Takizawa
Tokyo Denki University, Japan
E-mail {yos, kawa, eno, taki}@takilab.k.dendai.ac.jp

Abstract

Peer-to-peer information systems are composed of large number and various types of peer processes interconnected in networks like the Internet. Peers may suffer from not only stop fault but also attacks like intrusion and hacking. We have to discuss how to make a system tolerant of Byzantine faults of processes. Even Byzantine agreement protocols with message authentication imply large communication overhead $O(n^t)$ for the number n of processes and the maximum number t of faulty processes. In order to reduce the overhead, we consider a hierarchical group composed of subgroups. Each subgroup shares at least one correct process with some subgroups in order to be tolerant of faults of leaders in each subgroup. Even if a leader process of a subgroup is faulty, all the correct processes can make an agreement on the correct value in a whole group. We evaluate the protocol in terms of number of messages and rounds.

階層型グループを用いたビザンティン合意プロトコル

吉野 宏征 河浪 悟士 榎戸 智也 滝沢 誠
東京電機大学大学院理工学研究科情報システム工学専攻
E-mail {yos, kawa, eno, taki}@takilab.k.dendai.ac.jp

Peer-to-Peer 型のアプリケーションのように数千から数万のプロセスが協調動作を行う環境では、停止障害に加えてビザンティン障害を考える必要がある。ビザンティン合意プロトコルは、合意に達するための通信と処理負荷が大きく、実現が困難である。本論文では、階層化したグループを用いて、プロセスのビザンティン障害に対処し、合意に必要なメッセージ数と時間を減少させる方法を提案する。

1. Introduction

In peer-to-peer (P2P) systems [8], a collection of multiple processes are cooperating to achieve some objectives. The collection of the cooperating peer processes is referred to as *group*. Processes in a group are exchanging messages with each other in a network. A group of multiple processes make an agreement to realize the cooperation of the processes. Peers do not correctly behave due to attacks of intrusion, hijacking, spoofing, falsification of data [10]. Thus, some peers suffer from not only stop fault but also Byzantine faults [7].

In this paper, we assume the underlying network is reliable and synchronous [3]. We discuss how to realize the agreement among peer processes in a scalable group in presence of Byzantine faults of processes. Let n be the number of processes and t ($\leq n$) be the maximum number of faulty processes in a group of processes. In Byzantine agreement protocols [7], a leader process distributes a value v to all the member processes. On receipt of the value v , each member process forwards the value to the other member processes. Each process in turn receives values forwarded by other processes. Here, if some process is faulty, the faulty process sends a value v' different from v or does not send any value. This is one round. After exchanging the values in $t+1$ rounds, each correct process takes a majority value out of the values received. All correct processes can make an agreement only if

$n \geq 3t + 1$. Even in Byzantine agreement protocols with message authentication [4], the communication overhead is $O(n^t)$. Due to the large overhead, it is difficult, maybe impossible to realize the Byzantine agreement protocol. In this paper, we introduce a hierarchical group which is tolerant of Byzantine faults of processes to reduce the overhead of the Byzantine agreement protocol. In each subgroup, the Byzantine agreement protocol is adopted to make an agreement among all the correct processes. If a leader process of a subgroup is faulty, correct processes cannot agree on the correct value while the correct processes can agree on some value in the subgroup. In order to resolve this problem, each subgroup shares at least one correct process named *gateway* process with at least one subgroup. Even if the leader is faulty in a subgroup, at least one gateway process can forward the correct value obtained in another subgroups to all the correct processes in the subgroup.

In section 2, we overview Byzantine agreement protocols. In section 3, we discuss the hierarchical Byzantine agreement protocol. In section 4, we evaluate the protocol.

2. Byzantine agreement protocol

A system is composed of processes interconnected in a network. We assume the network to be reliable and synchronous [3]. In addition, the maximum delay

time between every pair of processes is bounded. A process is *correct* if and only if (iff) the process behaves only according to the specification. Otherwise, the process is *faulty*. There are following types of process faults [4]:

1. Crash (or stop) fault: A faulty process stops prematurely and does nothing.
2. Omission fault: A faulty process stops prematurely, or intermittently omits to send or receive messages, or both.
3. Byzantine (arbitrary) fault: A process can exhibit arbitrary behavior.

In peer-to-peer (P2P) systems, large number of processes in peer computers, mainly personal computers interconnected with the Internet. A group of $n (> 1)$ peer processes p_0, p_1, \dots, p_{n-1} are required to make an agreement on a value in presence of process faults in distributed applications, e.g. synchronization of processes and consensus in groupware. The agreement protocol is briefly described as follows:

1. At first, a process p_0 sends some value v to all the other processes p_1, \dots, p_{n-1} . The process p_0 is referred to as *leader* one. The others p_1, \dots, p_{n-1} are *member* processes.
2. Every correct member process receiving a value v from the leader process checks if all the other correct processes receive the same value v by exchanging values with each other.

Here, not only a member process but also a leader process may be faulty. A faulty process may send different values to different processes and may not send any value to some process. A group of multiple processes are referred to as make a *Byzantine agreement* iff both of the following conditions are satisfied [7]:

- IC1. All correct processes agree on a same value.
- IC2. If a leader process is correct, every correct process agrees on the value which the leader sends.

The second condition IC2 is required to hold since even a leader process may be faulty. If a leader process is faulty, all the correct processes agree on some *bottom* value \perp . If the leader process is correct, every correct process is required to make an agreement on the value which the leader process sends. In the agreement protocols, a leader process first sends a value to all the member processes. On receipt of a value from the leader process, each process forwards the value to all the other processes. Then, each process p_i takes a majority value v_i out of the values received. This is the first round. In one round, a leader process sends a value, member processes receive the value, and processes do some computation. Every process synchronously takes a round. Next, each member process p_i distributes to $(n-2)$ processes $p_1, \dots, p_{i-1}, p_{i+1}, \dots, p_{n-1}$ as a leader process. At the $(t+1)$ th

round, a process p_i takes a majority value v_i among the values which the process has received. In each round, a process p_i stores a value received from another process p_j in a variable v_j and a value received from a leader is v_i . $V_i = \{v_1, \dots, v_{n-1}\}$.

Let $majority(V)$ be a function which takes a majority value v in a set V of values. If there is no majority value in a set V , $majority(V) = \perp$. Let t be the maximum number of faulty processes in a group of n processes ($t < n$). The Byzantine agreement can be realized in a group of n processes only if $n > 3t + 1$ [7]. The number $O(n^t)$ of messages are exchanged and it takes $t+1$ rounds to make an agreement among all the correct processes.

In order to reduce the overhead, the Byzantine agreement with message authentication (BA) protocols [7] are discussed. Each process signs a message with its unforgeable signature for sender authentication and then sends the message. Let $x:i$ denote the value x signed by a process p_i . A notation $x:i_1:\dots:i_k:j$ stands for $(x:i_1:\dots:i_k):j$. Suppose a faulty process p_i sends a value v' to a process p_k after receiving a value $v:j$ from a process p_j . On receipt of a value $v':i$ from the process p_i , the process p_k detects that p_i is faulty since the p_j 's sign on the value v' is forged. On receipt of a value, each process p_i accumulates the value to a variable V_i if the value is properly signed. The BA protocol is briefly presented as follows:

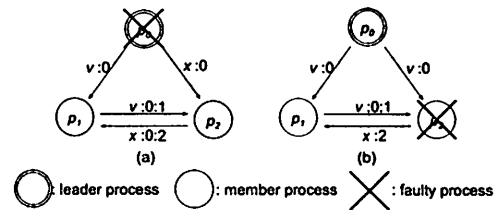


Figure 1. Byzantine agreement with message authentication (BA).

1. A leader process p_0 signs a value v and sends the signed value $v:0$ to all the member processes p_1, \dots, p_{n-1} .
2. On receipt of a value $v:j_0:j_1:\dots:j_k$ from a process p_{j_k} ($k \geq 0, j_0 = 0, j_h \in \{1, \dots, n-1\}$), if the value v is proper, i.e. v is not forged.
 - i. a process p_i adds the value v to V_i ;
 - ii. if $k < t$, p_i sends the signed value $v:0:j_1:\dots:j_k:i$ to every other process than $p_0, p_{j_1}, \dots, p_{j_k}$.

else (v is not correct) p_i perceives p_j to be faulty.
3. If p_i receives no more message, a single element $v (= majority(V_i))$ is obtained from V_i .

At step 3, each correct process p_i takes a majority value in the set V_i by using the function *majority*. Figure 1 shows three processes p_0 , p_1 , and p_2 where one of the processes is faulty. In case (a), the leader process p_0 is faulty. The leader p_0 sends different values v and x to a pair of member processes p_1 and p_2 , respectively. On receipt of the values, the member processes p_1 and p_2 forward the values $v:0:1$ and $x:0:2$ to p_2 and p_1 , respectively. The process p_1 receives values $v:0$ from p_0 and $x:0:2$ from p_2 . Here, p_1 finds the leader p_0 to be faulty since the p_0 's signature on the value $x:0:2$ is not forged by p_2 . Here, p_1 takes a value \perp . The other correct process p_2 takes \perp in a same way as p_1 .

In case (b), a member process p_2 is faulty. The leader process p_0 sends a signed value $v:0$ to the processes p_1 and p_2 . Then, the faulty process p_2 sends a value x to p_1 . Here, p_1 finds p_2 to be faulty. If at most one process is faulty, the process p_1 can take the value v because p_0 must be correct. If values are not signed, the process p_1 cannot decide which process p_0 or p_2 is faulty even if at most one process is faulty, i.e. $t = 1$. Thus, the BA protocol is required to satisfy the following property for a group of n processes:

A1. At least two correct processes exist in a group for reaching an agreement, i.e. $n > t + 1$.

In the BA protocol, each process sends a value to $(n - i)$ processes at the i th round. Hence, it takes $t + 1$ rounds and $(n - 1)(n - 2) \dots (n - t - 1)$ messages are transmitted at round $(t + 1)$ in a group. The computation and communication overheads are too large to realize the Byzantine agreement protocol in a large-scale group.

3. Hierarchical Byzantine Agreement

3.1 Hierarchical group

A group G is a collection of n peer processes p_0, p_1, \dots, p_{n-1} . Every pair of processes are assumed to reliably communicate with one another with the same bandwidth in a network. A process p_0 is a global leader process and the other processes are members in the group G . A set of $n - 1$ member processes p_1, \dots, p_{n-1} are partitioned into *subgroups* G_1, \dots, G_s ($s > 1$). Here, each subgroup G_i is composed of processes $p_{i0}, p_{i1}, \dots, p_{i,k_i-1}$ where $p_{ij} \in \{p_1, \dots, p_{n-1}\}$ ($j = 0, 1, \dots, k_i - 1$). A *root subgroup* G_0 is a collection of the processes $p_0, p_{10}, \dots, p_{s0}$ where p_{i0} is the local leader process of a subgroup G_i ($i = 1, \dots, s$). A Byzantine agreement with message authentication (BA) protocol is used for making a Byzantine agreement among k_i processes in a subgroup G_i .

The leader process p_0 first sends a value v to all the member processes p_{10}, \dots, p_{s0} . In each subgroup G_i ($i = 1, \dots, s$), a global leader process p_{i0} forwards a value

to all the member processes $p_{i1}, \dots, p_{i,k_i-1}$. On receipt of a value v from another process, a process p_{is} forwards the value to other processes to make an agreement on the value v sent by the global leader process p_0 . The communication and computation overheads for making a Byzantine agreement depend on the total number n of processes. Every subgroup includes the same number of processes so that the communication and computation overheads are uniformly distributed to all the processes. Each subgroup G_i can be furthermore composed of subgroups G_{i1}, \dots, G_{is_s} ($s_i > 1$). In this paper, we discuss a two-layered group for simplicity, i.e. a root subgroup G_0 with local subgroups G_1, \dots, G_s .

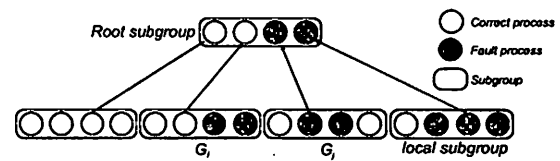


Figure 2. Two-layered group.

First, a global leader process p_0 sends a value v to all the member processes p_{10}, \dots, p_{s0} in a root subgroup G_0 . By using the BA protocol, each correct process p_{i0} obtains some value v on which all the correct processes agree in the root subgroup G_0 . Here, " $s > t + 1$ " is required to hold to make an agreement for maximum number t of faulty processes in the group G . If a process p_{i0} is correct, p_{i0} notifies all the processes of the agreed value v as a local leader process in a subgroup G_i . BA protocol in each subgroup G_i .

Next, suppose that a local leader process p_{i0} is faulty in a subgroup G_i . Here, a subgroup whose leader process is correct is referred to as *correct*. Otherwise, the subgroup is *faulty*. Suppose that every correct leader process makes an agreement on a value v in a root subgroup G_0 by using the BA protocol. A faulty local leader process p_{i0} may send another value different from the value v or may not send any value to each member process in the subgroup G_i . All the correct processes in a faulty subgroup G_i make an agreement on a bottom value \perp although the correct processes agree on the value v in every correct subgroup. We have to discuss how every correct process makes an agreement on the value v even in a faulty subgroup.

3.2 Shared subgroups

In order to resolve the inconsistent agreement in a hierarchical group, each subgroup shares at least one correct process named *gateway* process with at least one subgroup. Suppose a process p is included in both subgroups G_i and G_j as shown in Figure 3. Here, p

is a gateway of G_i and G_j , G_j and G_k are referred to as participated by the gateway p . Suppose the process p obtains a value v on which every correct process agrees in one subgroup G_i . In another subgroup G_j , a local leader process p_{j0} is faulty and sends a value $v' (\neq v)$ to the gateway process p . Through the Byzantine agreement with message authentication (BA) protocol in the subgroup G_j , the gateway process p finds the local leader process p_{j0} to be faulty and takes a value \perp . Here, every correct process including the process p knows which process is correct in the subgroup G_j . The gateway process p is referred to as *inconsistent* if one of the subgroups participated by p is faulty and p takes different values in at least a pair of the subgroups. Here, the process p sends the value v , on which p agrees in the subgroup G_i , to all the correct processes in G_j . Then, every correct process makes an agreement on the value v by the BA protocol. Each subgroup G_i is required to include more number of processes than $t+1$, i.e. $k_i > t+1$.

In addition, if a leader process of every subgroup is faulty, the Byzantine agreement cannot be realized. Hence, at least one subgroup is required to be correct, i.e. $s \geq t+1$.

If a pair of subgroups G_i and G_j are sure to share at least one correct process, the subgroups G_i and G_j are referred to as *directly connected* ($G_i \Leftrightarrow G_j$). Since the leader process of the subgroup G_i is faulty, the maximum number of faulty member processes in all the subgroups is $(t-1)$. Hence, if $|G_i \cap G_j| \geq t$, G_i and G_j are directly connected ($G_i \Leftrightarrow G_j$). In addition, G_i and G_j are *connected* ($G_i \leftrightarrow G_j$) if $G_i \Leftrightarrow G_j$ or $G_i \leftrightarrow G_k \leftrightarrow G_j$ for some subgroup G_k . If a pair of subgroups G_i and G_j share some number of processes, G_i and G_j are referred to as *intersect*. G_i and G_j are *related* ($G_i \equiv G_j$) if G_i and G_j intersect but are not directly connected. In Figure 3, $G_i \Leftrightarrow G_j$ if $t = 2$. Suppose $G_i \Leftrightarrow G_j \Leftrightarrow G_k$. If $(G_i \cap G_j) \cap (G_j \cap G_k) = \emptyset$, G_i and G_k are referred to as independently connected with G_j .

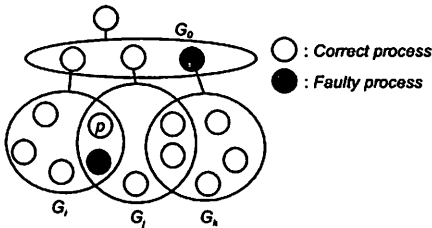


Figure 3. Shared subgroups ($t=2$).

In this paper, we make following assumptions:

[Assumptions]

1. Each process can send signed messages directly to every other process in a subgroup.

2. A faulty process can be detected by the message authentication.
3. No communication fault occurs.

3.3 Agreement protocol

We discuss how to make a Byzantine agreement on subgroups G_1, \dots, G_s for a collection of n processes p_0, p_1, \dots, p_{n-1} . A process p_0 is a leader, i.e. p_0 first sends a value to every leader process p_{i0} of every subgroup G_i ($i = 1, \dots, s$). Here, let G_0 be a root subgroup which is a collection of the leader process p_0 and the leader processes of subgroups, $G_0 = \{p_0, p_{10}, \dots, p_{s0}\}$.

Processes in a group G make a Byzantine agreement by the following protocol:

[Hierarchical agreement protocol]

1. A leader process p_0 sends a signed value to a process p_{i0} of every subgroup G_i ($i = 1, \dots, s$).
2. The leader processes p_{10}, \dots, p_{s0} exchange values according to the BA protocol. Every correct leader process p_{i0} of G_i agrees on a value v .
3. A leader process p_{i0} sends the value v to all the processes in a subgroup G_i .
4. The processes $p_{i1}, \dots, p_{i,k_i-1}$ make a Byzantine agreement on v by the BA protocol in G_i .
5. If a process p_{it} finds the leader p_{i0} to be correct, p_{it} agrees on v and then terminates.
6. The process p_{it} takes \perp and G_i is faulty if p_{i0} is faulty. The process p_{it} waits for a correct value from another correct member process.
7. If p_{it} is an inconsistent gateway process among the faulty subgroup G_i and another correct subgroup G_j , p_{it} distributes the value v on which p_{it} agrees in G_j to all the processes in the faulty subgroup G_i by the BA protocol.
8. A correct process p_{it} agrees on the value v in G_i . If p_{it} is still an inconsistent gateway with another faulty G_j , p_{it} forwards the value v to G_j at step 7.

[Theorem] A group G is composed of subgroups G_1, \dots, G_s . The Byzantine agreement is realized in the group G if the following conditions are satisfied:

1. $s > t$.
2. $k_i > t+1$ for every subgroup G_i .
3. Every pair of subgroups G_i and G_j are connected ($G_i \leftrightarrow G_j$).
4. Every gateway process is not a leader process in each subgroup.

[Proof] The first condition means there exists at least one correct leader of a subgroup. Hence, if a leader process p_0 is correct, every correct process makes an agreement on a value sent by p_0 in at least one subgroup G_i . From the second condition, the Byzantine

agreement can be realized in each subgroup. Hence, the correct value agreed in a correct subgroup G_i is propagated to all the subgroups. Since message authentication is used in every subgroup, every correct process can detect which process is faulty. After the agreement procedure initiated by the faulty leader, every correct member process take a proper value from a correct gateway. \square

3.4 Design of shared subgroups

We discuss how to construct a hierarchical group for a collection G of n processes p_0, p_1, \dots, p_{n-1} and maximum number t of faulty processes. We assume each subgroup G_i includes the same number k of processes so that every process spends a same amount of computation resource to make a Byzantine agreement. Subgroups G_0, G_1, \dots, G_s are constructed for the group G as follows:

1. A root subgroup G_0 is a collection $\{p_0, p_1, \dots, p_s\}$ of processes where p_0 is a leader process.
2. A subgroup G_i is a collection $\{p_i, p_{s+1+h(i-1)}, \dots, p_{s+h(i-1)+k-1}\}$ of processes ($i = 1, \dots, s$) where p_i is a leader process. Here, the suffix “ $s+1+x$ ” stands for “ $s+1+x \bmod (n-s-1)$ ”.
3. For a subgroup G_s , $s+1+h(s-1) \leq n$ and $s+h(s-1)+k-1 \geq n$, i.e. $(n+h-1)/(h+1) \geq s \geq (n+h-k+1)/(h+1)$. If every subgroup includes the same number of processes, $k = s \cdot (n+h+1)/(h+1) \geq s \geq (n+h+1)/(h+2)$.

The total number of effective processes in the group G with subgroups G_1, \dots, G_s is the summation of numbers of processes in subgroups, i.e. $(s+1) + s \cdot k = s(k+1) + 1$. The redundancy factor r_G for a group G is $[(s+1) + s \cdot k]/n$. In a non-hierarchical group G , $r_G = 1$. The redundancy factor shows additional overhead of each process in a hierarchical group. A pair of subgroups G_i and G_j include $[k-1 + h(i-j)]$ common processes $p_{s+1+h(j-1)}, \dots, p_{s+h(i-1)+k-1}$ if $h(i-j) + k - 1 > 0$.

[Theorem] A pair of subgroups G_i and G_j are directly connected ($G_i \leftrightarrow G_j$) if $h(i-j) + k - 1 \geq t$.

[Proof] If the leader process of every subgroup is correct, every correct process can make an agreement on a correct value. Suppose some leader process is faulty in a subgroup G_i . The maximum number of the other faulty processes is $t-1$. Hence, if $|G_i \cap G_j| \geq t$, it is sure at least one correct common process among G_i and G_j . \square

That is, if a leader process is correct in one of subgroups G_i and G_j , every correct process in each of G_i and G_j can make an agreement on the same value. Let us consider a group of 17 processes ($n = 17$). Figures 4 and 5 show hierarchical groups which include four subgroups G_1, G_2, G_3 and G_4 ($s = 4$) for $t = 1$ and $t = 2$, respectively. Each subgroup includes k ($k =$

5) and six ($k = 6$) processes, respectively. The redundancy factors are $r_G = 1.47$ in Figure 4 and $r_G = 1.56$ in Figure 5. In the groups shown in Figures 4 and 5, $h = 3$. The Byzantine agreement is realized for $t = 1$ and $t = 2$, respectively. $G_1 \leftrightarrow G_2 \leftrightarrow G_3 \leftrightarrow G_4$. G_1 and G_2 shares two processes in Figures 4 and 5, respectively.

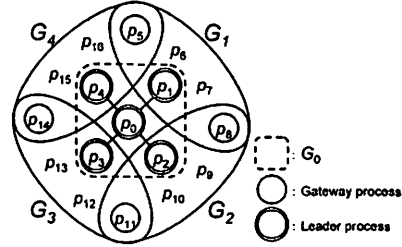


Figure 4. Shared subgroups ($t = 1$).

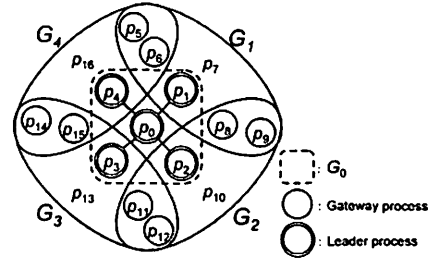


Figure 5. Shared subgroups ($t = 2$).

4. Evaluation

We evaluate the Byzantine agreement protocol for a shared hierarchical group G compared with the traditional Byzantine agreement protocol with message authentication in terms of how many messages are transmitted and how long it takes to make a Byzantine agreement among processes. Suppose the number of messages exchanged in each subgroup of k processes is given in a function $N(k, t) = (k-1)(k-2)\dots(k-t-1)$ for the maximum number t of faulty processes in the group G .

The minimum number of messages exchanged in the hierarchical group is $N(s, t) + s \cdot N(k, t)$. Here, a leader process of every subgroup is correct. It takes $(t+1)$ rounds in the root subgroup G_0 and $(t+1)$ rounds in each subgroup. Hence, it takes $2(t+1)$ rounds to make an agreement.

In the worst case, only one subgroup has a correct leader and the other subgroups are faulty. In addition, each subgroup has at least one and at most two subgroups which are directly connected with the subgroup, i.e. linearly chained as shown in Figure 6.

Here, only a subgroup G_1 is correct and the other subgroups G_2, \dots, G_s are faulty. Every correct process agrees on a value v in a subgroup G_1 . First, a correct gateway process between G_1 and G_2 forwards the value v in the subgroup G_2 . Then, a gateway process of G_2 and G_3 distributes the value v in G_3 and finally in G_s . Thus, $(s-1)N(k, t-1)$ messages are transmitted. Totally, $N(s, t) + s \cdot N(k, t) + (s-1) \cdot N(k, t-1)$ messages are transmitted. Here, it takes $2(t+1) + (s-1)t$ rounds. The worst case is shown in Figure 6.

The redundancy factor $[(s+1) + s \cdot k]/n$ shows the total processing overhead. We assume the processing overhead of each process in a subgroup is proportional to the number of messages transmitted, i.e. $N(k, t)$. The total processing overhead is given $(s+1) \cdot N(s, t) + s(k+1) \cdot N(k, t)$ in the best case. $(s+1) \cdot N(s, t) + s \cdot (k+1) \cdot N(k, t) + (s-1) \cdot (k+1) \cdot N(k, t-1)$ in the worst case.

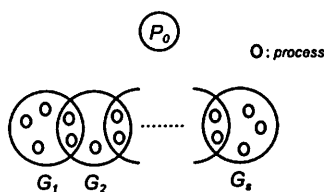


Figure 6. Chained subgroups.

First, we assume $k = s$, i.e. every subgroup includes the same number of processes. For each n and t , a hierarchical group where the numbers of messages and rounds are minimized is found. Figures 7 and 8 show the numbers of messages and rounds in a each minimum subgroup for the total number n of processes and ratio of the number of faulty processes t/n ($r = t/n$). In the hierarchical protocol, the number of messages and rounds can be reduced.

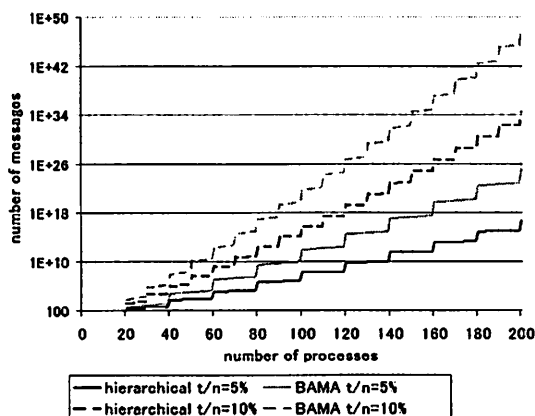


Figure 7. Number of messages.

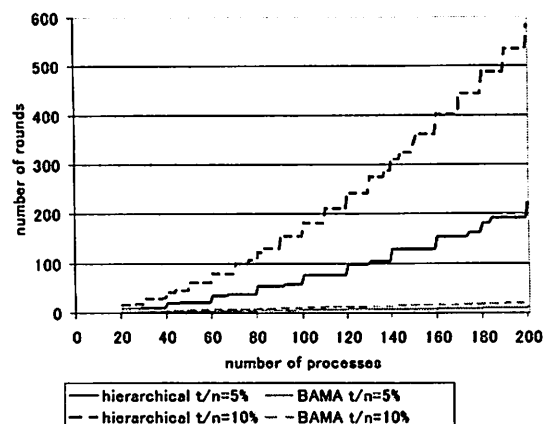


Figure 8. Number of rounds.

5. Concluding Remarks

In this paper, we discussed how to make a Byzantine agreement on the delivery and orderly of messages by using the hierarchical group. We showed how many messages can be reduced to make a Byzantine agreement in the evaluation.

References

- [1] C. Attiya, D. Dolev, and J. Gil. Asynchronous Byzantine Consensus. In *Proc. of the third annual ACM symposium on Principles of distributed computing table of contents*, pages 119–133, 1984.
- [2] D. Dolev and H. R. Strong. Authenticated algorithms for Byzantine Agreement. *SIAM journal on Computing*, 12(4):656–666, 1983.
- [3] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.
- [4] V. Hadzilacos and S. Toueg. Fault-Tolerant Broadcasts and Related Problems. In *Distributed Systems*, chapter 5, pages 97–145. Addison Wesley, 1993.
- [5] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Comm. ACM*, 21(7):558–565, 1978.
- [6] L. Lamport. The Weak Byzantine Generals Problem. *Journal of the Association for Computing Machinery*, Vol. 30(No. 3):668–676, 1983.
- [7] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, Vol. 4(No. 3):382–401, 1982.
- [8] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.
- [9] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of the Association for Computing Machinery*, Vol. 27(No. 2):228–234, 1980.
- [10] S. Young and D. Aitel. *The Hacker's Handbook*. Auerbach, 2004.