Reaching Agreement in Hierarchical Groups*

F. Nguilla Kooh, **S. Haddad**

E-mail: [nguilla,haddad@]lamsade.dauphine.fr
LAMSADE, Université Paris-Dauphine
Place du Maréchal de Lattre de Tassigny
75775 Paris Cédex 16, FRANCE

Abstract

We present a mechanism based on Chandra and Toueg unreliable failure detectors model. This mechanism can be used as building block for distributed systems running on Internet and requiring a non-blocking agreement between their components. The protocol we propose tolerates both local participants and groups of participants' failures in a large-scale and asynchronous environment prone to failures.

1 Introduction

Group [2, 1] and agreement (or simply consensus) paradigms [17] represent fundamental aspects in distributed computing. They are oftenly used in order to ensure the availability or the global consistency of a service in spite of some components' failures. Groupware, brokerage or transactional systems, management of replicated databases are some examples of real applications using these paradigms.

Many algorithms have been proposed to solve the consensus problem in various execution environment models (see for example [5, 7, 4]). However, the great majority of them supposes a mutual knowledge of the participants what limits mainly their application field to local area networks. In previous articles [11, 12, 10], we proposed some algorithms solving the consensus problem in a wide area network. These algorithms are intended to be used in asynchronous environments augmented with "unreliable" failure detectors [4].

One knows indeed that in a completely asynchronous environment the consensus problem is not solvable[8]. This result implies the impossibility of a consistent management of groups [3]. Hence, different complementary assumptions have been proposed. Among the models suggested, that of Chandra and Toueg has the advantage of making assumptions on the behavior of the detectors and not on the underlying network. Thus, various concrete environments can satisfy the conditions associated with the detectors.

As in our preceding works [11, 12, 10], we consider an environment constituted of groups of participants. Within a group, the participants know each other mutually and communicate directly. Each group has a broadcast address known by the participants of the other groups. This is the only possible interface to communicate with this group from outside. Each participant has a detector which tests the status of the other participants of the same group and that of the other groups.

Chandra and Toueg defined several degrees of reliability for the detectors and proposed algorithms for two of them: S and $\diamond S$. Our models of detector are characterized by two behaviors : detection inside a group and detection between groups. Our algorithms suppose an internal behavior of the detectors similar to the class S and a variable external behavior (either equivalent to the class S [11, 12] or to $\diamond S$ [10]). This approach has the interest to combine two classes of detectors, each of them providing the adequate properties to the configuration of network concerned : for example a stronger reliability of failure detection in local area networks (owing to their good characteristics: large bandwidth, predictable transit delay ...) than in the Internet for instance (low data flows, failures of the communications' links, routing problems ...).

In this paper, we propose a generic mechanism solving a consensus problem involving entities organized hierarchically (with more than two levels). This mechanism takes keeps and unifies the principles of our preceding algorithms. Consequently, it works with groups of entities, each entity being able to be either a subgroup of entities or a participant. At the root of this hierarchy, the detectors behave either as S or as $\diamond S$. In

*Supported in part by ESIGETEL, Département de la Recherche, 1, rue port de Valvins. 77215 Fontainebleau-Avon, FRANCE. Contact: nguilla@esigetel.fr

an internal node of the hierarchy the detectors behave such as the class S.

The interest of this new approach is to allow a decision-making of groups of variable granularity resulting from the decisions of the sub-groups or the participants of which they are made up.

The new approach satisfies the usual properties defining the consensus problem. It has the following advantages:

• its authorizes a local decision policy for each sub-group. This local policy enables entities of the expected sub-group to choose the value which will be proposed like sub-group decided value for the protocol's steps of the higher level;

• its involves anonymous sub-groups which communicate through sub-groups or individual addresses;

• the protocol consumes less messages than the originate algorithms by reducing the number of exchanged message and especially the distant communications;

• furthermore, the protocol works not only if "simple" participants fail but also when some sub-groups become unreachable (due to crash or link's failure).

This paper is structured as follow : the second section gives a survey of the group paradigm, the consensus problem and unreliable failure detectors model. The third section presents our model of execution by describing the properties defining the hierarchical problem of consensus and those characterizing our model of failure detector. In section four, we present the generic and hierarchical consensus protocols and its different instanciations. The last section discusses about aspects of implementation and gives some elements of conclusion.

2 Group, Consensus and Unreliable Failure Detectors Paradigms

2.1 The Group Paradigm

A group is a set of processes (or objects) that cooperate towards some common goal or share some global state that is distributed or replicated. Members of a group have a consistent view of the current composition of the group (*group's view*) via a group membership service. They are notified when a member joins or leaves the group (voluntarily or due to its failure). The group membership service tracks this change and transforms it into *group's view* agreed by all correct members.

The group paradigm has traditionally been used for achieving different purposes. [1, 15] : 1) for maintaining a share state (for fault-tolerance, or load balancing); 2) as services with clients; 3) for groupware; 4) for parallel programming or for disseminating information; 5) for network or distributed system management (collecting and propagating management information; 6) for mobility (application or object migration) and so forth. The distributed operating system ISIS [2] was the first to use this paradigm.

Different systems consider either static groups (membership does not change during the system's lifetime) or dynamic groups (where the group's view can change over time). Like in ISIS, we can distinguish different styles of groups : *Peer groups*(when a set of processes cooperate closely); *Client-server groups* (when any process communicate with any group via the group's name and when this process has the appropriate permissions); *Diffusion group* is a client-server group in which the clients register themselves but in which group members send messages to the full set of clients; and *Hierarchical group* is a structure built from multiple component groups, for reasons scale.

2.2 Agreement with Failure Detectors

Initially evoked by Pease et al [17], the consensus problem is now in the heart of the most of agreement problems (group membership, broadcast, Byzantine agreement, *k-set* agreement ...) [14, 13].

Informally, a consensus protocol enables a set of participants to decide on a common value which depends on their initial inputs, despite the crashes or the suspicions of some of them. Informally,

1) to ensure the *Validity* property, a value decided by some participant must have been proposed by some participant;

2) no two participants may decide different values *Uniform Agreement*;

3) and each correct participant eventually decides some value; that ensures the *Termination* of the protocol.

The difficulty to detect failures in a complete and accurate way in purely asynchronous system, make the consensus problem unsolvable in such systems [8].

That leads Chandra and Toueg to introduce the concept of *unreliable failure detectors*. A failure detector is a module that gives to a process hints about failures of the other processes of the systems with which it communicates. The unreliability comes from the fact that it can make mistake by suspecting a correct process or not suspecting a faulty one. One way to control the detectors is to require a kind of *completeness* (i.e. detecting a faulty process) and *accuracy* (i.e. not suspecting a correct process). Completeness enforces the

detection of any faulty process while accuracy restricts the mistakes (i.e. incorrect suspicions) the failure detectors can make.

Each class of failure detector is defined by the same completeness and accuracy properties.

The mechanism we propose uses the classes named $\diamond S^1$ and S^2.

3 Model

We consider a distributed system without timing assumptions (on communication delays or process relative speeds) and running in wide area network such as Internet. We assume that communications are asynchronous and reliable (a message sent to a correct and reachable entity will eventually be received).

The environment is represented by a group g composed of a set of entities organized hierarchically as a tree.

Each entity of the system is either a sub-group of entities or a "simple" participant. Each entity has a reference (identifier of group or participant) so that a participant in this organization can be indicated $g_1.g_2....g_p.i$ where i indicates the identifier of the participant g_p the group containing the participant, g_i the group containing the group g_{i-1} and g_1 is contained in g. Each entity has an address. In the case of a participant, it is its network address; in the case of a group, it is a broadcast address such that any message sent at this address is received by the participants belonging **directly or indirectly** to this group. Further and for simplicity, the broadcast address and the identifier of the entity will indicate the same thing.

• It will be said that a participant p is included in a group G if it belongs **directly or indirectly** to this group. For example, $g_1.g_2....g_p.i$ belongs to g_p and it is included in $g, g_1, g_2..., g_p$

3.1 Failure Detector Model

An entity is suspected or fails by crashing. We do not handle neither Byzantine nor links' failures. A group is faulty when all participants included in this group fail.

Our asynchronous system augmented with unreliable failure detector is close to the one proposed by Chandra and Toueg [4] with the difference that our approach is intended to be use in a WAN.

[1] strong completeness and eventual weak accuracy

[2] strong completeness and weak accuracy

A failure detector mechanism associated to each participant to the consensus protocol. It is a module which gives to the expected participant hints (possibly erroneous) about the operationnality of the entities with which it communicates. More precisely the participant $g_1.g_2....., g_p.i$ communicates with the entities which belong to the groups $g, g_1..., g_p$. This mechanism works like a black box and can be questioned at any time by the participant.

3.1.1 Detectors' Behaviors

Let g_i a group, $e1$ and $e2$ two entities of g_i. We distinguish two levels in our hierarchy. We call *external* level the level constituted of groups representing LANs. These groups are accessible directly through Internet. The *internal* levels represent all entities located in each LAN.

The new class of the detector we have defined is denoted $\prec S, \diamond S / S \succ$. That means in internal level, the class S will be used and between nodes of the external level the class $\diamond S$ or S could be used depending on the strength of the constraints of the systems or the applications. This variable combined failure detector behaves during an execution as follows:

Large Completeness: If $e1$ is a faulty entity (group or participant) then $e1$ will be suspected after a finite time by $e2$ if $e2$ is a participant or by all participant included in $e2$.

Large Accuracy

• In an internal level (i.e. within different entity belong indirectly with g, our detector itself acts like the detector S

For any group $g_i \neq g$ which remains correct during the execution, there exists a correct entity of g_i which will never be suspected faulty by the participants included in this group

• At the external level (i.e. between the entities which belong to g or of higher hierarchical level), the detector can behave like S or $\diamond S$

There is a correct entity belonging to g which will be trusted after a finite time by all participants included in g

4 Generic and Hierarchical Consensus

Our protocol relies on the following idea. A participant of a sub-group sg included in a group g acts in a redundant way such as its sub-group's representative within the group g during the decision-making process. To ensure the consistency of the actions of these participants after the initial phase and after each phase

corresponding to the reception of a message, an internal consensus within sg is launched in order to ensure sub-group consistency in terms of state and actions according to the the global protocol.

We give here the internal algorithm of consensus which is a recursive adaptation of that proposed for the detector $S \diamond S$ and S [12, 10]. The principal parameters handled are : the group g_0 in which the consensus takes place, the path in the tree from the group g_0 the participant $g_1.g_2.....g_p.i$ (indicated in the algorithm by *part*), the value of input (or initial value of an entity) and the identifier of consensus (several consensus are launched within a sub-group). It returns the decided value.

Initially, each participant proposes its value if it belongs to the group or recursively calls the algorithm to decide value of the entity of which it is a representative. The identifier of consensus is a list of values of which the length is equal to the depth of the recursive calls and whose each element represents the asynchronous number of round of the corresponding entity.

After sending a message to a sub-group sg, the psg's members receive directly or indirectly these messages by the broadcast mechanism; each message is time-stamped with the identifier of the consensus in order to avoid any confusion at the reception. Each reception gives place to an update of the local variables before proceeding the next step. As in the same way as initially, the participant which acts as a representative calls a new consensus to ensure the consistency mentioned above. This new consensus is time-stamped with the current identifier suffixed by the round of execution in the consensus.

4.1 Algorithms

Let g_0 a group and $g_1.g_2.....g_p$ sub-groups of entities belonging to g_0. α is g_0's "*estimate*" if α is agreed by all participants which belong directly or indirectly to g_0 (i.e. in $g_1.g_2.....g_p$).

-Phase 0: At the beginning of the protocol, each group chooses its value by the launch of the sub-group consensus involving all participants included in this group. This is achieved by the recursive call of the $S - Consensus$ procedure.

Each group's member proposes for the "group consensus" a value that it wishes to be the group's input value for the wide consensus protocol. At the end of the "group consensus", all group's members agree on a value.

4.1.1 $S \diamond S$-Consensus

At each round, a participant executes partially or totally the four phases that follow with respect to the role it plays during the protocol **as long as a final decision has not been taken**.

The algorithm works in asynchronous rounds :

At each round some sub-group coordinates the whole set of sub-groups. We note this "coordinating sub-group" C_g (equal to the sub-group round number modulo G). A participant which belongs to this sub-group is called a "*coordinating*" participant (noted $P_{(c,p)}$) and the participants of the other sub-groups "*simple participants*".

-Phase 1 : All participants send their group's "*estimate*" to the "*coordinating*" group via its broadcast address. The first estimates are initial groups' inputs.

-Phase 2 : Each participant of the current "coordinating group" gathers $\lceil (G+1) \rceil / 2$ estimates related to the first messages received from $\lceil (G+1) \rceil / 2$ groups. When a member collects this number of *estimates*, it selects the most recent defined. It launches the consensus procedure in order to be synchronized with its peers. After that, any *coordinating* participant sends the *estimate* decided in its group. This estimate is the one with the highest time-stamp (i.e. the highest round)

-Phase 3 : In this phase the "simple participants" send an acknowledgment if they receive the estimate of the coordinating group and a non-acknowledgment if they suspect this group.

-Phase 4 : As phase 2, the coordinating group waits for a majority of acknowledgments or non-acknowledgments. The first participant which reaches this number launches a consensus. If the consensus gives only acknowledgments, the participants decide from the estimate of the expected group.

We consider the point of view of the participant i of a group noted g_0, $g_1.g_2.....g_p.i$ or more simply *part*.

• ADP represents all LANs participating to the consensus; • IdC identifies each execution of a consensus procedure; • $\triangle_{part}$ is a vector containing messages received at different rounds. • $estimate_{part}$ is *part*'s estimate; • r_{part} is p's current round number; • ts_{part} is the last round in which *part* updated $estimate_{part}$, initially set to 0; • r_g is the last round in which the domain g updated $estimate_g$; • $\alpha \in \mathbf{CFD_{part}}$ means that process (g,p) suspects α by querying its combined failure detector **CFD**.

Each participant called *part* executes totally or partially the algorithm below according to the role played by its group.

$S \diamond S$-**Consensus**(g_0, $g_1.g_2.....g_p.i$,InputValue, IdC)

Phase 0 (local):

$V_{part} \leftarrow \prec \perp, \perp, ..., \perp \succ_{g0}$ /*V_{part}: *Vector of values maintained by a participant part of the group* g_0*/

/* Executed by each member *part* of a group g_0*/

0.1. $state_{part} \leftarrow undecided$

0.2. $ts_{part} \leftarrow 0$

0.3. $r_{part} \leftarrow ts_{(g,p)}$

0.4. **If** p=0 **Then** $\triangle_{part}[i] \leftarrow InputValue$

0.5. **Else** $\triangle_{part} \leftarrow$ **S-Consensus**(g_0,

0.5b $g_1.g_2.....g_p.i$,InputValue,(IdC,0))

0.6. **EndIf**

While $state_{part} = undecided$

$r_{part} \leftarrow r_{part} + 1$

$c_{g0} \leftarrow (r_{part} \ mod \ G) + 1$

{c_{g0} is the index of the current coordinating domain}

Phase 1 : /* Each participant i sends the $estimate_{part}$ to the current coordinating group*/

1.1. **Send**$(i, (IdC, r_{part}), estimate_{(g,p)}, ts_{part})$ **To** c_{g0}

Phase 2 /*Each member of the coordinating domain gathers $\lceil (G+1) \rceil /2)$ estimates and proposes a new domain's estimate*/

2.1. **If** $part \in c_{g0}$ **Then**

2.2. **Wait until** [**For** $\lceil (G+1) \rceil /2)$ domains g',

2.3. **Received** $(g', (IdC, r_{part})$,

2.4. $estimate_{g'}, ts_{g'}$ **From** $g']$

2.5. $\triangle_{part}[(IdC, r_{part})] \leftarrow$

2.6. $\{(g', (IdC, r_{part}), estimate'_g, ts_{g'})$

2.7. $|part$ **Received** $(g', (IdC, r_{part})$,

2.8. $estimate_{(g',q)}, ts_(g', q))$ **From** (g', q) }

2.9. **S-Consensus**(g_1, $g_1.g_2.....g_p.i$,$\triangle_{part}$, $(IdC, r_{part}))$

2.10 $t \leftarrow$ largest $ts_{g'}$ **Such That**

2.11 $(g', r_{part}, estimate_{g'}, ts_{g'}$

2.12 $\in \triangle_{part}[r_{part}]$

2.13 $estimate_{part} \leftarrow$ **Select one** $estimate_{r_{g'}}$

2.13b /* Deterministic selection */

2.14 **Such That**$(q, (IdC, r_{part}), estimate_{g'},$

2.15 $ts_{g'}) \in \triangle_{part}[(IdC, r_{part})]$

2.16 **Send** $(i, (IdC, r_{part}), estimate_{part})$ **To ADP**

Phase 3 /*All simple participants wait for the new group's estimate sent by a member of the current coordinating group*/

3.Ini $estimate_c \leftarrow null$

3.0. **Wait until** [$\exists part \in c_g$

3.1. **Received**$(c_{part}, (IdC, r_{part}), estimate_{c_{part}})$

3.2. **From** c_{part} **Or** $c_g \in CFD_{part}]$

3.3. **S-Consensus**(g_1, $g_1.g_2.....g_p.i$,$estimate_{c_{part}}$, $(IdC, r_{part}))$

3.4. **If** $estimate_c \neq null$ **Then**

3.5. /* $part$ received $estimate_{c_{part}}$*/

3.6. $estimate_{part} \leftarrow estimate_{c_{part}}$

3.7. $ts_{part} \leftarrow r_{part}$

3.8. $r_{g0} \leftarrow ts_{part}$

3.9. **Send** $(i, (IdC, r_{part}), ack)$ **To** c_{part}

3.10 **Else**

3.11 **Send**$(p, (IdC, r_{part}), nack)$ **To** c_{g0}

3.12 /* p suspects that c_{g0} crashed */

Phase 4 {Each member of the current coordinating group waits for $\lceil (G+1) \rceil /2$ replies. If $\lceil (G+1) \rceil /2$ group adopted their group's estimate, all correct members of the coordinating group R-broadcast a decide message after a consensus }

4.0. **If** $part \in c_{g0}$

4.1. **Wait until** [**For** $\lceil (G+1) \rceil /2)$ domains g',

4.1b $\exists g'$,: **Received**$(g', (IdC, r_{part}), ack)$

4.1c **Or** $(g', (idenditifier, r_{part}), nack)]$

4.2. $V_{part} \leftarrow null$

4.3. **If** [**For** (G+1)/2 group g', $\exists (g', q)$:

4.3b **Received** $(g', (IdC, r_{part}), ack)]$

4.4. $V_{part} \leftarrow ack$

4.5. **Else**

4.6. $V_{g,p} \leftarrow nack$

4.7. **S-Consensus**(g_1, $g_1.g_2.....g_p.i$,$V_{g,p}$,

4.7b $(IdC, r_{part}))$

4.8. **if** $V_{part} = ack$

4.9. **R_GroupBroadcast** ($part$,

4.9b (IdC, r_{part}),

4.10. $estimate_{part}, decide)$

/* If *part* R-delivers a decide message, *part* decides accordingly */

H.0. **When** R-delivers $(g', r_{g'}$,

H.0b $estimate_{g'}, decide)$

H.1. **If** $state_{part} = undecided$

H.2. **Decide**$(estimate_{g'})$

H.3. $state_{part} \leftarrow decided$

Fig. 2: Solving consensus problem hierarchically

4.1.2 Proof: (Sketches)

Let us recall that we assume that there is at least one correct simple participant which is never suspected as long as its group is not crashed and that there is a majority of groups that remain corrects during any run.

Due to space constraints, we will give only some points on which are based the proof of consensus properties satisfied by our protocol.

A "coordinating" participant is a member of the current coordinating sub-group.

-*Result 0* :All the *estimates* received by a coordinating member are values proposed by sub-groups.

A value proposed by a group belonging directly to g (i.e. at the higher level of the hierarchy excluding the root) is a value proposed by a participant and successively approved by the successive groups on which it belongs. So every outcome value is a group value resulting to the recursive launch of the *S-Consensus* procedure (*line 0.5*).

-*Result 1* : **Locally and globally participants decide the same value**

Thanks to the successive launches of the sub-group consensus, all sub-group's members will have the same decided value at any step of the protocol. Furthermore, each participant is informed for each decision taken by the entities of the higher level through its sub-group address. By considering each sub-group as a macro-process the reader can apply the same demonstration as mentioned in [4].

-*Result 2* : **No participant is blocked on a wait statement**

Let us suppose that a process blocks on a wait statement and let us take the earliest round and the earliest point on that round where this blocking occurs. If it is local consensus then all correct simple participants are participating. By the consensus (local detectors) properties there can't be a blocking [4]. If it is an external waiting statement by a simple participant, the wide strong completeness property ensures that the waiting will stop. If it is an external waiting statement by a coordinating participant, the hypothesis of a majority of correct groups guarantees the reception of at least a majority of messages (by the hypotheses) and consequently no blocking could occur.

The main properties are satisfied. The *Validity* property is verified (it is a direct consequence of *Result 0*). No two participants (resp. sub-groups) decide differently. It is a direct consequence of *Result 1* and that ensures the *Uniform Agreement*. The *Termination* condition relies on the completeness property and on the *Result 2*. Suppose that the algorithm has reached the time when one correct sub-group at the higher level is no longer suspected by its peers. Hence, as no participant is blocked (by *Result 2*), there will be a round when this non-suspected sub-group will become the coordinating sub-group and then the global decision will surely be taken at this time.

4.1.3 S-Consensus

Due to space constraints we will give only a brief description of this protocol.

The protocols used the class S in external and internal levels of the hierarchy customized to WAN. The protocol works (terminates) even with one participant (and subsequently with one higher level sub-group) that remains correct.

The protocol works in four phases similarly like the $S \diamond S$ protocol with the difference that there is not a coordinating sub-group. The reader can refer to [11, 12] and can easily adapts the protocols presented in those papers for more than two levels by replacing *Local-Consensus* procedure by the $S - Consensus$ with the additional parameters such as in $S \diamond S$ procedure.

4.1.4 Some Comments

If one compares the algorithms we have proposed with the original algorithm, one can make two remarks. Firstly, the new assumptions on the detectors are stronger since having to be verified within each sub-group. Secondly, the number of exchanged messages decreases significantly (see [11, 10] for an evaluation in the case of a tree of two levels).

One could evoke that the recursivity increases the protocol complexity according to the depth of the hierarchy (tree). This is not incorrect. Although, the recursivity is necessary at any step of the protocol for systems requiring a "hard" consistency. For "soft" consistency, the recursivity could be used only in *Phase0* when choosing the initial sub-group's value.

5 Conclusion and Perspective

We have presented a recursive and generic mechanism to solve the consensus problem between entities organized hierarchically as a tree. The leaves are the participants themselves and the other nodes are the sub-groups.

The genericity is obtained by modifying the algorithms of literature according to some simple rules at the initial phase, and according to sending and the reception of message.

The new approach used a variable failure detector where, at the higher level of the hierarchy, it is possible to choose either the detector class S or $\diamond S$ according to characteristics of the interconnected network.

The consensus protocol we have proposed reaches the goals pursued by our preceding algorithms [11, 12, 10] and that it generalizes. Over and above the reduction of the number of messages, the conservation of making-decision strategies specific to each group for intermediate decisions, a greater flexibility of organization is provided through the concept of hierarchical group and that of variable behavior of the detector.

The prototyping of this mechanism will mainly require to solve two implementation problems : the man-

agement of an hierarchical group and the implementation of the detectors.

The CORBA[3] Large Group Service (called *CLAGS*) we are implementing is an architecture offering infrastructure for large-scale agreement (relying on our generic protocol). It is a group service overcoming the lack of group communication in CORBA. It is inspired from OGS [9, 6] (designed to work only in a LAN). With *thread* mechanisms it allows the management of asynchronous messages. This makes it possible to mitigate the limitations of the synchronous CORBA object invocation service.

The large-scale monitoring furnishes support for detecting failure of local or remote entities. It will be based in an adaptable mechanism defining time-outs. These time-outs will depend on the parameters of the underlying networks. It may vary over the time according to the location of the entities involved, the network characteristics and the time of communication. This flexibility will permit to find an "optimal" time-out to ensure in one hand, the reactivity of the system according to failure and in the other hand to avoid incorrect suspicions when the time-out (after which a silent entity could be suspected to be faulty) is not well-tuned.

References

[1] O. Babaoglu and A. Schiper. On group communication in large-scale distributed systems. *OS review, JACM*, 29(1):62–67, Jan. 1995.

[2] Kenneth P. Birman. *The process group approach to reliable distributed computing*, volume Reliable Distributed Computing with Isis Tooklit, pages 27–57. IEEE CSSP., 10662 Los Vaqueros Circle PO Box 3014 Los Alamitos, CA 90720-1264, 1994.

[3] D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost. On the impossibility of group membership. Technical report, 95-1548, CS. Dept, Cornell Univ., Ithaca, NY 14853, Oct. 1995.

[4] T. D. Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *JACM*, 43(2):225–267, 1996.

[5] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *JACM*, 35(2):288–323, April 1988.

[3] Common Object Request Broker Architecture [16]

[6] P. Felber. *The CORBA Object Group Service*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne-Suisse, 1998.

[7] C. Fetzer and F. Cristian. On the possibility of consensus in asynchronous systems. In *Proc. of the Int. Symp. on F-T systems*, Dec. 1995.

[8] Michael J. Fisher, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *JACM.*, 32(2):374–382, April 1985.

[9] R. Guerraoui and A. Schiper. Consensus service: a modular approach for building agreement protocols. In *Proc. of the 26th IEEE FTCS, Sendai*, pages 168–177, June 1996.

[10] S. Haddad and F. Nguilla Kooh. A consensus protocol using different failure detectors in large-scale networks. In *Proc. of 14th ISCA International Conference on Computer and Their Applications, Cancun-Mexico*, April. 1999.

[11] S. Haddad, F. K. Nguilla, and A. El-Fallah S. A consensus protocol for wide area networks. In *Proc.of DAPSYS'98, Budapest-Hungary*, Sept. 1998.

[12] F. Nguilla Kooh. Hierarchical approach for solving agreement problems in wide distributed systems. In *Proc. of IASTED Int. Conf. PDCN'98, Brisbane-Australia*, 1998.

[13] L. Lamport and M.J Fisher. Byzantine generals and transaction commit protocols. Technical report, OP. 62. SRI Int., Menlo Park, Calif, 1982.

[14] L. Lamport and M. Pease R. Shostak. The byzantine generals problem. *ACM Trans. Program. lang. Syst.*, 4(3):52–78, July 1982.

[15] Silvano Maffeis. The object group design pattern. In *Proc. of the USENIX Conf. on OO Technologies, Toronto*, June 1996.

[16] OMG. The common object request broker : architecture and specification. Available on http://wwww.omg.org.

[17] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JACM.*, 27(2):228–234, April 1980.