

## Chapter 9

# Fault Management in a Multicast Routing Environment

### *Kernel Based Tree Protocol, a Case Study\**

Mohamed Dâfir ECH-CHERIF EL KETTANI <sup>(1)</sup>  
Younes SOUISSI <sup>(2)</sup>

<sup>(1)</sup> *Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes  
ENSIAS – BP 713 – Rabat Agdal – Morocco  
e-mail: dafir@ensias.um5souissi.ac.ma*

<sup>(2)</sup> *Ecole Mohammedia d'Ingénieurs  
Department of Computer Science – EMI – BP 765 – Agdal, Rabat – Morocco  
e-mail: souissi@emi.ac.ma*

**Key words:** Inter Domain Multicast Routing, Consensus, Fault Management, Adaptive Communications, Scalability

**Abstract:** The conception of multicast routing protocols relies generally on the optimisation of Quality of Service parameters. Unfortunately, we rarely find propositions that consider fault tolerance as a criterion to enhance QoS associated to the group: We mean by “fault tolerance” the ability to manage failures of nodes and links of the distribution tree, in the network. A fault tolerant approach at the routing level will allow more reliability at the *network* and *service* levels.

Currently, in a multicast routing environment, when a failure occurs, the solution consists in destroying the distribution tree, and starting from the beginning to construct again the whole distribution tree, which is a costly solution.

We propose a solution avoiding that, by giving fault tolerant extensions to multicast routing protocols. Our proposition is based on the resolution of the *consensus problem*. Consensus problems were studied by Chandra and Toueg: We propose an *inter domain* version of the algorithm. We put the emphasis on scalability issues, due to the hierarchical structure of the network, such as the Internet. We study the case of KBT protocol, a QoS-sensitive protocol, adapted to *inter domain* multicast routing.

\* This work is partially supported by PARS MI 16 project

## 1. INTRODUCTION

In this paper, we are interested in fault management in a multicast routing environment. The solution that we propose gives fault tolerant extensions to multicast routing protocols in general, and particularly to inter domain multicast routing protocols. We mean by “fault tolerance” the ability to manage failures associated to nodes and links that belong to the distribution tree, in the interconnection network. A fault tolerant approach at the routing level will allow more reliability associated to such *network* protocols. It will also associate to the multicast routing protocol a reliable multimedia *service*.

Currently, in a multicast routing environment, when a failure occurs, the solution generally consists in destroying the distribution tree, and starting from the beginning to construct again the whole distribution tree, since every member will join the group again individually, which is a costly issue. This is the case of CBT protocol. Also, resource discovery relies on a mechanism that does not scale [1].

We look for a solution that avoids these problems, and takes advantage of the hierarchical structure of the interconnection network: it supposes that the interconnection network is organised into domains. When the failure of a node occurs, the distribution tree breaks out, but no pruning of the distribution tree happens. We propose a solution that “elects” a new node to replace the failing one, and makes the necessary work to connect all the components of the broken distribution tree to that new node.

Indeed, we first partition the initial distribution tree into sub-trees (one sub-tree per domain). Then each sub-tree will have to decide the value associated to the new “elected node”: But, that decided value must be the same among all the domains, which justifies the use of a *consensus* protocol.

Our proposition is based on the resolution of *consensus problem* in an inter domain multicast routing environment. Consensus problems were studied by Chandra and Toueg [3]. We propose an inter domain version of this protocol, solving thus scalability issues, especially in terms of transmitted messages. We study the case of KBT protocol, a QoS-sensitive protocol, adapted to *inter domain* multicast routing. We also analyse the complexity of the solution, and show the advantages of such a policy.

The rest of the paper is structured as follows: In section 2, we present the paradigms associated to KBT protocol, in order to illustrate further fault management in that case. In section 3, we describe the system model, including the failure detector model in an inter domain routing environment. In section 4, we present the consensus algorithm associated to KBT protocol. In section 5, we evaluate the performance of the protocol: We analyse the complexity of the algorithm and compare results with other algorithms. Section 6 corresponds to the conclusion and perspectives of the work.

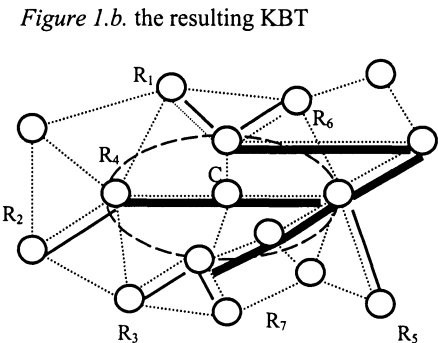
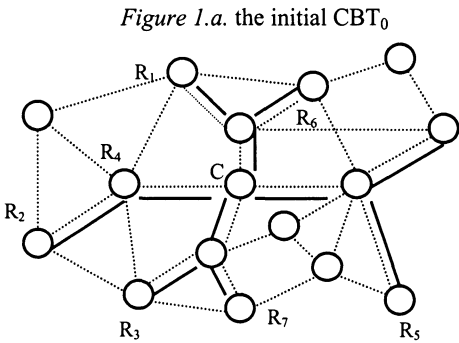
2. KBT PROTOCOL: STATE OF THE ART

2.1 Overview

Kernel Based Tree protocol is an adaptive inter domain multicast routing protocol. Such a solution has already been presented as part of a hole project, the goal of which was to define an “Inter Domain QoS Multicast Routing Protocol”. The first experimentation led to encouraging results [6]. The details of KBT protocol have already been presented in [4]. We present in this paragraph the main characteristics of KBT, in order to treat fault tolerant extensions in next sections.

KBT protocol combines both of the strategies associated to the construction of distribution trees: “Source Based Tree” strategy (noted SBT), and “Shared Tree” strategy (noted ST). These strategies are adapted to two extreme situations of group members configurations referring to the density of group members in the Internet, called “dense” and “sparse” [7]. Combining SBT and ST leads to adaptation properties of the group.

Our strategy consists in grouping initially group members through a core based tree (noted  $CBT_0$ , figure 1.a) –according to CBT protocol [1]-, and then defining “interception points” on that tree. The position of “interception points” may change, according to the evolution of group membership and network characteristics [4]. These points represent the “kernel” of the future distribution tree associated to the group. A message sent from a source will transit through 2 trees: the first is a source based tree (SBT) linking that source to “interception points”, and the second is the branch of  $CBT_0$ , starting at the “interception point”. We call the obtained distribution tree “**Kernel Based Tree**” (KBT). Since KBT is an inter domain multicast routing protocol, we suppose that we place one “interception point” per domain involved in the construction of KBT tree. Figure 1.b corresponds to the situation where the (KBT) has 4 “interception points”: a source based tree (SBT) links  $S_1$  source to the “interception points”.



## 2.2 Management Organisation of KBT Protocol

From a network management viewpoint, and in association to the distribution tree, we can say that KBT architecture is composed of (figure 2):

- The group manager (or group co-ordinator)
- The group initiator (the first node asking for group creation)
- The “core” of  $CBT_0$
- The “Interception Point” (noted SIP for Source Interception Point and RIP for Receiver Interception Point)
- The “Receiver Interception Point” Candidate
- The group member (Receiver or Source)

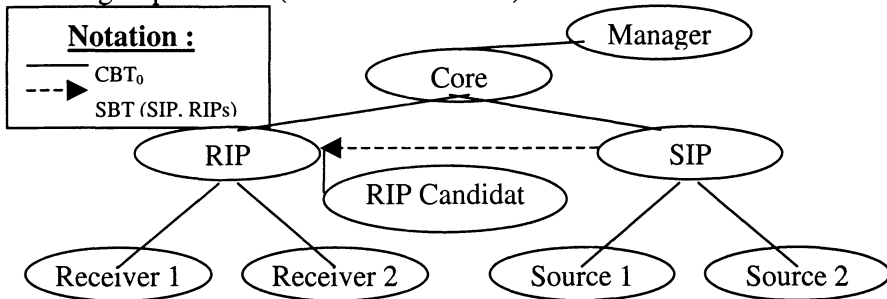


Figure 2. Nodes' dependency

Figure 2 illustrates the dependency graph between KBT nodes. The group manager and the “core” are the main components of the group. The failure of one of them will provoke the failure of the whole group.

## 3. MODELING FAILURES

### 3.1 Classification of Failures

We can divide failures into two categories: *local* and *global*.

A *local* failure will not stop the multicast service toward the totality of group members. Its impact on the coherence of the group will be limited. The failure of an “Interception Point” in KBT [5] is an illustration.

A *global* failure is a failure that will stop the multicast service to the whole group members. Its influence on system coherence will be serious, otherwise disastrous. For example, the failure of the “core” of the multicast tree in CBT or KBT is considered as *global*. Also, the failure of the “group manager” in KBT will be *global*.

The purpose of this paper is to deal with *global* failures only.

### 3.2 Working Rules

The global system works according to 2 main principles:

1. Communications between nodes:

The nodes of the multicast tree associated to a group will be partitioned into domains. Communications inside a domain will be supported by the local multicast tree associated to the group in that domain, whereas we'll have point to point communications at the inter domain level: since a failure is *global*, intra domain communications are not affected by this failure.

2. Detection of a failure:

We associate a failure detector model to our inter domain working environment. Indeed, we suppose that each node of the multicast tree has a failure detector, which is a module that must be able to test the failure of any other node, no matter where the other node is localised (whether inside the same domain, or not). Failure detector model is presented in the next section.

### 3.3 Failure Detector Model

A failure detector is defined as a module associated to each node of the distribution tree. It outputs the set of nodes of the distribution tree that it currently suspects to have crashed [3]. From a consensus problem viewpoint, two different nodes  $n_1$  and  $n_2$  may not necessarily suspect the same set of failing nodes, because the network is an asynchronous system [10]. So, a failure detector is defined by 2 sets of properties [3]:

#### Completeness

- **Strong Completeness:** eventually every node that crashes is permanently suspected by *every* correct node.
- **Weak Completeness:** eventually every node that crashes is permanently suspected by *some* correct node.

#### Accuracy

- **Strong Accuracy:** no node is suspected before it crashes.
- **Weak Accuracy:** some correct node is never suspected.
- **Eventual Strong Accuracy:** there is a time after which correct nodes are not suspected by any correct process.
- **Eventual Weak Accuracy:** there is a time after which some correct process is never suspected by any correct process.

Chandra and Toueg deduce 8 classes of failure detectors:

<div style="text-align: center;"> <div style="transform: rotate(-45deg); display: inline-block;">Accuracy</div> </div>	Strong	Weak	Eventual Strong	Eventual Weak
Strong	P	S	$\Diamond P$	$\Diamond S$
Weak	Q	W	$\Diamond Q$	$\Diamond W$

They also show that in an interconnection network environment (like the Internet), the behaviour of a failure detector associated to a node is characterised by “Strong Completeness” and “Eventual Weak Accuracy” properties: such a failure detector belongs to  $\diamond S$  class.

However, due to the structure of the interconnection network (organised into domains), we can extend the failure detector model proposed by Chandra and Toueg, based on the class  $\diamond S$  [3].

Hence, we define a new class of failure detectors, noted by  $\diamond \check{S}$ , giving hierarchical extension to the failure detector model: This class will act as if each node maintains two failure detectors, the first dedicated to internal (intra domain) communications, and the second for inter domain communications.

This class  $\diamond \check{S}$  will be characterised by two properties, noted by:

- “*hierarchical strong completeness*”: intra domain completeness and inter domain completeness
- “*hierarchical weak accuracy*”: intra domain weak accuracy and inter domain weak accuracy

It is important to make this hierarchical distinction, since the interconnection network may be confronted to important heterogeneity characteristics, such as delay, or bandwidth, between intra and inter domain communications.

## 4. THE CONSENSUS PROTOCOL

### 4.1 Related Work

Chandra and Toueg proposed solutions to the consensus problem, specific to the different basic classes of failure detectors [3].

We can describe informally a consensus protocol as follows, given a set of correct nodes. At the beginning of the protocol, each correct node is given an input value, and at the end, the non-crashed nodes must have decided on a common output value, belonging to the set of input values.

The following properties define the consensus problem [3]:

- **Termination**: every correct node eventually decides some value.
- **Integrity**: a node decides at most once.
- **Agreement**: no two correct nodes decide different values.
- **Validity**: if a node proposes some value, then this value must have been proposed by some node.

Chandra and Toueg show that the consensus problem can be solved using  $\diamond S$  class, with at least a majority of correct nodes: no blocking happens.

## 4.2 Overview and Example

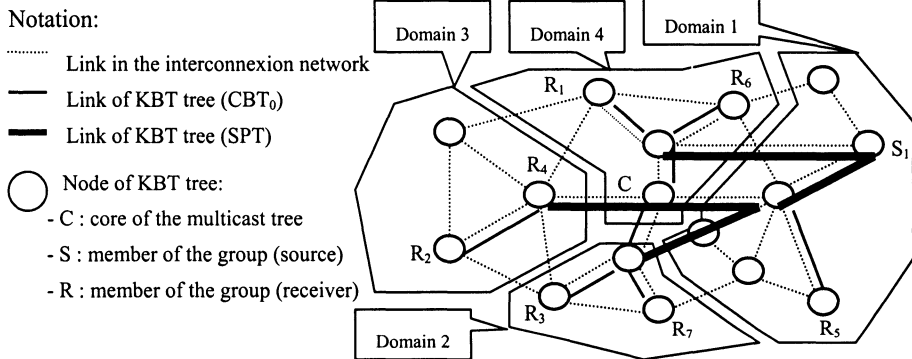
We propose an inter domain version of the consensus protocol, solving thus scalability issues. Failure detectors belong to  $\diamond S$  class.

For example, suppose that we have a group working under KBT protocol (figure 3). We consider that “core” failure is a *global* failure. If the failure of the node C happens (C is the “core” of the group), we’ll first partition the initial multicast tree  $CBT_0$  into sub-trees, according to domains (figure 3).

At the end of this first step, each sub-tree proposes one potential new “core” candidate to the group. The choice of this new “core” candidate is operated thanks to QoS considerations, locally to each domain.

However, the final value of the new “core” must be the same among all the sub-trees: a consensus protocol, at the inter domain level, will let all the sub-trees decide on a common final value.

Figure 3. KBT multicast tree (4 domains, 1 source, 7 receivers)



## 4.3 Principle of the Solution

The consensus problem -based on  $\diamond S$  class- can be solved with the hypothesis that the maximum number of failing nodes is less than half of implied nodes. Chandra and Toueg show that, at any time, all the nodes may be erroneously added to the list of suspects. However, there is a correct node and a time after which that node is not suspected to have crashed [3]. In the case of  $\diamond S$  class, we make the same suppositions, since:

- A node of  $\diamond S$  class has at the same time two failure detectors.
- The properties of each failure detector are taken from  $\diamond S$  class.

So, the proposed consensus protocol can be solved with the hypothesis:

- Each implied *correct* domain proposes a local value. So, it is supposed not to be a *failing* domain. In a *failing domain*, the maximum number of failing nodes is less than half of the nodes of the group in that domain.

- A majority of implied domains in the consensus process are not failing: they are correct.

## 4.4 Description of the protocol

The proposed protocol uses the rotating co-ordinator paradigm [2] [9], at the *intra* and *inter* domain levels. Our solution is summarised in 3 main steps (figure 4). Details are presented in section 4.5.

### Step 1:

We start with a local consensus in each domain  $D_i$  of the interconnection network. That's why each domain  $D_i$  will have a local co-ordinator. The rotation of local co-ordinators will act in an asynchronous way in terms of rounds. Each round is characterised by the four classical steps of consensus protocol [3]. At the end of this first step, each domain  $D_i$  proposes a common value -noted  $\Delta_i$ - to all the nodes of that domain among the proposed values  $\Delta_{i,j}$ . The local co-ordinator of domain  $D_i$  is noted  $P_{\text{local}}(i)$ .

### Step 2:

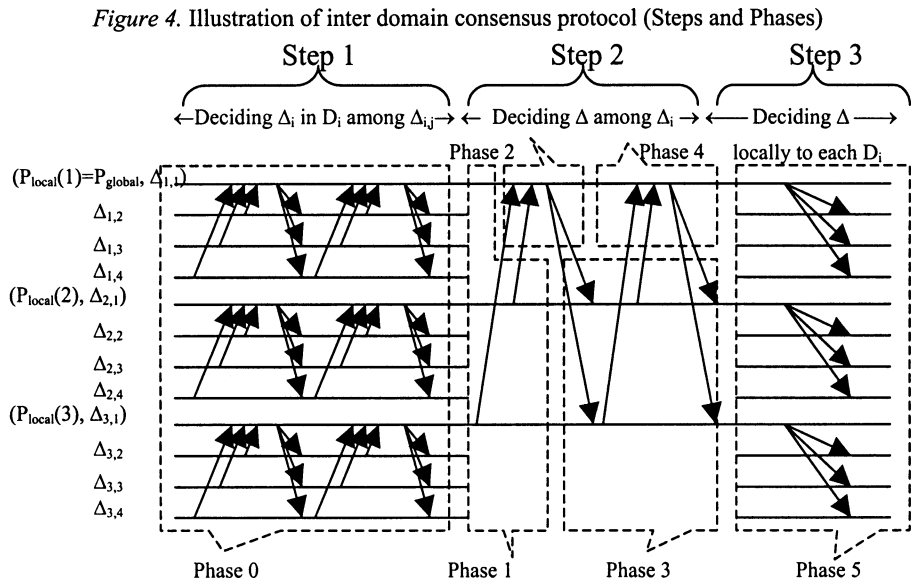
We apply the consensus algorithm at the inter domain level, on the set of local co-ordinators  $P_{\text{local}}(i)$  of domains  $D_i$ . Hence, each domain  $D_i$  proposes an initial value  $\Delta_i$ . This second step needs a global co-ordinator, which will meet one of the local co-ordinators  $P_{\text{local}}(i)$ , defined in step 1.

At the end of this step, each local co-ordinator has received the common final value  $\Delta$ , decided by the global co-ordinator.

### Step 3:

In this last step, each local co-ordinator  $P_{\text{local}}(i)$  in domain  $D_i$  broadcasts the decided value  $\Delta$  to each node of his domain.

The following figure summarises the main steps of the proposed protocol. In this example, we suppose that we have three domains  $D_1$ ,  $D_2$ , and  $D_3$ . We also suppose that  $P_{\text{local}}(1)$  is the global co-ordinator.



## 4.5 Algorithm

Each local co-ordinator node  $P_{local}(i)$  executes the following algorithm. It will either execute that algorithm totally or partially, depending on the role of that node in the consensus protocol, which will be one of the following:

Local co-ordinating node, noted  $P_{local}(i)$ , belonging to a domain  $D_i$

Global inter-domain co-ordinating node, noted  $P_{global}$ .

Notation:

*Local\_consensus()*: consensus algorithm of Chandra and Toueg, executed in the domain mentioned as a parameter. It returns the estimated value  $\Delta_i$ .

*F\_Broadcast\_local()*: function that broadcasts the estimated value  $\Delta = \text{Estimate}_{global, P_{local}(j)}$  from a local co-ordinator to each node of its domain.

*F\_Broadcast\_global()*: function that broadcasts the estimated value  $\Delta$  from the global co-ordinator to local co-ordinator of each domain.

*F\_rcv\_msg\_bcast()*: reliable function launched by each domain's local co-ordinator, waiting for the estimated value  $\Delta$  from the global co-ordinator. It broadcasts reliably (locally to each domain) the decided value.

*Decide()*: function that makes the final decision.

$\text{Estimate}_{global, P_{local}(i)}$ : the estimate of the global expected value  $\Delta$ , in  $P_{local}(i)$  node of domain  $D_i$ .

Figure 4 summarises the main phases of the algorithm. The algorithm of the “inter\_propose” procedure of the inter domain consensus protocol is:

**Procedure** inter\_propose( $P_{local}(i)$ ,  $\Delta_i$ )

**Phase 0 (executed by each local co-ordinator)**

$\text{Estimate}_{global, P_{local}(i)} = \text{Local\_Consensus}(D_i) = \Delta_i$ ;  $\text{State}_i = 0$ ;  $R = 0$ ;  $\text{TS}_i = 0$ ;  
while ( $\text{State}_i = 0$ ) do  
     $\{R = R + 1$ ;  $P_{global} = (R \bmod G) + 1$ ;}

**Phase 1**

Send ( $P_{local}(i)$ ,  $R$ ,  $\text{Estimate}_{global, P_{local}(i)}$ ,  $\text{TS}_i$ ) to  $P_{global}$ ;

**Phase 2**

if ( $P_{local}(i) = P_{global}$ ) then  
    **wait** until [ $\lceil (G+1)/2 \rceil$  nodes  $P_{local}(j)$ , with  $j \neq i$  :  
        receive ( $P_{local}(j)$ ,  $R$ ,  $\text{Estimate}_{global, P_{local}(j)}$ ,  $\text{TS}_j$ );  
         $\text{msgs}_i[R_i] = \{(P_{local}(j), R, \text{Estimate}_{global, P_{local}(j)}, \text{TS}_j) /$   
             $P_{local}(i) \text{ received } (P_{local}(j), R, \text{Estimate}_{global, P_{local}(j)}, \text{TS}_j)\}$ ;  
         $t = \max \{\text{TS}_j\}$ ;  
         $\text{Estimate}_{global, P_{local}(i)} = \text{Estimate}_{global, P_{local}(j)} / \{t = \text{TS}_j \text{ and}$   
             $(P_{local}(j), R, \text{Estimate}_{global, P_{local}(j)}, \text{TS}_j) \in \text{msgs}_i[R_i]\}$ ;  
        send ( $P_{local}(i)$ ,  $R$ ,  $\text{Estimate}_{global, P_{local}(i)}$ ) to all  $P_{local}(j)$ , with  $j \neq i$ ;

**Phase 3**

**wait** until  $\{(received(P_{global}, R, Estimate_{global,C})) \text{ or suspect failure of } (P_{global})\}$ ;  
 if  $(received(P_{global}, R, Estimate_{global,C})=true)$  then  
      $Estimate_{global,P_{local}(i)}=Estimate_{global,P_{global}}; TS_i=r; send(P_{local}(i), R, ack) \text{ to } P_{global};$   
 else  
      $send(P_{local}(i), R, nack) \text{ to } P_{global};$

**Phase 4**

if  $(P_{local}(i)=P_{global})$  then  
     **wait** until  $(for \lceil (G+1)/2 \rceil \text{ nodes } P_{local}(j), \text{ with } j \neq i : \\ received(P_{local}(j), R, ack) \text{ or received}(P_{local}(j), R, nack));$   
     if  $(for \lceil (G+1)/2 \rceil \text{ nodes } P_{local}(j), \text{ with } j \neq i: received(P_{local}(j), R, ack)=true)$   
 then  
      $F\_Broadcast\_global(P_{local}(i), R, Estimate_{global,P_{local}(i)}, decide);$

**Phase 5 (executed by each local co-ordinator)**

when  $(F\_rcv\_msg\_bcast(P_{local}(i), R, Estimate_{global,P_{local}(j)}, decide))$   
 if  $(State_i = 0)$  then  
      $\Delta = Estimate_{global,P_{local}(j)};$   
      $Decide(\Delta); State_i = 1; F\_Broadcast\_local(P_{local}(i), R, \Delta, decide);$

## 4.6 Proof of Correctness

We proof that the proposed algorithm verifies the four properties that characterise the consensus problem (Section 4.1). The adopted approach to proof these properties is similar to the one chosen by Chandra and Toueg. For more information, a detailed description exists in [3]. A failure detector of class  $\Diamond S$ , is characterised by intra and inter domain strong completeness, and intra and inter domain eventual weak accuracy

### 4.6.1 Property 1

**Termination:** every correct node eventually decides some value.

**Proof:** It means that no node is blocked on a **wait** statement.

Let us suppose that a node blocks on a wait statement, and let us take the earliest round and point on that round where this blocking occurs.

If it is local consensus, it can't be blocking, by consensus [3].

If it is a waiting statement in phase 2, 3, or 4 of the algorithm, the “hierarchical strong completeness” property will ensure that the waiting will stop. In the case of a waiting statement of the global co-ordinator, then the hypothesis of a majority of correct domains (see section 4.1) guarantees the reception of at least a majority of messages. So, no blocking occurs.

#### 4.6.2 Property 2

**Integrity:** a node decides at most once.

**Proof:**

The algorithm satisfies the integrity property, because each node decides at most once (*Phase 5*): the decide function is called once.

#### 4.6.3 Property 3

**Agreement:** no two correct nodes decide different values.

**Proof:**

If no process decides, the property is trivially true.

If any process decides, let us take the time at which the first decision has been taken. The global inter domain co-ordinating node has eventually received a majority of acknowledgements. This implies that  $\lceil (G+1)/2 \rceil$  of local co-ordinating nodes detain this estimate. We proof that this estimate holds indefinitely. We apply the demonstration in [3], considering in that case each domain as a process. The proof is by induction on the round number. More details can be found in lemma 19 of [3].

In phase 5, since each correct local co-ordinating node has previously executed the F\_rcv\_msg\_bcast function, it will broadcast this value to all the nodes of the domain. The decided value must have been previously decided by the global co-ordinator.

#### 4.6.4 Property 4

**Validity:** if a node proposes some value, then this value must have been proposed by some node.

**Proof:**

From the algorithm, the decided values are taken from estimates at two levels:

*Intra domain level:* all the estimates  $\Delta_{i,j}$  received by any local co-ordinator in phase 0 are proposed values from the nodes of the same domain. The result is a proposed value  $\Delta_i$  at the intra domain level.

*Inter domain level:* all the estimates that a global co-ordinator receives in phase 2 are proposed values from local co-ordinators. So, the value decided by a global co-ordinator must be the value proposed by some node.

As a consequence, the algorithm proposed in section 4.5 solves Consensus using  $\diamond S$  class.

## 5. PERFORMANCE EVALUATION

### 5.1 Complexity Analysis of the Solution

Suppose that we have  $G$  domains, and ' $n$ ' nodes in each domain. The complexity of the algorithm proposed in section 4.5 in terms of exchanged messages is in  $O(G^2)+O(Gn^2)$ :

Indeed, the local consensus operation (phase 0) consumes  $4(n-1)+(n-1)^2$  messages per domain:  $4(n-1)$  is the number of exchanged messages between the nodes and the local co-ordinator, in order to take the local consensus value.  $(n-1)^2$  is the number of exchanged messages due to the final local broadcast function [3]. This broadcast function is supposed to be reliable. So, the global number of messages is equal to  $G*[4(n-1)+(n-1)^2]$ .

The global inter domain consensus operation (phases 1, 2, 3, and 4) consumes  $4(G-1)+(G-1)^2$ . The reasons are the same that in local consensus problem, considering in that case a domain instead of a node.

The final broadcast function in phase 5 consumes  $(n-1)^2$  messages per domain: the broadcast function is a reliable broadcast function. So, the total number of messages is  $G*(n-1)^2$ .

Hence, the complexity of the whole algorithm is in  $O(G^2)+O(Gn^2)$ .

### 5.2 Comparative Analysis

There is an improvement, concerning scalability, in terms of exchanged messages in comparison with Chandra and Toueg algorithm, which is in  $O(G^2n^2)$ .

We can also find that our solution is better than the one proposed by [8], where the complexity is in  $O(G^2+n^2+Gn^2)$ .

## 6. CONCLUSION

Fault-tolerant management in a *multicast routing environment* is a new issue in *inter domain* multicast routing.

In this paper, we have studied the case of KBT protocol, which is a generic adaptive inter domain multicast routing protocol. After this, we presented the main concepts of the protocol, in terms of construction of the distribution tree, and distributed management capabilities: we showed that KBT protocol can integrate at the routing level a certain level of control and management. This offers a reliable multimedia service associated to the multicast group.

Then, we associated a failure model to the system, by presenting a classification of failures associated to the system, and then proposing a new class of failure detectors, adapted to the hierarchical structure of the interconnection network. We finally showed why the problem can be considered as a classical consensus problem.

We also showed how we can optimise that classical consensus problem, through a hierarchical approach. We proposed the algorithm associated to the hierarchical consensus problem, and proofed its correctness.

Performance evaluation of the solution included complexity analysis, and comparison with the consensus protocol: we obtained good results.

The applications of our solution can be extended to protocols others than KBT: "core" election mechanism in CBT and PIM-SM for example. Other applications can be seen in any hierarchical inter domain distributed environment that needs to maintain a global state, or to elect a common value between components of the system.

## REFERENCE

- [1] A. Ballardie, "Core Based Trees (CBT ver. 2) Multicast Routing Architecture", Internet RFC 2201, September 1997.
- [2] T.D. Chandra; and S. Toueg, "Time and Message Efficient Reliable Broadcasts", In Proceedings of 4<sup>th</sup> International Workshop on Distributed Algorithms (WDAG-4), Springer Verlag, November 1996.
- [3] T.D. Chandra; and S. Toueg, "Unreliable Failure Detectors for reliable distributed systems", Journal of the ACM, vol 43, n°2, pp-225-267, 1996.
- [4] M.D. Ech-Cherif El Kettani, "KBT Protocol Architecture", Technical Report n° 02/98, Ecole Mohammedia d'Ingénieurs, Rabat, Morocco.
- [5] M.D. Ech-Cherif El Kettani; and Y. Souissi, "An Adaptive Architecture for Inter Domain QoS Multicast Routing", in French, in Calculateurs Parallèles: Routage dans les réseaux, vol 11, n°1/99, Hermes, pp-59-86.
- [6] M.D. Ech-Cherif El Kettani; and Y. Souissi, "Performance Evaluation of Multicast Routing Protocols: Comparative Study and Proposition of an Adaptive Solution", CFIP'99 Proceedings, Nancy, France, 26-29 April 1999, Hermes, pp-317-332.
- [7] C. Huitema, "Routing in the Internet", Prentice-Hall, Englewood Cliffs, N.J., 1995.
- [8] F. Nguilla Kooch, "Hierarchical Approach for Solving Agreement Problems in Wide Distributed Systems", In Proc. of Second IASTED International Conference on Parallel and Distributed Computing and Networks, December 14-16 1998 Brisbane, Australia.
- [9] R. Reishuck "A New Solution for the Byzantine General's Problem", RJ 3673, IBM Research Lab., Nov. 1982.
- [10] M. Raynal, "Synchronisation and Global State in Distributed Systems", in French, Eyrolles Editions, 1992.