Fault-tolerance in Mobile Environments: A Partition Detection System

Muhammad Usman Bhatti and Denis Conan
GET / INT, CNRS UMR SAMOVAR
9 rue Charles Fourier, 91011 Évry, France
usman.bhatti,denis.conan@int-evry.fr

Abstract

Wireless networks are more ubiquitous than before and they present a whole new set of problems with their inception. Disconnection is one of such problems where mobile terminals can disconnect. Failure is another problem that is inherited from the distributed systems. The two may create partitions in the distributed systems which are necessary to be detected for fault tolerance in mobile environments. In this work, we present a partition detection service based on the knowledge of network topology. This service is interoperable with group communication system.

1. Introduction

Recent advancements in wireless data networking and portable information appliances have given the concept of mobile computing. Users can access information and services irrespective of their movement and physical location. Wireless communication, data processing and information services are becoming more and more important. With the increasing use of mobile terminals and mobile applications, mobility is an extra-functional feature which has become very significant.

Mobile terminals are exposed to vast environments. The mobility of a mobile terminal gives rise to frequent disconnections. These disconnections can be of two types: voluntary disconnections and involuntary disconnections; the former ones are decided by the users and the latter ones, a result of absence of wireless network signals. Disconnections can be very frequent and mobile terminals should continue working even while they are disconnected from the network. Hence, a mechanism is needed for measuring signal strength in order to anticipate for the forthcoming disruption in the network connectivity. We can call it connectivity detector. In addition, disconnection detector is necessary for the voluntary disconnection and involuntary disconnections in order to send an "alert" message to the other nodes declaring its disconnection. [8] presents the idea of disconnection and failure management in mobile applications.

Fault-tolerance is essential for distributed applications. Fault-tolerance comes in two phases: fault detection and fault correction [6]. Fault detection helps in maintaining application's safety and fault correction aides in maintaining application's liveness. In pure asynchronous distributed systems, consensus is insolvable in the presence of even one faulty process [5]. This problem arises from the fact that we cannot differentiate amongst the faulty processes and the processes which are too slow. Nevertheless, unreliable failure detectors have been proposed, which help us solve the problem of consensus [3].

Group communication systems (GCSs) are widely recognized as powerful building blocks for supporting consistency and fault-tolerance in distributed applications. The basic idea supported by GCSs is the notion of *multicast group*. Multicast groups are created on the fly by a *group membership service*. Traditionally, GCSs are employed in replicated objects and database consistency applications. Distributed systems are prone to process crashes as well as link failures. Failures may cause a component or several sets of components to detach from the system, thus making a separate group disparting from the main network, making partition of the system. Partitions may result in service reduction or degradation but need not necessarily render the application completely unavailable. Partitions are a fact of life in most distributed systems and they tend to become more frequent as the geographic extent of the system grows or its connectivity weakens due to the presence of mobile units and wireless links. Thus, the partitions should perform as autonomous distributed systems providing services to their clients. The notion of partitionable GCS is an example where all the partitions are allowed to proceed in their computations [7].

In this work, we are trying to workout the problem of

partition detection in wireless group communication system. Section 2 describes the existing work which we are going to reuse for our purposes. We describe the notion of failure and disconnection detectors. Section 3, we compare the aspects of partitions and failures. Section 4 repeats the same exercise for partitions and disconnections. In Section 5, we develop algorithms to differentiate among disconnection, failure and partition. In Section 7, we present a group membership service. Finally, Section 8 concludes the article and gives perspectives.

2. Related Work

We consider asynchronous distributed systems. They are distributed systems without bounds on message delay, clock drift or time necessary to execute a step. The system consists of processes and the processes communicate by message passing. As we have no bound on message delay, we cannot distinguish if a message is only taking too long to reach its destination or it is a failure [5]. To circumvent this impossibility result, [3] proposed failure detectors, which monitor a subset of processes for failures and can make mistakes, thus the name *unreliable failure detectors*. Each process has an access to a failure detector module. Each module monitors a subset of processes in the system, and maintains a list of those that it currently suspects to have crashed. Failure detectors can be erroneous in their suspicions: they can suspect that process p has crashed while it is still running. Later on, they will remove p from the list of suspects if suspicion was erroneous. Failure detectors have two properties in terms of their functionality. Completeness states that there is a correct process which suspects every faulty process. Completeness is an important property as it satisfies the safety requirements for a failure detector. Completeness can be divided into two properties: *weak completeness* and *strong completeness*. Completeness in itself is not a useful property and has to be augmented with an accuracy property which restricts the mistakes that failure detector can make. Thus, accuracy states that no process is suspected before it crashes. Accuracy satisfies the liveness requirements of a system. Accuracy is divided into four properties, which are *strong accuracy*, *weak accuracy*, *eventual strong accuracy*, and *eventual weak accuracy*.

Apart from failures, mobile computing presents another challenge for wireless distributed system developers, which arises from the mobility of terminals. A mobile process, part of a distributed system, may not be slow or faulty but it may not find itself connected to the network because it has moved out of the communication range. Connectivity detectors are based on the idea of connectivity managers, first presented in [4]. The idea is to monitor the network resources to foresee the network disconnection. When a network disconnection occurs on the link, which is used by the application, the connectivity manager detects the disconnection event and notifies either the application or some other service. In order to insulate the application from the insignificant variations in resource level, the connectivity manager relies on an hysteresis mechanism for smoothing variations in resource availability. The connectivity information is local to each node. For this reason, disconnection detectors were introduced which could transmit this local connectivity information to all the connected processes [8]. Disconnection detector, like failure detector, is described in abstract terms to generalize the model and to avoid any implementation-specific details. For this purpose, abstract properties like *disconnection completeness* and *disconnection accuracy* have been defined.

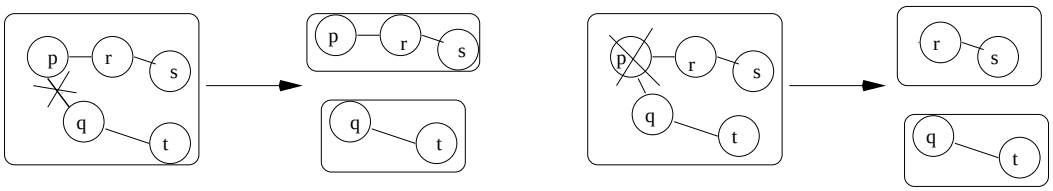
A failure occurs when a process crashes due to an internal failure and does not make any progress. A partition on the other hand, is a problem of the network or a connecting node, while the processes do not crash. In the following sections, we explore these issues.

3. Partition and Failure

The general model of failure detection works as follows: the sender sends a “ping” message and the receiver replies with an “ACK” message. Thus, both sender and receiver know that both of them are “alive”. When there is an absence of a reply for a certain amount of time, called “timeout”, the sender declares the receiver as faulty, or more generally "problematic". Figure 1-a shows that the message m is lost due to a link failure even if both processes, p and q, are alive. Figure 1 depicts this situation where a process q sends a message m to another process p. Figure 1-b shows that the receiver crashes and does not reply to the message m. In both situations, the process on the other side of the network is declared faulty, while that process is not responding or is not receiving the message due to link problems. This deficiency comes from the inherent mechanism of failure detection using the “ping” message. Thus, we have an impossibility result here that we cannot distinguish between link failure and process failure. This result comes from the fact that we cannot distinguish between link failure and process failure by the existing failure detection techniques, defined in Section 2.

A partition occurs when two processes detach themselves such that every process within the partition considers the other processes within the partition to be alive. In the literature, processes outside a partition are considered to be faulty. Therefore, more information needs to be collected for the distinction of failure and partition.

Two processes are called *reachable* if they can communicate with each other directly, or through some other process, which routes the messages to correct processes. A number of events can cause the reachability of the processes to



(a) Link Failures leading to Partitions

(b) Process Failures leading to Partitions

Figure 1. Failure and Partition

be changed into *unreachability*, namely link crashes, buffer overflows, incorrect and inconsistent routing tables. A process crash, may as well, render the two processes as unreachable.

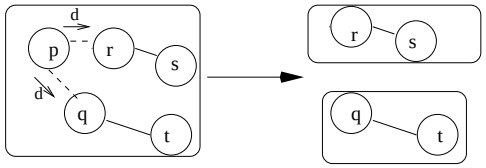
Heartbeat failure detector for partitions is based on the algorithm for partitionable networks defined by [1]. We modify the algorithm so that every process generates the set of processes reachable through each neighbor of its neighbors. These list are then used by the partition detector in Section 5.

4. Partition and Disconnection

While failure detectors make it impossible to distinguish failures from partitions, disconnection detectors does not have this shortcoming because when a process disconnects, it sends a disconnection message to all the processes it is connected to. In this section, we mainly discuss the peculiarities associated with disconnection detector.

As defined earlier, processes send an "alert" message before they disconnect, thus announcing any disconnection leading to a partition. But, still we need additional information to know if the disconnecting process is creating a situation depicted in Figure 2. In the figure, process p disconnects by sending message d to detach two sets of processes. One thing to note here is that with the disconnection message, process q only knows about the disconnection of process p. Had it known it was connected to process r through p, then it would have declared that process p disconnects to form two partitions of the network. Hence, we need some network topology information for detecting any disconnection leading to partitions. One more thing to consider is that processes might be disconnected and reconnect afterwards restoring the original topology as in Figure 2.

We reuse the very same notion of reachability as introduced earlier, that is, two processes are reachable if they can communicate with each other directly, or using a third process as a router. Disconnections may change reachability as well. Looking at Figure 2, we can infer that reachability of processes p and q changes completely thus changing the reachability of other processes, making q unreachable to processes r and s.



Disconnection leading to Partitions

Figure 2. Disconnection and Partition

5. Partition Detection

The architecture for partition detection and membership management is presented in Figure 3. Partition detection is performed using the information collected from the disconnection detector and the failure detector. This information is used by the membership service for view formation. In the two preceding sections, we have seen that we cannot distinguish amongst disconnection, failure and partition using the existing works. Therefore, additional information has to be collected. We develop two algorithms for partition detection: The neighborhood topology and the global topology. In this section, we sketch these two algorithms that we develop in the framework of this work. In Section 5.1, we present the partition detector based on neighborhood information. In section 5.2, we present the partition detector based on global topology information.

5.1. Partition Detection with Neighborhood Topology

Neighborhood topology tries to discover partition formation using the network topology of the neighbors. The idea of neighborhood connectivity is to find the neighbors of the neighbors, as defined earlier. In Figure 4, process p tries to discover the neighbors of process w and q. We construct our neighborhood topology in two layers, as shown in Figure 3. The lower layer, consisting of the failure and the

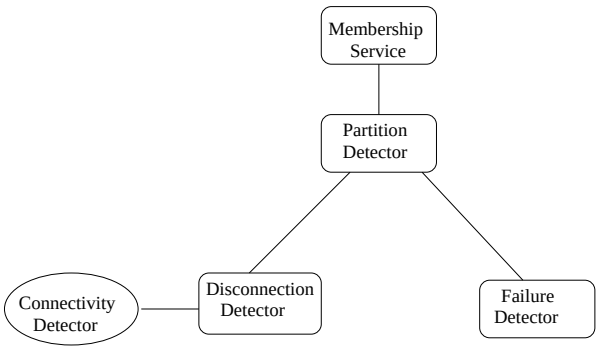


Figure 3. Disconnection, Failure and Partition Detector

disconnection detectors, collects the reachability information and passes it to the partition detector, which uses this information for finding partitions in the network. Partition detection starts as soon as a disconnection or a failure is detected in the system. Every process that disconnects or fails eventually provokes a partition detection at a neighbor and every process reachable through only that neighbor is declared partitioned. In this way, neighborhood topology leads to partition detection by neighbors which disseminate this information to all the processes in the network.

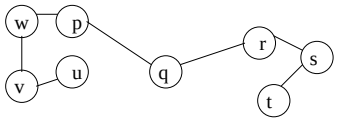


Figure 4. Reachability Pattern

5.2. Partition Detection with Global Topology

The second partition detector algorithm also uses the two previously defined algorithms, failure and disconnection detection for partition detection. The algorithm constructs a knowledge of the global topology instead of the neighborhood topology. We assume that the topology (processes plus links) is known at the starting time of the distributed application —*i.e.*, there exist configuration data describing the initial configuration of the distributed application and these data are known to every process of the distributed application. Then, during the execution, new processes and new links are explicitly added and committed by all the processes of a partition. Each process updates the graph by tagging nodes due to disconnections, failure, or partitions. One major change is that partition message is not needed to be sent to any neighbors. This is because the knowledge of partitions is global in this algorithm and all the reachable processes can detect the processes that have partitioned.

6. Topology and Application-based Decisions

Our work differentiates between physical topology and logical topology. The difference between the physical topology and the logical topology comes from the fact that in our work, we only consider two processes reachable if they have the ability of communicate with each other. For us, this is a higher level of abstraction than actually looking at physical links that link two or more processes. We combine the reachability information with the physical topology by discovering the links connecting two neighbors in the partition detector, meaning that we gather the reachability information with the help of the individual neighbors. Consequently, every process constructs its local view by discovering the processes reachable through a neighbor and collecting them in one set. Thus, a neighborhood logical topology is built, instead of actual link in order to estimate the connectivity patterns. The second algorithm keeps a global information of each link and process. In case of link and process failures, the global topology is affected. This change is seen by all the processes within the current partition and they tag the links and processes.

Failure detectors and neighborhood information can only provide hints for applications. Thus, the sets of processes in the two partitions can only "speculate" what has gone wrong on the other side, since they cannot communicate directly. It is the application which will decide, considering its functionality, whether to drop the users or open new connections to them. Thus, there can be an optimistic approach and a pessimistic approach.

7. Group Membership Algorithm

We propose the following algorithm for the membership service. The partition detectors provide the set of reachable processes and the membership service tries to build a view consisting of all the mutually reachable processes in $reachable_p$ ($reachable_p$ corresponds to the variable in the partition detector). Every process p that detects a change in its set $reachable_p$ sends a message $(\mathsf{JOIN\_VIEW}, new\_vid, reachable_p)$ to all the processes in the new set $reachable_p$. We assume that there is an entity in the systems that generates the view identifiers (vid). Thus, it generates view identifiers (vid) that are monotonically increasing with time. Every process that receives the view change message may be in two states: it may be in a view having a vid lower than that of the sending process or the process is already in the process of changing a view. For the first case, the process q accepts the mes-

sage, changes its set $reachable_q$ according to the information sent by $reachable_p$ and sends an "ACK" message. The process that initiates the view change process, adds all the processes which sent an "ACK" to the new view. A new view message is sent to all the process in the new view of the form $(new\_vid, vcomp)$ corresponding to new view identifier and the view composition respectively. A disconnected process may install a new view consisting of its own; so, it can carry on its computation even in the disconnected mode.

In general, we can say that we have proposed a disconnection, failure and partition detection service that can be a foundation for a group communication systems. All the processes that are reachable at one moment can become part of the current view. One such membership algorithm has been proposed by [2]. They build their membership service on top of the set of the reachable processes. In the original algorithm, they use only the failure detectors to build the set $reachable$. We plan to reuse their membership algorithm on top of the partition detectors defined above. A view is formed for all the processes in the set $reachable$. The membership algorithm presented in their work is a comprehensive one. But the only difference lies in the underlying model. In our model, we consider links that do not recover after crashing while they consider links that can recover even after they crash.

8. Conclusions and Perspectives

Mobile Computing is gaining more and more importance with time. Need of the day is to target the problems which are emerging due to mobility of the terminals. Disconnection is one of such problems, which springs up when a mobile terminal moves out of the communication range. Another problem that may occur in both fixed and wireless network is the problem of process and link crashes. These may hamper the progress of a distributed application or may render it completely unavailable. Thus, there is a need to detect and correct such unexpected behavior. There can be a scenario where disconnection or failure of a process can render two sets of components completely unreachable, called partitions. Group communication systems define a powerful paradigm for distributed systems where processes take the very same steps and exchange the same set of messages in order to preserve the overall consistency of the application. Processes are organized as multicast groups, called views. Every process within a view shares the same set of messages, which is called view synchrony.

In this work, we have tried to distinguish the three: disconnection, failure and partition. Disconnection and failure may lead to the formation of autonomous network components which can only communicate within their groups. We have modified the already existing disconnection detector to work for partitionable networks with fair links. The already existing heartbeat failure detector for partitionable networks has been modified to discover the reachability patterns of the underlying network. In the first partition detector, we try to discover the neighborhood topology. The idea of neighborhood topology is to discover the processes that are reachable from some neighbor. If that neighbor disconnects and fails, all the processes that were reachable only through that neighbor are declared to have partitioned.

The second partition detector builds an overall view of the system with the initial processes and links. Afterwards, every new process and link is committed by all the processes. In case of failure or disconnection, the reachability between two processes is verified and if the disconnection and failure has affected reachability, the unreachable processes are added to the list of partitioned processes. If there are network partitions, every partition keeps track of the changes visible to the partition and this information is made global on merges. Since we cannot ascertain the status of partitioned processes, we try to develop the heuristics or opinions based on the application requirements. These speculations can be optimistic and pessimistic. Certain ideas for optimization of failure detection can be developed based on the collected topology information.

Wireless group communication systems are becoming need of the day with the increasing use of wireless applications. Users can share collaborative applications, they can play multi-player games, and they keep data consistency while they use wireless group communication systems. The partition detectors defined in Section 5.1 and Section 5.2 can support any membership service that builds it views consisting of the members of the reachable set.

Thus, we are giving here a framework for partition detection which is a general one, that is the solution proposed does not depend on network type or network topology. This framework can be adapted for various application types. We foresee that the disconnection, failure and partition detection can be based on one generic service, which can minimize the number of messages exchanged between various processes. For mobile terminals, where the battery is already too small to support normal operation of few hours, the computations and message complexity can be a huge burden. Consequently, there is a need for optimizing algorithms for their effectiveness.

References

[1] M. Aguilera, W. Chen, and S. Toueg. Using the Heartbeat Failure Detector for Quiescent Reliable Communication and Consensus in Partitionable Networks. *Theoretical Computer Science*, 220(1):3–30, June 1999.

[2] Ö. Babaoğlu, R. Davoli, and A. Montresor. Group Communication in Partitionable Systems: Specifi cation and Algorithms. *IEEE Transactions on Software Engineering*, 27(4):308–336, 2001.

[3] T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2), Mar. 1996.

[4] D. Conan, S. Chabridon, O. Villin, and G. Bernard. Disconnected Operations in Mobile Environments. In *Proc. 2nd IPDPS Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Ft. Lauderdale, Florida (USA), Apr. 2002.

[5] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.

[6] F. Gärtner. Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments. *ACM Computing Surveys*, 31(1):1–26, Mar. 1999.

[7] A. Montresor, R. Davoli, and Ö. Babaoğlu. Middleware for Dependable Network Services in Partitionable Distributed Systems. Technical report, University of Bologna, 1999.

[8] L. Temal and D. Conan. Détections de défaillances, de connectivité et de déconnexions. In *Proc. 1st Francophone Conference on Ubiquity and Mobility*, pages 90–97, Nice, France, June 2004. ACM Press. In French.