Toward a Solution to Partitionable Group Membership for MANETs

Léon Lim and Denis Conan
Institut Télécom, Télécom SudParis
UMR CNRS Samovar
{Leon.Lim,Denis.Conan}@telecom-sudparis.eu

Keywords: MANETs, dynamic partitionable systems, partitionable group membership, abortable consensus.

Context of the study

Ubiquitous computing environments are characterised by a diversity of mobile nodes and networks, and in particular, mobile ad-hoc networks. *Mobile Ad-hoc NETworks* (MANETs) are self-organising networks that lack a fixed infrastructure, and due to nodes arrivals, departures, crashes and movements, they are very dynamic networks. The topology changes occur both rapidly and unexpectedly and nodes (processes) can dynamically enter and leave the system. Thus, a distributed system built over MANETs can be partitioned. Distributed systems that are built over MANETs must be partition-tolerant, that is network partitioning may result in a degradation of services, but not necessarily in their unavailability. Fault-tolerant applications in partitionable system models generally rely on the two services of *Group Communication System*: *(1)* group membership service and *(2)* reliable multicast service. Informally, *group membership* specifies the *view* a process has on the current group it belongs to whereas *reliable multicast* provides reliable message diffusion within the same group. In this work, we focus on the group membership service, and more precisely, on group membership services for partitionable systems built over MANETs.

Group membership in dynamic and partitionable systems

Group membership consists of two sub-problems [10]: *(i)* determining the set of processes that are currently *up*, and *(ii)* ensuring that processes agree on the successive value of the this set. In the literature, two types of group membership services have emerged: primary-partition [9] and partitionable [5, 2]. Roughly speaking, *primary-partition* group membership maintains a single agreed view of the group. Such a group membership is intended for classical not-partition-tolerant systems. Since only a single view of the group can exist, primary-partition group membership requires strong assumptions on the system to satisfy the agreement property. These assumptions are strong enough to show that group membership is impossible in systems with crash failures [4]. In contrast to primary-partition ones, *partitionable group memberships* allow processes to disagree on the current membership of the group. Collaborative applications [3], resource allocation management [1], and distributed monitoring [7] are examples of applications that support permanent partitioning and thus go on running on multiple partitions. Such partitions may present some *eventual stability* [6] so that liveness of the computation can be guaranteed, that is a stability period lasts long enough so that eventually all the participant nodes of the partition are connected to each other and can communicate in a timely manner. Partitions can merge into larger partitions when the communication links between them are re-established. Thus, partitionable group memberships allow `split` and `merge` operations on partitions. By allowing disagreement, they escape from the impossibility result of [4]. However, they run into another fundamental issue. Specifying partitionable group membership for partitionable systems faces two orthogonal goals [4, 8]: *(1)* the specification must be weak enough to be solvable (implementable); and *(2)* it must be strong enough to simplify the design of fault-tolerant distributed applications in partitionable systems.

Open issues in the specification of partitionable group membership

To be useful, a partitionable group membership service must ensure at least the *virtual synchrony* property which is considered in the literature to be a basic property of partitionable membership services [5]. Two prominent specifications of partitionable group membership that ensure virtual synchrony are [5] and [2]. They sketch the two categories of partitionable group membership specifications which have been proposed in the literature and which differ about their liveness property: *(1)* liveness must hold only in stable partitions [5], and *(2)* liveness

must be ensured in every partition [2]. In [5], the authors define a completely stable partition as "a set of processes that are eventually alive and connected to each other, and the link from any process in this set to any process outside the set is down". The specification of [5] does not ensure liveness of the system when two processes p and q are forever intermittently reachable. This unstable case disappears in the specification of [2] with the consideration of fair channels, and thus [2] guarantees liveness not only in stable partitions. However, as shown in [8], these two specifications are not satisfactory. For instance, the specification in [5] can be satisfied by a "trivial but useless implementation" and the specification in [2] cannot be implemented without strong synchrony assumptions. Furthermore, the system model in both [5, 2] is static and the network is initially fully connected.

Partitionable group membership as a sequence of abortable consensus

In our work, we propose a system model that characterises the dynamic behaviour of stable partitions in MANETs. Nodes that stay in a partition during a period that lasts enough are said to be *stable*, and *unstable* otherwise. In our model, the liveness property of partitions can be guaranteed even if they are not completely stable. To this means, we have defined a weak stability condition based upon the application-dependant parameter α which is a threshold value used to capture the liveness property of a partition. In each partition, α stable processes are required to execute distributed computations. Then, we propose a way to solve group membership in partitionable systems built over MANETs by adapting the Paxos protocol for such systems. This results in a specification of a form of consensus for partitionable systems called abortable consensus. *Abortable consensus* ($\mathcal{AC}$) is a combination of two abstractions: eventual α partition-participant detector ($\Diamond\mathcal{PPD}^{\alpha}$) and eventual register per partition ($\Diamond\mathcal{RPP}$). $\Diamond\mathcal{PPD}^{\alpha}$ is specified to abstract liveness in a partition whereas $\Diamond\mathcal{RPP}$ encapsulates safety in the same partition. The role of $\Diamond\mathcal{PPD}^{\alpha}$ is to make trade-offs between agreement and progress by eventually detecting the *stability condition* of α processes in a partition and eventually providing the leader among them. The participating nodes are selected among reachable nodes by some stability criterion that may satisfy the stability condition. A *stability criterion* is a parameter that is used to determine which nodes are the most stable ones. Finally, $\Diamond\mathcal{RPP}$ provides a distributed storage in a partition with a write-once semantics. The acts of locking and storing a value in the register can fail in two cases: *(1)* in case of contention or *(2)* in case of non-satisfied stability condition. The first case is the same as in the original Paxos algorithm: a proposer abandons a proposal if some proposer has begun trying to issue a higher-numbered one, but the consensus instance is not necessarily abandoned. In the second case, the proposer not only abandons the proposal but also the consensus instance if there is no α stable processes in its partition.

Then, the partitionable group membership problem is solved by a transformation into a sequence of $\mathcal{AC}$, where each $\mathcal{AC}$ is executed by the participant nodes in the current view. When the decision returned by the abortable consensus is a set of processes α-Set, these processes are the members of the next view. However, unlike a regular consensus, the returned value is not necessarily a value that was proposed by some process. It could be a specific value *abort* meaning that the consensus has aborted because the stability condition is not satisfied. In this case, processes re-compute their own α-Set, and then begin executing a new $\mathcal{AC}$ instance.

References

[1] T. Anker, D. Dolev, and I. Keidar. Fault tolerant video on demand services. In *Proc. 19th IEEE ICDCS*, pages 244–252, 1999.

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

[3] K.P. Birman, R. Friedman, M. Hayden, and I. Rhee. Middleware support for distributed multimedia and collaborative computing. *Software: Practice and Experience*, 29(14):1285–1312, 1999.

[4] T.D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost. On the impossibility of group membership. In *ACM PODC*, pages 322–330, 1996.

[5] G.V. Chockler, I. Keidar, and R. Vitenberg. Group Communication Specifications: A Comprehensive Study. *ACM Computing Surveys*, 33(4):427–469, December 2001.

[6] A. Mostefaoui, M. Raynal, C. Travers, S. Patterson, D. Agrawal, and A. El Abbadi. From Static Distributed Systems to Dynamic Systems. In *Proc. 24th IEEE SRDS*, pages 109–118, Florianpolis, Brazil, October 2005.

[7] P. Murray. A Distributed State Monitoring Service for Adaptive Application Management. In *Proc. IEEE DSN*, pages 200–205, 2005.

[8] S. Pleish, O. Rütti, and A. Schiper. On the Specification of Partitionable Group Membership. In *Proc. 7th EDCC*, pages 37–45, May 2008.

[9] A. Schiper. Brief announcement: dynamic group communication. In *ACM PODC*, page 113, 2003.

[10] A. Schiper and S. Toueg. From Set Membership to Group Membership: A Separation of Concerns. *IEEE Transactions on Dependable and Secure Computing*, 3(1):2, 2006.