J. Parallel Distrib. Comput. 74 (2014) 2708–2721

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Partitionable group membership for Mobile Ad hoc Networks

L. Lim *, D. Conan

Institut Mines-Télécom, Télécom SudParis, CNRS UMR SAMOVAR, Évry, France

HIGHLIGHTS

- Distributed system model adapted to the dynamic characteristics of MANETs;
- Eventual α partition-participant detector for MANETs;
- Eventual register per partition for MANETs;
- Abortable consensus for MANETs;
- Abortable-consensus-based partitionable group membership.

ARTICLE INFO

Article history:
Received 21 June 2013
Received in revised form
24 January 2014
Accepted 9 March 2014
Available online 17 March 2014

Keywords:
Partitionable group membership
Dynamic partitionable systems
MANETs
Abortable consensus

ABSTRACT

Group membership is a fundamental building block that facilitates the development of fault-tolerant systems. The specification of group membership in partitionable systems has not yet reached the same level of maturity as in primary partition systems. Existing specifications do not satisfy the following two antagonistic requirements: (i) the specification must be weak enough to be solvable (implementable); (ii) it must be strong enough to simplify the design of fault-tolerant distributed applications in partitionable systems. In this article, we propose: (1) a new distributed system model that takes into account the formation of dynamic paths, (2) a specification of partitionable group membership for MANETS called $\mathcal{PGM}$, and (3) an implementation of $\mathcal{PGM}$ designed by adapting a well-known solution of primary partition group membership, namely Paxos. This results in the specification of an abortable consensus as the combination of two abstractions: an eventual α partition-participant detector and an eventual register per partition that guarantee liveness and safety per partition, respectively. Then, partitionable group membership is solved by transformation into a sequence of abortable consensus.

1. Introduction

Mobile Ad hoc NETworks (MANETs) are self-organising networks that lack a fixed infrastructure. The task of building distributed systems upon MANETs raises numerous challenges due to operational constraints: variability of wireless communication bandwidth and throughput due to unreliable physical layer, dynamicity of nodes arrivals and departures, heterogeneity of hand-held devices in terms of battery power and bandwidth capability, mobility behaviour of nomad users, etc. As a consequence, topology changes occur both rapidly and unexpectedly and nodes (processes) can dynamically enter and leave the system. In other words, a distributed system built over MANETs is partitionable. According to the CAP (*Consistency*, *Availability* and *Partition-tolerance*) theorem, it is impossible for a synchronous distributed system service to provide C, A and P at the same time [17,24]: consistency implies that each response to a request is atomic; availability requires that each node receiving a request must respond; partition-tolerance reflects the tolerance of message losses. If the system is partially synchronous then two of the three desirable properties can be achieved. Since network partitioning is an intrinsic characteristic of MANETs, P must be provided while ensuring some trade-off between C and A. In other words, network partitioning may result in a degradation of the services, but not necessarily in their unavailability: partitioned groups must continue to operate as autonomous distributed systems. Basically, there is a need to consider a mechanism (or middleware service) that manages partitioned groups explicitly in order to mitigate the effects of network partitions on C and A. Therefore, we tackle in this article the problem of partitionable group membership for MANETs.

Group membership is a middleware service that maintains views of the membership of the group at each process. A view is a list of processes (the members) with an identifier for unique

* Corresponding author.
E-mail addresses: leon.lim@telecom-sudparis.eu (L. Lim), denis.conan@telecom-sudparis.eu (D. Conan).

http://dx.doi.org/10.1016/j.jpdc.2014.03.003
0743-7315/

identification. In the literature, two types of group membership services have emerged: primary partition and partitionable services [21]. *Primary partition* group membership maintains a single agreed view of the core cluster of processes—i.e., the so-called primary partition. [28] shows that primary partition membership can be solved by a sequence of consensus, where each consensus is executed by the processes in the current view and the decision returned by the consensus is a set of processes; these processes are the members of the next view. *Partitionable group membership* maintains all partitions uniformly. By allowing concurrent views, partitionable group membership does not require strong agreement as [the one] in primary partition group membership [19], but its specification faces two antagonistic requirements: (i) the specification must be weak enough to be implementable; (ii) it must be strong enough to simplify the design of fault-tolerant distributed applications in partitionable systems. Collaborative applications [12], resource allocation management [6], and distributed monitoring [36] are examples of applications that support permanent partitioning and thus are able to run on multiple partitions.

Several partitionable group membership specifications have been proposed and surveyed in [5,8,21,38]. Two of the prominent specifications are in [8,21]. They sketch the two categories of partitionable group membership specifications that differ about their liveness property: (i) liveness must hold only in completely stable partitions [21], and (ii) liveness must be ensured in every partition [8]. However, one of the two requirements cited above is not satisfied in these specifications: the specification in [21] requires a strong stability condition that can be satisfied by a trivial but useless implementation; the specification in [8] requires a weak stability condition that cannot be implemented without assuming a model that is somewhat in contradiction with dynamic systems. As a consequence, we focus in this article on the specification of partitionable group membership that is implementable and strong enough.

Since consensus can be used to solve primary partition group membership, we argue that another type of consensus may be used to solve partitionable group membership. One of the well-known consensus protocols in primary partition systems is the Synod algorithm of Paxos [30,31]. Since Paxos is adaptable and makes use of a sequence of consensus natively, we propose a solution to the partitionable group membership problem by adapting Paxos.

The contribution presented in this article is threefold. Firstly, we define a distributed system model for MANETs with a weak stability condition based upon the application-dependent parameter named α, which is a threshold value used to capture the liveness property of a partition: α stable processes are required to execute distributed computations in a partition. We also define a heartbeat-counter-based stability criterion, which is a parameter that is used by processes to determine the most stable nodes among mutually reachable ones. Secondly, we adapt the Paxos protocol by following the methods proposed in [14,15]. This results in a specification of a form of consensus, called abortable consensus ($\mathcal{AC}$). $\mathcal{AC}$ is a combination of two abstractions: an eventual α partition-participant detector ($\diamond\mathcal{PPD}$) and an eventual register per partition ($\diamond\mathcal{RPP}$). For short, $\diamond\mathcal{PPD}$ is specified to abstract liveness in a partition whereas $\diamond\mathcal{RPP}$ encapsulates safety in the same partition. Then, the partitionable group membership problem is solved by a transformation into a sequence of $\mathcal{AC}$. Thirdly, we provide algorithms that implement all these abstractions, and proofs of the algorithms (presented in Appendices A and B).

The rest of the article is organised as follows. We define a dynamic distributed system model for MANETs in three steps: we present first elements in Section 2; we summarise in Section 3 the problems in existing specifications by focusing on the specifications in [21,8]; and, we enrich our system model with fairness, reachability and timeliness properties in Section 4. Afterwards, we specify abortable-consensus-based partitionable group membership in Section 5. Then, in Section 6, we implement the specification $\mathcal{PGM}$, including $\mathcal{AC}$ and its two modules $\diamond\mathcal{PPD}$ and $\diamond\mathcal{RPP}$. Finally, we discuss some related works and conclude the article in Sections 7 and 8, respectively.

2. Distributed system model

In this section, we define the first elements of our distributed system model by characterising the dynamicity of MANETs and presenting preliminary definitions.

2.1. Crash-recovery and infinite arrival models

We consider a dynamic distributed system composed of infinitely many mobile nodes. We assume that nodes are uniquely identified, and consider one process per node. Thus, the system consists of an infinite countable set of processes $\mathbb{P} = \{\ldots p_i, p_j, p_k \ldots\}$. A process is *correct* in an execution if it does not fail in that execution. We consider the crash-recovery model in which each process has, in addition to its regular volatile memory, a stable storage that allows the process to store part or all of its state via the primitive *store*(). This allows the process to restart upon failure with the same identifier by recovering its state via the primitive *recover*(). We assume the *execution integrity* property stipulating that the recovery of a process is necessarily preceded by its failure.

Nodes can dynamically enter the system or leave it by crashing, recovering, disconnecting, or reconnecting. By definition [2,34], the number of processes that have joined the system minus the number of departures is called *concurrency*. We then consider the *infinite arrival model with bounded concurrency* investigated in [2,34]: in any bounded period of time, only finitely many nodes take steps; the total number of nodes in a single execution may grow to infinity as time passes; however, each execution has a maximum concurrency that is finite but unknown. In addition, processes do not know $\mathbb{P}$—i.e., the processes in $\mathbb{P}$ do not necessarily know each other.

2.2. Asynchronous event-based composition model

We consider the *asynchronous event-based composition model* [27] to specify interfaces and properties of distributed algorithms and systems. In this model, each process is composed of a set of software modules called components. Each primitive or composite component is identified by its name and is characterised by an interface presenting different types of events that the component can accept and produce. Distributed algorithms are typically made of a collection of at least one component per process and these components are supposed to satisfy some properties. We consider $\mathbb{E}$ to be the set of possible events including at least the events store(), crash() and recover(). Thereafter, $\mathbb{E}$ is enriched with other events related to group membership and MANETs.

2.3. Execution, global history and causal order

The execution of a process is modelled by a sequence of events. The local history of process p during an execution is a sequence of events or absences of events $h_p = (e_p^1 e_p^2 e_p^3 \ldots)$, where an absence is denoted by ϵ and e_p^i corresponds to the ith event of process p. The index p and the exponent i are omitted when the context is clear. In addition, we consider the existence of a discrete global clock that is not accessible to the processes. We take the range $\mathbb{T}$ of the clock's tick to be the set $\mathbb{N}$ of natural numbers. The global history of an execution is a function $H : \mathbb{P} \times \mathbb{T} \to \mathbb{E} \cup \epsilon$. If p executes an event $e \in \mathbb{E}$ at instant t then $H(p, t) = e$; $H(p, t) = \epsilon$ meaning that p does not execute any event at instant t. Let $I \subseteq \mathbb{T}$ be an interval, we write $e \in H(p, I)$ if p executes some event e in I—i.e., $\exists t \in I : H(p, t) = e$. We also consider the "happened-before" relation between events as defined in [29].