SITAN: Services for Ad Hoc Networks with Unknown Participants

David Matos, Nuno Neves and Alysson Bessani
Universidade de Lisboa, Faculdade de Ciências, LASIGE
dmatos@lasige.di.fc.ul.pt, {nuno,bessani}@di.fc.ul.pt

***Abstract*—The evolution of mobile devices with various capabilities (e.g., smartphones and tablets), together with their ability to collaborate in impromptu ad hoc networks, opens new opportunities for the design of innovative distributed applications. The development of these applications has however to address several difficulties, such as the unreliability of the network, the imprecise set of participants, or the presence of malicious nodes. In this paper we describe a middleware, called SITAN, that offers a number of communication and coordination services specially conceived for these settings. These services are implemented by a stack of Byzantine fault-tolerant protocols, enabling applications that are built on top of them to operate correctly despite the uncertainty of the environment. SITAN was evaluated in the network simulator NS-3, with results showing that it can perform well in many scenarios. An implementation in the Android OS is currently under way.**

I. Introduction

The proliferation of mobile devices with different capacities can support the execution of an increasing number of distributed applications (e.g., group cooperation apps, like a shared agenda, poll support, or file sharing). These applications usually require broadcast/multicast capabilities, together with operations that assist on the collaboration, which are tricky to implement in a mobile ad hoc network (MANET).

This paper presents SITAN, a middleware that offers services for the implementation of applications in ad hoc networks. Three major challenges had to be addressed in the design of SITAN. *Lack of knowledge* is something inherent of this type environment, and it is perceived by having an inaccurate view of which nodes are present in the network and how they are connected to each other. Moreover, since nodes are mobile, the knowledge about the network organization can become outdated relatively fast. Another problem of MANETs is the fact that wireless communications, due to their exposure and simplicity of access, are vulnerable to many *distinct forms of attack*. Therefore, it is advisable to consider that potentially some nodes may become compromised by a malicious adversary (i.e., they suffer a Byzantine failure) and that parts of the network may also misbehave. Lastly, in order for the middleware to be useful, it should be *efficient*. This means that protocols should take advantage of the characteristics of MANETs, such as the ability to broadcast messages to a node's neighbors with little cost.

SITAN is organized as a stack of protocols, which are divided in three main layers: communication, group membership and agreement (see Fig 1). The protocols were in some cases adapted and optimized from previous proposals, while others are novel. All protocols were implemented from scratch. The two lower layers follow the organization described by Alchieri et al. [1]. A straightforward implementation of the layers had however several limitations in performance, and consequently it was necessary to redesign some of the protocols. The binary consensus protocol is based on Turquois by Moniz et al. [4]. This algorithm was proposed for resource-constrained devices and it circumvents two impossibility results relevant to MANETS [3], [5]. The multivalue and vector consensus protocols are new (see [2] for a description of the properties of these consensus protocols).

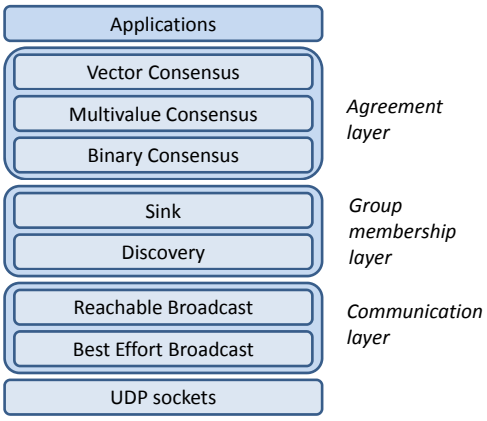

Fig. 1. SITAN protocol stack

II. System Model

The system is composed by a group Π of processes (also called participants or nodes) belonging to larger universe U. Since the network is created in an ad hoc manner, a process $i \in \Pi$ may only be aware of a subgroup $\Pi_i \subseteq \Pi$. Processes can suffer Byzantine faults, which means that they can become silent, send messages with wrong values, or work together with other Byzantine processes to corrupt the system. It is assumed a bound f on the number of faulty processes in Π. The protocols ensure the correctness of the system as long as $n \geq 3f + 1$ (where n is the total number of nodes in Π). The fault model also assumes that the network can experience Byzantine faults. For example, it can have omissions, reorder or corrupt messages.

A process i can only send a message directly to another process j if $j \in \Pi_i$, i.e., if i knows j. As expected, if i receives a message from j and $j \notin \Pi_i$, then i will add j to Π_i (i.e., i becomes acquainted to j and can send messages to it). All processes have a unique identifier and a pair of public (pu_i)

and private (pr_i) keys. Processes attempt to cache a copy of the public key of the other processes, but in case it is not available, the owner of the public key can always append it to its messages. Every process has the ability to check the validity of public keys of other processes (e.g., public keys are placed in a certificate signed by a well known entity). Public/private keys are used to disseminate session keys in a secure way, which then support the creation of authenticated reliable (point-to-point or multicast) channels.

With regard to synchrony, the system is assumed to be asynchronous. All protocols make progress at the speed that the network delivers messages, meaning that if the network is slow they will take longer to conclude (e.g., if the network is under attack, processes will continue to retransmit messages until their eventual arrival, allowing the protocols to complete). The binary consensus protocol is able to terminate under this model with high probability by employing a randomization technique [4].

III. Protocol Stack

The protocol stack is composed by a communication, a group membership and an agreement layer (see Fig 1). Applications can call operations in any of three layers to take advantage of the available services.

The communication layer uses the underlying broadcast primitive (UDP) of the mobile devices to disseminate values. The *Best Effort Broadcast* sends messages securely to the immediate neighbors of a source i. As messages arrive to j, the set of known processes in Π_j is updated. *Reachable Broadcast* allows a process i to broadcast a message reliably to every participant that can be $reached$ in the ad hoc network. When the message is propagated through the network, it includes data about the nodes that were visited. As side effect, when the message arrives at a process, it can support the construction of a graph with the locally perceived network organization. Notice however that this graph is potentially incomplete and distinct among nodes (e.g., some connections may be missing because they go through a Byzantine node that erases packets).

The membership layer finds a subgroup of processes that has sufficient connectivity to run consensus, which is called the *sink* of the network. The *Discovery* protocol enables every participant to expand the local knowledge, letting a process find out the set with the maximum number of nodes that it can connect to in the ad hoc network. The *Sink* protocol takes this information and defines the sink, ensuring that everybody recognizes who belongs to this subgroup.

Consensus protocols are run among the *sink* processes, and once they terminate, the decided value is propagated to the rest of the network. *Binary consensus* is the lowest protocol on the consensus layer, and it allows processes to agree on a value $v \in \{0, 1\}$. The communication primitive used by consensus is the reachable broadcast, which disseminates messages in the *sink*. Next, there is *Multivalued consensus* that extends binary consensus and decides on non-binary values (i.e., values from an arbitrary domain). The top protocol implements *Vector consensus* that agrees on a set of values. This set has the property that the majority of values was proposed by correct processes, ensuring that various other operations can be implemented at the application layer (e.g., a ballot).

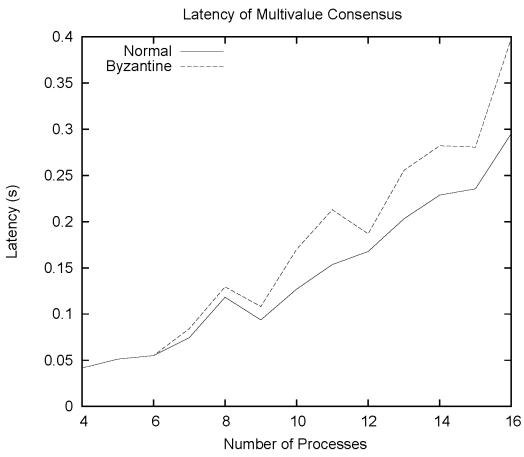

Fig. 2. Latency of multivalued consensus with divergent proposals.

IV. Results

Fig. 2 displays results collected in the NS-3 simulator for the latency of multivalued consensus when proposals are divergent (worse case setting). In this example, all nodes belong to the sink, and values are displayed for normal (fault-free) and Byzantine executions. Byzantine faults were applied to $f = \lfloor (n-1)/3 \rfloor$ processes, by maliciously changing the identity of the message sender field (a personification attack). The graph shows that although there is an increasing latency when the number of processes grows, in both cases the times are at most a few hundred milliseconds, which quite satisfactory for MANET applications.

V. Conclusion

This paper describes a middleware that can assist on the development of applications for MANETS. The middleware is implemented by a number of protocols, allowing the configuration and management of the network and supporting the execution of a few communication primitives and three flavors of agreement. These protocols were designed to address the main characteristics of MANETS, including Byzantine failures and unknown participants. To the best of our knowledge, this is the first middleware that has all these characteristics.

Acknowledgments: This work was supported by the EC through projects FP7-607109 (SEGRID), and by the FCT through the SITAN project (PTDC/EIA-EIA/113729/2009) and the LaSIGE Strategic Project (PEst-OE/EEI/UI0408/2014).

References

[1] E. Alchieri, A. Bessani, and S. Fraga. Byzantine Consensus with Unknown Participants. In *Proc. of the International Conference on Principles of Distributed Systems*, 2008.

[2] M. Correia, G. Veronese, N. Neves, and P. Verissimo. Byzantine Consensus in Asynchronous Message-Passing Systems: a Survey. *International Journal of Critical Computer-Based Systems*, 2(2), 2011.

[3] M. Fischer, N. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2), 1985.

[4] H. Moniz, N. Neves, and M. Correia. Turquois: Byzantine Consensus in Wireless Ad Hoc Networks. In *Proc. of the International Conference on Dependable Systems and Networks*, 2010.

[5] N. Santoro and P. Widmayer. Time is Not a Healer. In *Proc. of the Annual Symposium on Theoretical Aspects of Computer Science*, 1989.