

A Middleware for Constructing Highly Available, Fault Tolerant, and Attack Tolerant Services

Shivakant Mishra
Department of Computer Science
University of Colorado, Campus Box 0430
Boulder, CO 80309-0430, USA.

Abstract

This paper describes the design of a middleware that provides support for constructing highly available, secure, fault-tolerant, and attack-tolerant services. The central component of this middleware is a group communication service that comprises of six network protocols: atomic broadcast, group membership, failure detection, attack detection, group access control, and secure intermember communication protocols. The paper describes the motivation behind each of these protocols and discusses their functionalities.

Keywords: Attack tolerance, high availability, security, fault tolerance, survivability.

1 Introduction

There are two clear trends in modern computing systems: (1) its complexity is increasing significantly, and (2) its use in constructing critical as well as non-critical applications is increasing exponentially. These two trends call for a fundamental research in the area of high availability and dependability support of modern computing systems. Dependability of a computing system refers to the ability of that system to continue to operate correctly and provide the correct services even in the presence of component failures and malicious security attacks. In fact, with increasing use of modern computing systems in our daily lives, any future computing system that does not provide adequate support for dependability will not only be useless, but it may cause serious damage to the institution where it is used.

The fundamental requirement of the next generation of computing services is that they *always* remain correct and available (high availability), even in the presence of significant increase in communication

or processing loads, one or more component failures (fault tolerance), and malicious security attacks (attack tolerance). Current computing systems cannot support the construction of services with such stringent requirements. The key reason for this limitation is that while the issues of high availability, fault tolerance, and security have been addressed at various depths by different researchers, they have mostly been addressed independent of one another. When techniques developed from these different research efforts are integrated in a single system, they often conflict with one another and result in a substandard system.

This paper describes the design of a middleware to support the construction of computing services that provide high availability in the presence of one or more component failures and malicious security attacks. The important features of this middleware are support for replication, fault tolerance, security, and attack tolerance. The central component of this middleware is a group communication service consisting of six network protocols: atomic broadcast protocol, group membership protocol, group access control protocol, secure intermember communication protocol, failure detection protocol, and attack detection protocol.

This paper makes three important contributions in the area of building highly available and dependable distributed services. First, it addresses the issues of high availability, fault tolerance, security, and attack tolerance in an integrated manner. This has resulted in avoiding any conflicts that commonly occur between techniques developed independent of one another to address these issues. To the best of our knowledge, we are not aware of any other research project that has addressed these four issues in an integrated manner. Second, this paper addresses the important issue of attack tolerance, i.e. the ability of a system to operate correctly even when the security of one or more of its components has been compromised. This is relatively

a new research area. Its focus is to improve computing system dependability by isolating system components whose security has been compromised, and allowing the rest of the system to continue to operate correctly. Finally, this paper extends the traditional functionalities of group communication services by incorporating support for security and attack tolerance.

2 Issues

The fundamental challenge in designing a middleware that provides an integrated support for high availability, fault tolerance, security, and attack tolerance is integrating the existing fault-tolerance techniques and security techniques. Integrating these techniques is an extremely difficult task. Some of the important reasons for this difficulty are:

1. Fault-tolerance techniques rely on redundancy in a system. For example, object replication, which is a popular technique to deal with object failures, results in redundant objects in a system [4]. The key idea these techniques exploit is that if a component fails, another component can take over the failed component's functionalities. However, redundancy in a system increases the vulnerability of that system to security violations. As a result, security techniques typically attempt to avoid redundancy in a system.
2. Security techniques typically rely on restricting access to a service and provide access to an entity only after extensive credential checks. For example, a mobile agent is allowed to run on a server only after it provides sufficient credentials in the form of certificates [2]. Fault-tolerance techniques, on the other hand, rely on the openness of a system. These techniques make decisions based on observing the behavior of a system. For example, failure detectors observe the external behavior of an object to determine object failures.
3. Security techniques often attempt to hide the identity of a system by purposely providing false information. For example, the TCP/IP stack fingerprint scrubber described in [5] converts ambiguous traffic from a heterogeneous group of hosts into sanitized packets to avoid revealing the operating system running on a host. Fault-tolerance techniques, on the other hand rely on a cooperative interaction between redundant entities.

3 Middleware

3.1 Architecture

Figure 1 shows the architecture of the proposed middleware. The central component of this middleware is a group communication service [1, 3]. Clients interact with a group (one of the group members) using a special protocol called the Group Access Control Protocol (GACP). This protocol is intended to allow secure communication between trusted clients and trusted groups. Group members interact with one another using a special protocol called the Secure Intermember Communication Protocol (SICP). Again, this protocol is intended to provide support for secure communication between trusted group members.

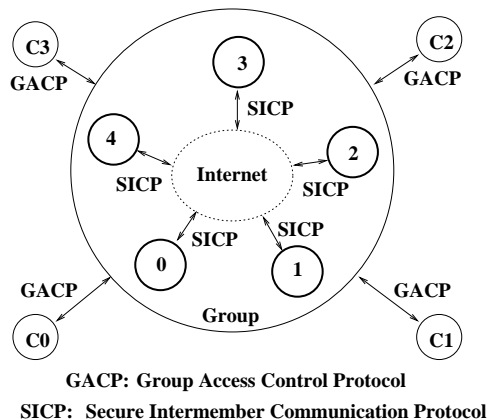


Figure 1: Preliminary Architecture.

Figure 2 shows the dependencies between different protocols that comprise the proposed middleware. The atomic broadcast and group membership protocols are the standard protocols provided by a group communication system to support replication and fault tolerance. The atomic broadcast protocol provides a multicast mechanism by which different group members can exchange messages in such a way that (i) all group members deliver messages to their application in the same order (*order property*), and (ii) either every group members delivers a message or none do (*atomicity property*). The group membership protocol maintains a consistent view of group membership in the presence of member failures and new members joining the group.

Traditionally, a group membership protocol depends on a failure detection protocol [3] that reports group member failures to the group membership protocol. We have extended this design by including an attack tolerance protocol in the proposed middleware.

Like the failure detection protocol, the attack detection protocol also monitors all group members. It reports to the group membership protocol, whenever it suspects that a group member is under security attack and its security has been compromised.

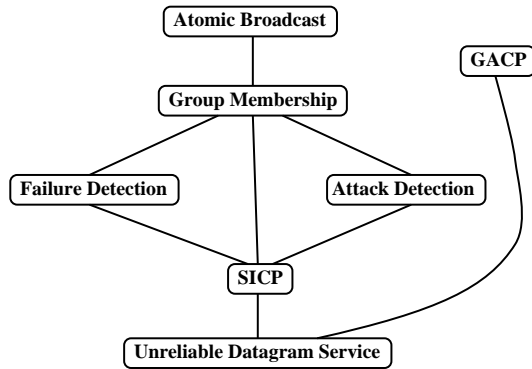


Figure 2: Protocol Dependencies.

3.2 GACP and SICP

One of the challenges of operating in an insecure computing environment is to allow trusted entities to interact with one another with as little restrictions as possible and, at the same time, prevent untrusted entities from interacting with the trusted entities or intercepting the information exchanged between trusted entities. In the context of the proposed middleware, this implies that we need appropriate mechanisms for allowing interactions between trusted clients and trusted group members, and among different trusted group members. Protocols GACP and SICP have been designed to provide these functionalities to the group members and the clients.

The main goal of GACP is to allow authorized clients to interact with an authorized group (one of the group members) in a secure manner. There are three specific functionalities that this protocol provides:

1. **Group authentication:** A client can authenticate a group before it interacts with it. This is to prevent malicious entities, who may be imposing as a group, from interacting with the client.
2. **Client authentication:** A group (one of the group members) can authenticate a client before it interacts with it. This is to prevent malicious entities, who may be imposing as clients, from interacting with the group.
3. **Secure communication:** Clients can exchange information with group members in a secure man-

ner. This prevents malicious entities from gaining access to the information exchanged.

The requirements of GACP are very similar to the requirements of the agent admission control protocol that we designed and implemented as a part of the DaAgent project [2]. GACP design is based on the design of the DaAgent agent admission control protocol, and we are currently implementing it using secure socket layer (SSL) [6].

The main goal of SICP is to allow authorized group members to interact with one another in a secure manner. There are two functionalities of this protocol:

1. **Member authentication:** A group member can authenticate another group member before it interacts with the other group member. This is to prevent malicious entities, who may impose themselves as group members, from interacting with real group members.
2. **Secure Communication:** Group members can exchange information with one another in a secure manner. This prevents malicious entities from gaining access to the information exchanged.

The requirements of SICP are simpler than the requirements of GACP. This is because there are only a limited number of well-known entities (group members), as opposed to unlimited number of not-so-well-known clients in GACP, participating in this protocol. The design of this protocol is again based on the design of the DaAgent agent admission control protocol, and it is being implemented using SSL.

3.3 Attack Tolerance

Security protocols such as GACP and SICP are intended to prevent the proposed middleware from allowing malicious entities to gain access in the system. However, it may still be possible that a particular system on which a group member is running may be compromised by some malicious intruders. This will result in the group member being compromised. In the proposed middleware, we plan to address this problem by isolating the compromised group member and removing it from the group. We call this feature of the middleware attack tolerance—the middleware will continue to provide the its functionality even when the security of some of its parts has been compromised.

The key issue in making the proposed middleware attack tolerant is how do other group members learn or detect that a particular group member has been compromised. This is a very difficult problem because

a compromised group member may attempt to fool other group members into believing that it is behaving according to its specification.

At present, we have addressed this problem by making use of the intrusion detection software. An intrusion detection software monitors each group member for signs of intrusion. This software has been integrated with the attack detection protocol that notifies other group members as soon as an intrusion in a group member is detected. Figure 3 shows this design.

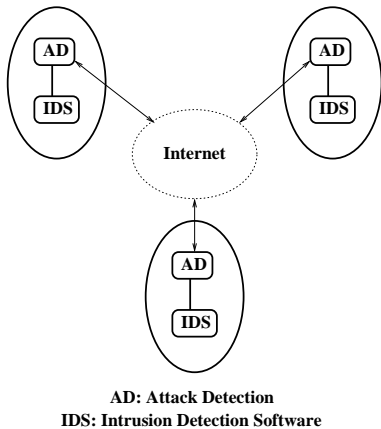


Figure 3: Architecture of Attack detection.

We plan to use another alternative to address this problem. We plan to experiment with detecting if a group member has been compromised by observing its external behavior. In this alternative, each group member will observe other group members for signs of intrusion. These signs can be based on anomalous behavior or finger prints as used in the design of intrusion detection software.

3.4 Robust Group Membership Protocol

Interactions between the group membership protocol and the security protocols such as SICP, GACP, and attack detection require that the design and implementation of group membership protocol be addressed all over again. Here are the some of the new problems that the group membership protocol in the proposed middleware addresses:

1. When a new group is formed, the membership protocol does not allow malicious entities to participate in the new group formation.
2. When a new group is created to exclude one or more compromised group members of an earlier group, the membership protocol ensures that the

compromised group members are prevented from participating in the new group formation.

3. The membership protocol addresses the scenario where a compromised member may report (perhaps with malicious intentions) that another (correct) member has failed or been compromised.

4 Conclusion

High availability, security, fault tolerance, and attack tolerance are the fundamental requirements of modern computing services. Addressing these issues in an integrated manner is perhaps the most important challenge facing us today. This paper describes the design of a middleware that addresses all these four issues in an integrated manner. The central component of the proposed middleware is a group communication service that comprises of six network protocols that together provide support for high availability, security, fault tolerance, and attack tolerance. The proposed middleware is currently being implemented.

References

- [1] K. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atomic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, Aug 1991.
- [2] S. Mishra, Y. Huang, and H. Kuntur. Programming environment of DaAgent. In *Proceedings of the 12 IASTED International Conference on Parallel and Distributed Computing Systems*, Las Vegas, NV, Nov 2000.
- [3] S. Mishra, L. Peterson, and R. Schlichting. Consul: A communication substrate for fault-tolerant distributed programs. *Distributed Systems Engineering*, 1(2):87–103, Dec 1993.
- [4] F. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, Dec 1990.
- [5] M. Smart, R. Malan, and F. Jahanian. Defeating tcp/ip stack fingerprinting. In *Proceedings of the 9th USENIX Security Symposium*, Aug 2000.
- [6] SSL. The SSL protocol, version 3. Available from <http://home.netscape.com/eng/ssl/ssl-toc.html>, March 1996.