

Middleware Support for Constructing Highly Available and Dependable Services

Shivakant Mishra and Soontaree Tanaraksiritavorn
Department of Computer Science
University of Colorado, Campus Box 0430
Boulder, CO 80309-0430, USA.
Emails: {mishras|tanaraks}@cs.colorado.edu

The main goal of this project is to design, implement, and evaluate a middleware that provides an integrated, system-level support for constructing highly available and dependable applications. This middleware is intended to support the construction of applications with 5 important characteristics: high availability, trustworthiness, attack tolerance, fault tolerance, and high performance. The central component of this middleware is a group communication service, which provides support of replicating objects, and an integrated support for security, attack tolerance, and fault tolerance in a hostile computing environment.

Group communication services have been successfully used to construct critical and non-critical applications. Most of these services provide support for object replication in the presence of failures. Recently, some group communication services have been extended to provide support for security of information exchanged with in a group. The main goal of our project is to extend these services to provide support for attack tolerance. We are (re)designing two group communication service components: trustworthiness detector and trustworthy group membership protocol.

1 Trustworthiness Detector

Trustworthiness detector is a very important component of this middleware. The main goal of this component is to raise a suspicion *in a timely manner*, whenever one or more group members cannot be trusted. This suspicion may eventually result in the creation of a new group by the group membership protocol. Some important challenges in building a trustworthiness detector are: (1) it is not clear what trustworthiness of a component means, (2) timeliness of a trustworthiness detector is critical, (3) performance of the group communication system can degrade if the trustworthiness detector reports too many false positives, (4) a malicious entity may be able to attack and compromise multiple group members simultaneously, (5) a compromised group member may try to defeat the trustworthiness detection process, and (6) if multiple group members are compromised

with in a short time interval, a coordinated attack from multiple group members to defeat the trustworthiness detection process is possible. We are focusing on four approaches to address each of these challenges:

1. focus on the observable effects,
2. detection in depth,
3. asymmetric design, and
4. use of a trusted computing component (TCC).

To understand what the trustworthiness of a group member really means, we plan to borrow ideas from the fault-tolerance research. In particular, we observe that the design of a fault-tolerant system is typically based on the observable behavior of a component (failure models). It is important to note that failure models do not focus on causes of failures. On similar lines, we plan to focus on the observable effects of a component in designing the trustworthiness detector. For example, an observable effect is whether the component is responding to an event in an expected time period. Another observable effect is whether the component has deviated too far from its normal behavior.

The second approach we plan to use is “detection in depth”. The basic idea is to structure various mechanisms of different capabilities for detecting any signs of untrustworthiness in a component in several distinct layers. A malicious entity that attempts to defeat the detection process will have to defeat each of these layers. As a starting point, we plan to experiment with four distinct layers in the proposed detection in depth technique: (1) operating system level, (2) network level, (3) application level and (4) peer level. At the operating system level, an appropriate OS-based intrusion detector can be used, while at the network level, an appropriate network-based intrusion detector can be used. Detection mechanism at the application level can watch out for signs of abnormal conditions such as one or more function invocations being blocked for a long time, excessive number of timeouts, object crashes, etc.

Peer-level detection mechanism consists of a set of peer-level detectors, each running at a different node. A peer level detector monitors one or more group members, and watches for signs of untrustworthiness in those members. Notice that a peer-level detector runs on a node that is different from the nodes on which the member(s) it is monitoring is (are) running. A peer-level detector can watch out for conditions that may be abnormal in a group communication environment. For example, trustworthiness of a group member can be suspected if it has suddenly started delaying the stability of messages multicast in a group.

To address the issue of coordinated attack to defeat a trustworthiness detector, we are investigating an asymmetric design. The basic idea is that the detection mechanisms at different group members differ from one another in various aspects. For example, they may differ from one another in the algorithms and data structures used, design techniques employed, implementation details, and so on. For this, we are also investigating the use of code obfuscation techniques to transform one implementation of a protocol used in one group member to a different implementation to be used in another group member. These differences are intended to make it hard for a malicious entity to break the trustworthiness detectors of several group members with in a short period of time.

Finally, we are investigating the use of a TCC to check the trustworthiness of a group member by sending appropriate inquiry messages to that member and analyzing the response time and the response received. Advantage of using a TCC to do this is that since TCC is a trusted component, any abnormalities detected can be assumed to originate from the target member.

2 Trustworthy Group Membership Protocol

A group membership protocol maintains a consistent, system-wide view of the membership of a group. It creates a new group in the beginning, and changes membership whenever some member fails, or a new process wants to join the group. A need to tolerate security attacks, and ensure the security of the membership protocol itself raises some new challenges: (1) compromised group members should be removed from the group as soon as possible, (2) security of the messages exchanged as a part of the membership protocol should be maintained, (3) only trusted entities should be allowed to participate in a group membership protocol, and only trusted entities should become the members of a group, and (4) the membership protocol should itself be able to withstand some common security attacks, including replay and denial of service attacks. We are focusing on two approaches to address each of these challenges:

1. authentication via TCC, and

2. suspended membership state.

To ensure that only trusted entities participate in a membership protocol, and only trusted entities become members of a group, we need a mechanism that ensures that various entities participating in a membership protocol are authenticated before their participation. In addition, we need mechanisms to ensure that unauthenticated entities are not able to interfere with the membership protocol. We plan to investigate a design that uses a TCC for authentication and generating various cryptographic keys needed by authenticated entities to exchange membership messages in a secure manner. Using a TCC for authentication in the membership protocol results in a clean separation between the algorithms used to establish a group membership and the security mechanisms used to make the protocol trustworthy. We plan to design this TCC to allow a choice of actual authentication mechanism used based on the application requirements, and intrusion tolerant based on techniques to protect keys stored in TCC from potential intruders.

Experience from intrusion detection research has shown that it is not easy to detect intrusions, and most current intrusion detectors suffer from high false positives rate, i.e. they report an intrusion when there is none. This false positives rate becomes even higher, if there are timeliness constraints. As mentioned earlier, timeliness is an important requirement of a trustworthiness detector or a membership protocol. This requirement has a side effect that is critical in a group communication service. A high false positives rate of a trustworthiness detector means that members that have not been compromised will be suspected, and the membership protocol will be initiated to remove them.

However, performance of an application is typically degraded, and sometimes the application can even be blocked, while the membership protocol is in the process of forming a new membership. Thus, on one hand a trustworthiness detector that reduces the false positives rate may detect the signs of untrustworthiness too late. On the other hand, a trustworthiness detector that attempts to report signs of untrustworthiness in a timely manner may result in worsening the application performance.

We address this problem by introducing a new membership state of a group member: a *suspended group membership state*. When a trustworthiness detector suspects a group member, its state is changed to a suspended group membership state. The basic idea is that a member with a suspended group membership state is not allowed to participate in group communication activities. However, the membership protocol is initiated to create a new membership only after the untrustworthiness of the suspended member has been thoroughly tested. This testing can come from either a more extensive testing of the suspected member by a collaboration of the trustworthiness detectors of the current group members, or by making use of a TCC.