

SURVIVABILITY THROUGH DYNAMIC RECONFIGURATION

Subramaniam R. Sthanu

<subbu@mit.edu>

Steven R. Lerman

<lerman@mit.edu>

Thomas M. Parks

<parks@ll.mit.edu>

Center for Educational Computing Initiatives
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139

Lincoln Laboratory
Massachusetts Institute of Technology
244 Wood St.
Lexington, MA 02173

ABSTRACT

Information survivability encompasses many aspects of security and reliability for computers, communication networks, and information systems in general. In this paper, we concentrate on the use of dynamic reconfiguration as a diversity technique to survive denial-of-service attacks. By dynamically switching among several configurations, a group communication session can continue despite successful attacks against individual network nodes or particular protocols. To demonstrate this concept, we are developing a collaborative planning tool with a shared drawing area, a text-based chat area, and voice communication. This tool is implemented in the Java programming language.

INFORMATION SURVIVABILITY

Information survivability is more than just computer security. Survivability can be defined as the ability of a system to complete its mission, in whole or in part, in a timely manner, despite the incapacitation of significant portions of the system by attack or accident. The construction of large-scale information systems from off-the-shelf components whose internal structure is quite well known to the community at large, combined with the complexity of designing and implementing software, suggests that no amount of hardening can guarantee the invulnerability of a system to external attacks [1]. It is because of this inability to build breach-proof systems, that we need to build systems that are robust in the presence of attacks. Systems

should be able to survive even those attacks that cannot be completely repelled.

Our primary concern is the ability of a distributed information system, such as our prototype collaborative planning tool, to gracefully degrade and survive denial-of-service attacks directed at individual network nodes or particular protocols.

DYNAMIC RECONFIGURATION

We propose to use dynamic reconfiguration as a survivability mechanism. We are using Java programming language to take advantage of the class libraries provided for high-level network programming. The system can be pre-programmed with several diverse communication configurations. When a disruptive attack is detected, the system switches from its current configuration to a new configuration where the impact of the attack will be less severe and the system can continue to function uninterrupted.

Suppose, for example, that the server in a client-server collaborative planning session crashes or becomes overloaded due to a denial-of-service attack. At this point the system is reconfigured in reaction to the attack: the clients disconnect from the server and fall back to direct peer-to-peer connections.

DESIGN ELEMENTS

Distributed information systems can be implemented with one of several different communication configurations. Each configuration can be implemented by one of several protocols. We describe several configurations and protocols relevant to the design of the group communication architecture for a collaborative planning tool.

*The work of Subramaniam R. Sthanu and Steven R. Lerman was conducted through collaborative participation in the Advanced Telecommunications and Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

† The work of Thomas M. Parks was sponsored by the Department of the Air Force. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Air Force.

Presented at the Advanced Telecommunications and Information Distribution Research Program Conference. College Park, Maryland. February 1998.

Peer-to-Peer: In a peer-to-peer configuration, all communicating network nodes are equals, with no central controller. Group communication can be accomplished with multiple point-to-point (unicast) connections. If there are N participants then each of them has N-1 connections to the other participants. This requires a total of $N(N-1)/2$ bi-directional connections.

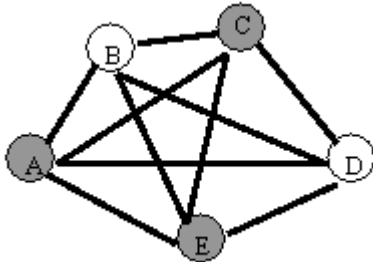


Figure 1. Peer-to-peer configuration among five nodes.

This is a robust configuration, for even if one of the network nodes is attacked the remaining nodes can continue to communicate uninterrupted. But this configuration has a disadvantage in that it does not make good use of bandwidth. This configuration is not efficient for large numbers of participants.

Client-Server: In a client-server configuration, a central server communicates with a group of clients. The clients depend on the server and are subordinate to it. Group communication can be accomplished with multiple unicast connections between the server and the clients. This requires only N bi-directional connections to support N clients.

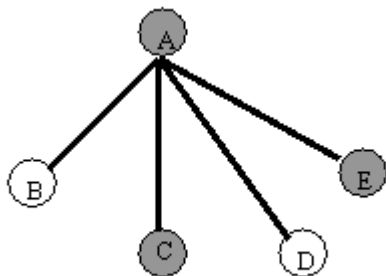


Figure 2. Client-server configuration with a server (A) and four clients (B, C, D and E).

This configuration makes more efficient use of bandwidth than a peer-to-peer configuration, but it is not as robust. A denial-of-service attack on the server could slow or even stop the entire system. Client-server, being one of the most

common configurations for distributed information systems, needs robustness against such attacks to be built into the system for more reliable service.

Multicast: Group communication can be accomplished by conventional unicast, with the sender establishing separate connections to each receiver. As long as the group is small, this approach is reasonable. But as the group size increases, the inefficient utilization of bandwidth can affect performance.

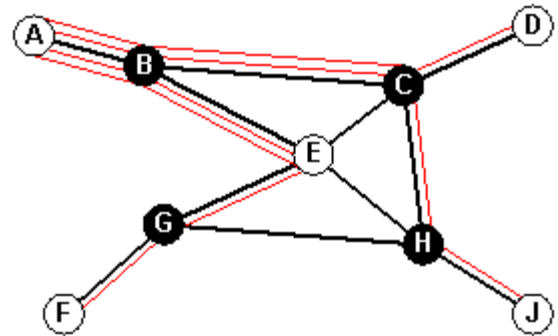


Figure 3a. Unicast connections from a single sender (A) to multiple recipients (D, E, F and J).

Multicast is a more scalable approach in which data streams are replicated by routers within the network instead of requiring the sender to generate redundant data streams. This makes much more efficient use of bandwidth because no more than one copy of a data stream traverses any link in the network.

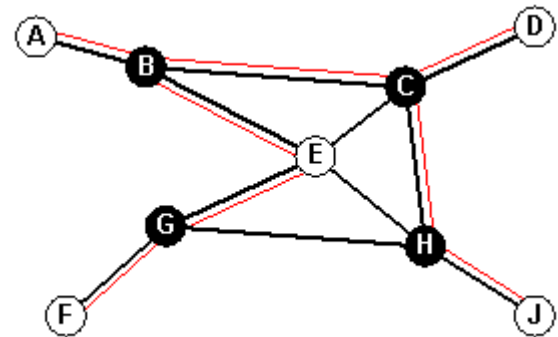


Figure 3b. : Multicast from single sender (A) to multiple recipients (D, E, F and J).

Our goal is to build a system capable of existing in multiple configurations, such as the ones mentioned here, and with a capability to switch between configurations dy-

namically without any interruption to the ongoing group communication session. In the next section we further explain our prototype collaborative planning tool and how such a capability might help survive attacks that cannot be repelled.

PROTOTYPE IMPLEMENTATION

Our prototype collaborative planning tool currently has three components: shared whiteboard, text-based chat, and voice communication. The whiteboard and chat components require reliable data delivery to all group members, while the voice component must tolerate some data loss because delay requirements make it impossible to wait for retransmissions. All three components of the tool are vulnerable to attack.

We use the Java programming language for the prototype implementation to take advantage of the high-level networking class abstractions which enable rapid development of the prototype. The entire system as implemented required less than 1300 lines of code in Java. Java also supports multi-threading and object serialization which are used in the implementation. [2]

We did not attempt to develop new communication configurations or new protocols. Instead we used the well-known configurations explained in the previous section and standard TCP and UDP protocols to build the components of our collaboration tool [5]. In the current version of the prototype, all reconfiguration is done manually. Future versions may incorporate some form of intrusion detection coupled with an intelligent controller to automate reconfiguration.

Because the whiteboard and chat components of the tool both require reliable data transfer of data, they were built to use TCP, the reliable Transmission Control Protocol. The voice component, which can tolerate some data loss, uses UDP, the unreliable User Datagram Protocol.

The collaboration tool containing these three components was implemented to follow three different configurations. The conference begins in a peer-to-peer configuration where there is full connectivity among all the nodes. The remaining members of the conference can continue uninterrupted even if one of the nodes fails due to an attack. Meanwhile suitable recovery actions can be taken at the affected node.

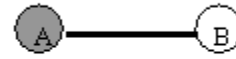


Figure 4a. : Conference begins – peer-to-peer connection between two participants

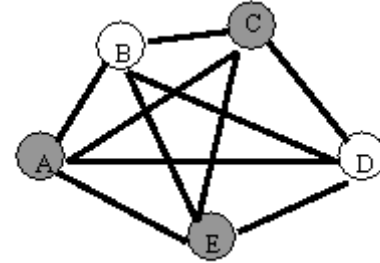


Figure 4b. Conference continues – fully inter-connected peer-to-peer connections among all participants

This is a robust configuration, but it wastes bandwidth. After the number of participants reaches a threshold, we might want to reconfigure the system from peer-to-peer to client-server. One of the participants is designated to act as a server and the others are now clients.

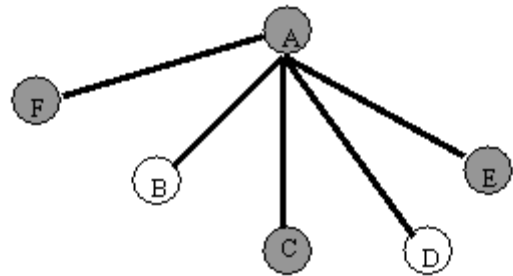


Figure 4c. As group size grows, system switches to client-server configuration to utilize bandwidth.

Bandwidth is better utilized, though this configuration still is not the best because all packets are being unicast. If the number of participants becomes very large then the system could reconfigure once again from client-server to multi-cast.

Due to the lack of Java implementations of reliable multicast protocols, only the voice component of the tool supports multicast at this time. However we are evaluating Java implementations of reliable multicast such as Light weight reliable multicast protocol [3] and the possibility of using the same in our system.

If the server is attacked, then the system fails and the conference is interrupted.

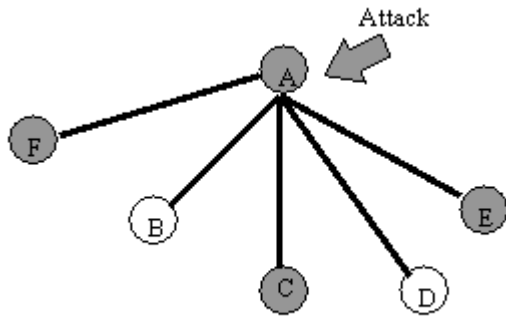


Figure 4d. When the server is attacked, clients disconnect from Server.

To survive this attack, we can dynamically reconfigure the system to go back to the fully interconnected peer-to-peer configuration. The system survives the attack and the conference continuous, though with fewer participants.

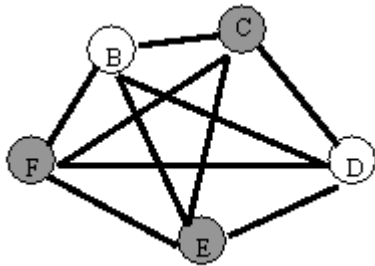


Figure 4e. : Dynamically reconfigure back to a fully interconnected peer-to-peer configuration.

If required, a new server can be selected and the system resumes client-server operation. In the mean time, the attacked node can be repaired and later rejoin the conference. Thus by switching between different configurations, the system can use dynamic reconfiguration to survive denial-of-service attacks.

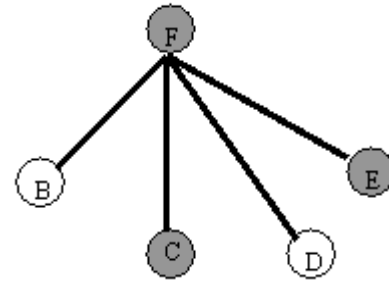


Figure 4f. : A new server is launched and the conference proceeds in a client-server configuration.

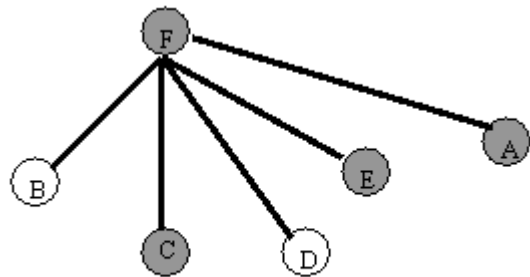


Figure 4g. : The attacked node rejoins the conference after having been repaired.

CONCLUSIONS

This research does not focus on security issues such as privacy and intrusion detection. The primary concern is to survive denial-of-service attacks. To demonstrate this concept, we implemented a prototype collaborative planning tool using the Java programming language [4]. The prototype collaborative tools are used to demonstrate and evaluate the concept of dynamic reconfiguration as a survivability strategy. In the current prototype, dynamic reconfiguration is still a manual process. Failures and attacks are not detected automatically, and user intervention is required to switch configurations.

This prototype demonstrates the capability to dynamically switch between configurations and protocols during an active group communication session. This capability to control the underlying network communication system in a manner that is transparent to the application is the first step in surviving attacks and system failures.

FUTURE WORK

The current prototype demonstrates a reconfigurable communication system. An intrusion detection component could be added on to automate the reconfiguration process. For example, the clients in a client-server configuration could use an election protocol to designate a new server when a failure is detected. Other aspects of network security, such as the use of cryptography for privacy and authentication, could be addressed to harden the system and make it more difficult, but not impossible, to launch successful attacks. However, because attacks will still be possible, the ability to dynamically reconfigure is crucial.

The collaboration tool could be enhanced by adding functionality to existing components, such as loading images in the whiteboard, or by adding new components, such as video communication. Reliability could be enhanced to provide late joiners with data that they may have missed.

The reconfigurable communication system could be enhanced by adding reliable multicast protocols to the collection of configurations. We could also take advantage of the dynamic capabilities of Java to distribute objects that represent network connections. This would make it possible to dynamically update all nodes with code that implements a new protocol. Thus, instead of switching among several pre-set configurations, the application could be dynamically updated with objects that know how to transport themselves across the network with a new protocol.

BIBLIOGRAPHY

1. Howard F. Lipson, Thomas A. Longstaff. Coming Attractions in Survivable Systems. Draft Technical Report. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213. June 1996.
<http://www.cert.org/research/start_page.html>
2. Merlin Hughes, Conrad Hughes, Michael Shoffner, Maria Winslow. *Java Network Programming*. Manning Publications Company. 1997.
3. Tie Liao. Light-weight Reliable Multicast Protocol as an Extension to RTP. Technical Report. Inria Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France. December, 1997.
<http://monet.inria.fr/lrmp/lrmp_rtp.html>
4. Ken Arnold, James Gosling. *The Java Programming Language*. Addison-Wesley. 1996.

5. Dimitri Bertsekas, Robert Gallager. *Data Networks*. Second Edition. Prentice Hall. 1991.

* The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.