The CoBFIT Toolkit

HariGovind Ramasamy
IBM Zurich Research Laboratory
Switzerland 8803
hvr@zurich.ibm.com

Mouna Seri and William H. Sanders
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{seri,whs}@crhc.uiuc.edu

Categories and Subject Descriptors: D.2.13 [Software Engineering]: Reusable Software–*Reusable Libraries*; C.2.4 [Computer-Communication Networks]: Distributed Systems–*Distributed Applications*

General Terms: Algorithms, Experimentation, Security, Theory.

Keywords: Intrusion Tolerance, Byzantine Faults, Replication, Software Toolkit.

CoBFIT

We have developed a software toolkit, called *CoBFIT*, that implements a comprehensive suite of protocols for efficient and dynamic replication of a stateful service in an asynchronous environment such as the Internet. The CoBFIT toolkit can be used to make a generic service fault-tolerant using the state machine replication approach [7], provided the service can be structured as a deterministic state machine. The toolkit is implemented in C++ using the ACE object-oriented network programming toolkit [6] and is on the verge of an open-source release.

Figure 1 shows the various components in the CoBFIT toolkit. The *framework components* form the foundation upon which *protocol components* are built. Detailed descriptions of all the CoBFIT protocol and framework components are given in [5].

The CoBFIT toolkit implements many framework components that provide a common foundation not only for the specific protocols that we have implemented, but for similar distributed fault-tolerant protocols as well. The foundation includes several services that are commonly needed for implementing distributed protocols, including event handling, network communication, management of protocol components, and cryptographic primitives (by interfacing with the Cryptlib library [1]).

The CoBFIT suite of protocols is unique in that unlike the protocols in previous group communication systems, ours do not make correctness conditional upon the ability to remove suspected members from the group. The CoBFIT protocols can be utilized individually to provide their specific properties (e.g., reliable broadcast) as well as together in concert to realize a dynamic replication group that satisfies the virtual synchrony property [4] in the asynchronous model. The toolkit includes the following asynchronous Byzantine-fault-tolerant protocols:

Broadcast Consistent broadcast (CBroadcast) and reliable broadcast (RBroadcast) [3],

Agreement Binary agreement (ABBA) and multi-valued agreement (MVBA) [3],

Replication Parsimonious protocols (PABC and APE) for the agreement and execution phases of state machine replication (respectively) that achieve optimal efficiency for certain protocol metrics of interest (e.g., message complexity and overall resource utilization) [5], and

Dynamic Group Management Group membership agreement (GMA), policy-based admission and removal of group members, and reconfiguration of replication protocols to provide virtual synchrony properties [5].

The CoBFIT toolkit provides a unified graphical user interface (GUI) for instantiating and managing a replicated service. The GUI was built using the JGraph open-source graph component [2]. The GUI serves as a central management console from which a user can define the attributes of each replica node (Figure 2) and the network topology (Figure 3), specify and alter fault tolerance requirements (Figure 4), start a replication group (Figure 5), monitor the execution of the replicas, and do fault injection experiments.

We have demonstrated the capabilities of the toolkit in the context of a toy application, namely distributed Towers of Hanoi. We have tested the protocols successfully in the presence of extensive fault injection campaigns and evaluated them in LAN and WAN (e.g., Planetlab) settings [5].

Acknowledgments

This material is based upon work supported in part by the National Science Foundation under Grant No. CNS-0406351 and by DARPA contract F30602-00-C-0172. We would like to recognize Luke St.Clair for his work in implementing the ABBA and MVBA protocols. We thank Jenny Applequist for her editorial comments.

References

[1] Cryptlib Toolkit. http://www.cs.auckland.ac.nz/~pgut001/cryptlib/.

[2] The JGraph Home Page. http://www.jgraph.com/.

[3] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. Secure and Efficient Asynchronous Broadcast Protocols. In *Advances in Cryptology: CRYPTO 2001 (J. Kilian, ed.), LNCS-2139*, pages 524–541. Springer, 2001.

[4] R. Friedman and R. van Renesse. Strong and Weak Virtual Synchrony in Horus. In *Proceedings of the 15th Symposium on Reliable Distributed Systems*, pages 140–149, Oct 1996.

[5] H. V. Ramasamy. *Parsimonious Service Replication for Tolerating Malicious Attacks in Asynchronous Environments.* PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[6] D. Schmidt and S. Huston. *C++ Network Programming: Systematic Reuse with ACE and Frameworks.* Addison-Wesley Longman, 2003.

[7] F. B. Schneider. Implementing Fault-Tolerant Services using the State Machine Approach: A Tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990.

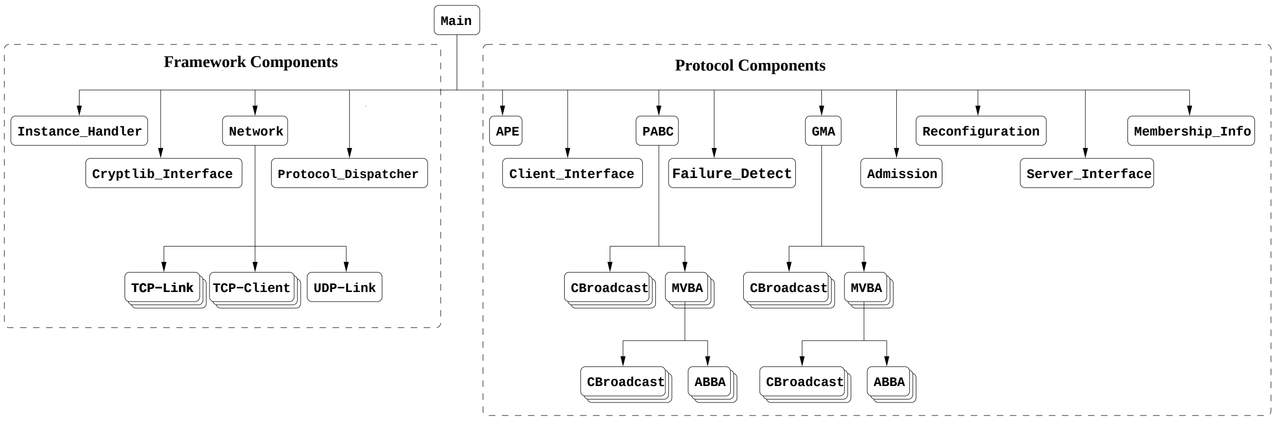
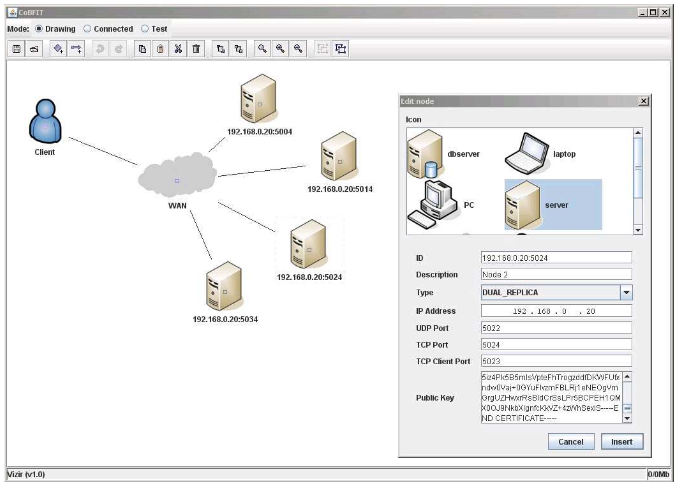


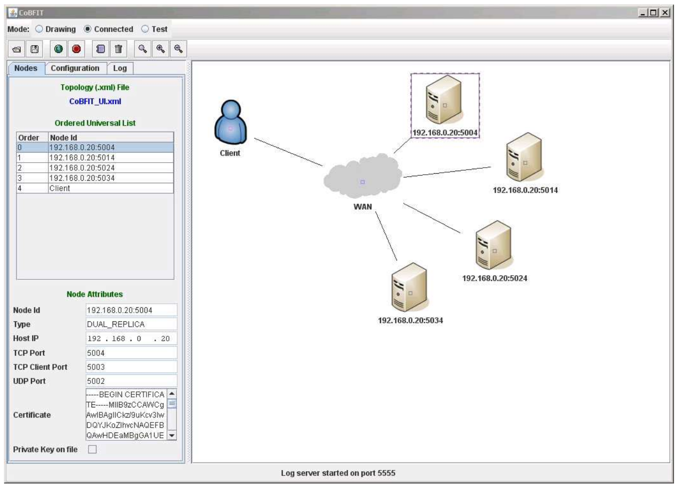
Figure 1: Component Instantiation Hierarchy



Mode: [✓ Drawing] Connected Test

ID	192.168.0.20:5024
Description	Node 2
Type	DUAL_REPLICA
IP Address	192 . 168 . 0 . 20
UDP Port	5022
TCP Port	5024
TCP Client Port	5023
Public Key	5iz4Pk5B5mIsVpteFhTrogzddfDKWFUfx ndw0Vaj+0GYuFlvzmFBLRj1eNEOgVm GrgUZHwxrRsBldCrSsLPr5BCPEH1QM X0OJ9NkbXignfcKkVZ+4zWhSexiS-----END CERTIFICATE-----

Figure 2: GUI for Editing Node Attributes



Mode: Drawing [✓ Connected] Test

Topology (.xml) File
CoBFIT_UI.xml

Ordered Universal List

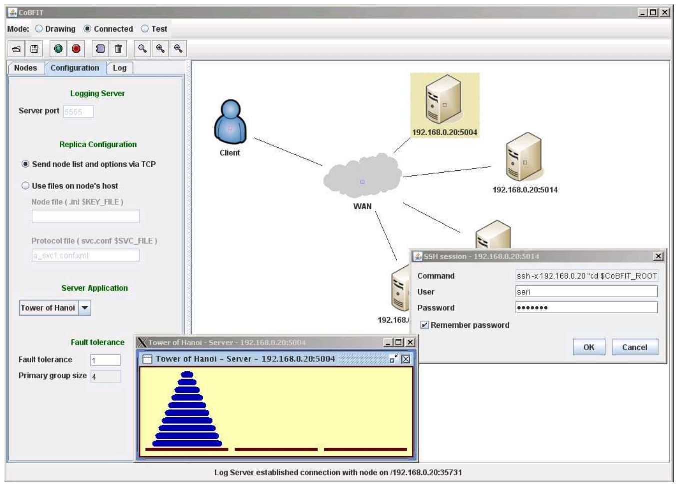
Order	Node Id
0	192.168.0.20:5004
1	192.168.0.20:5014
2	192.168.0.20:5024
3	192.168.0.20:5034
4	Client

Node Attributes

Node Id	192.168.0.20:5004
Type	DUAL_REPLICA
Host IP	192 . 168 . 0 . 20
TCP Port	5004
TCP Client Port	5003
UDP Port	5002
Certificate	-----BEGIN CERTIFICATE-----MIIB9zCCAWCg AwIBAgIICkz/9uKcv3lw DQYJKoZIhvcNAQEFB QAwHDEaMBgGA1UE
Private Key on file	☐

Log server started on port 5555

Figure 3: GUI for Editing Network Topology


CoBFIT
Nodes
Configuration
Log
Client
WAN
192.168.0.20:5004
192.168.0.20:5014
192.168.
Tower of Hanoi - Server - 192.168.0.20:5004
Tower of Hanoi - Server - 192.168.0.20:5004
SSH session - 192.168.0.20:5014
OK
Cancel

Mode: Drawing [✓ Connected] Test

Logging Server
Server port 5555

Replica Configuration
[✓ Send node list and options via TCP]
Use files on node's host
Node file (.ini $KEY_FILE)
Protocol file (svc.conf $SVC_FILE) a_svc1.conf.xml

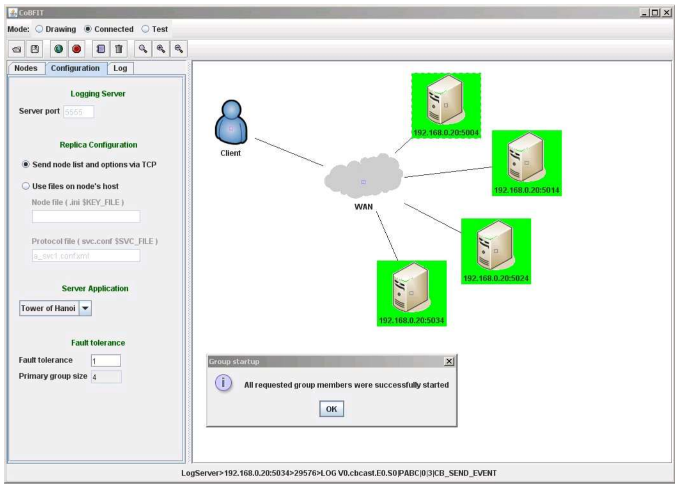
Server Application
Tower of Hanoi

Fault tolerance
Fault tolerance 1
Primary group size 4

Command ssh -x 192.168.0.20 "cd $CoBFIT_ROOT
User seri
Password •••••••
[✓] Remember password

Log Server established connection with node on /192.168.0.20:35731

Figure 4: GUI for Instantiating a Replication Group



Mode: Drawing [✓ Connected] Test

Logging Server
Server port 5555

Replica Configuration
[✓ Send node list and options via TCP]
Use files on node's host
Node file (.ini $KEY_FILE)
Protocol file (svc.conf $SVC_FILE) a_svc1.conf.xml

Server Application
Tower of Hanoi

Fault tolerance
Fault tolerance 1
Primary group size 4

All requested group members were successfully started

LogServer>192.168.0.20:5034>29576>LOG V0.cbcast.E0.S0|PABC|0|3|CB_SEND_EVENT

Figure 5: GUI for Monitoring, Testing, and Fault-Injecting a Replication Group