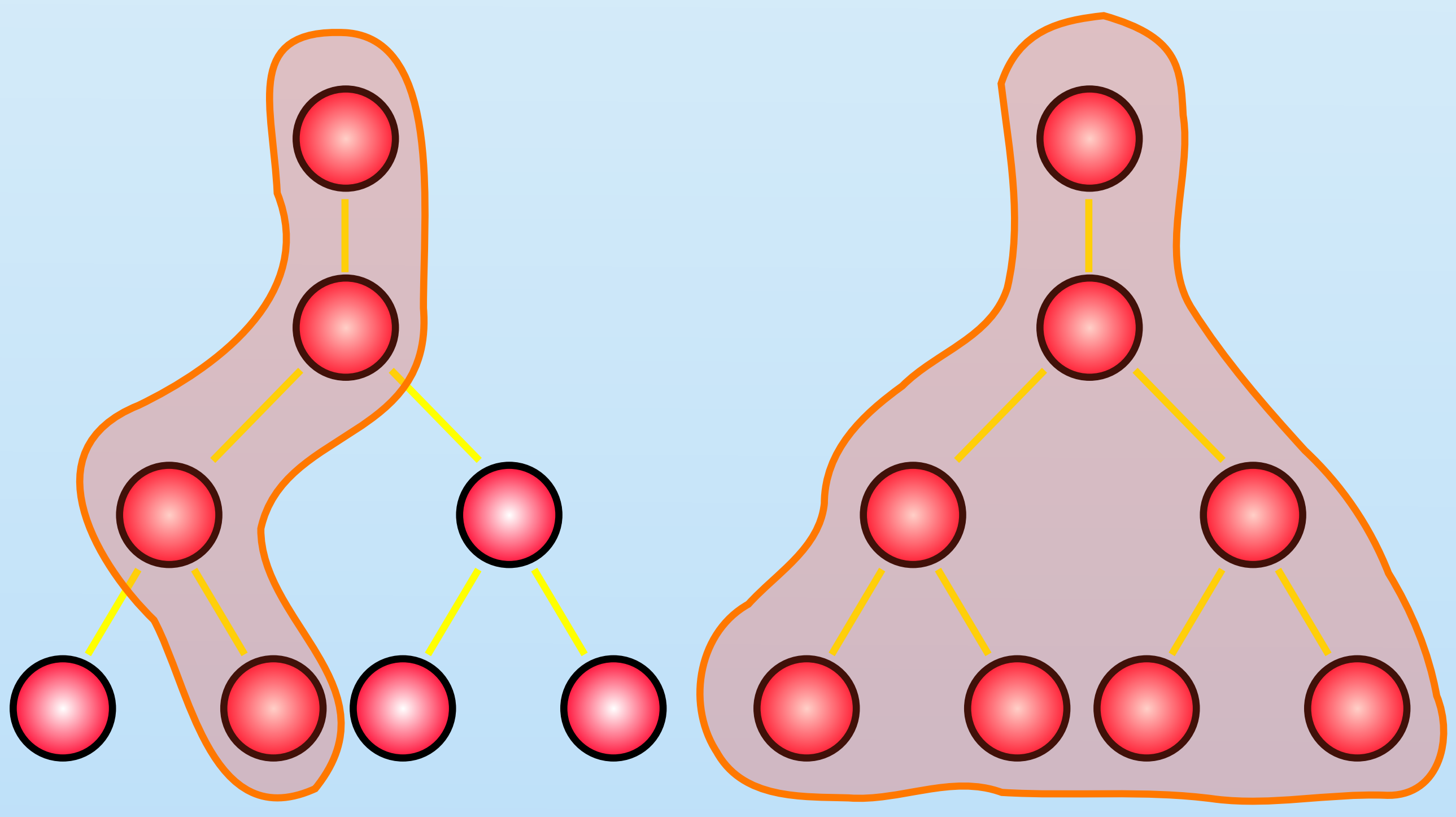
Verifying Networked Programs Using a Model Checker Extension

Abstract

Model checking finds failures in software by exploring every possible execution schedule. Until recently it has been mainly applied on stand-alone applications. We propose an extension for a Java model checker to support networked programs. It contains a cache module, which captures data streams between a target process and a peer process. Captured data are replayed by the cache module when a duplicate request is sent. This demonstration shows how we found a defect in a WebDAV client with a model checker and our extension.

Background

Software model checking verifies software by exploring every possible schedule whereas software testing only executes the program through one thread schedule for each run. Java PathFinder (JPF), a model checker for Java, is used as a base model checker for our development. It includes its own Java Virtual Machine, which explores all thread schedules of the program. Although it is designed to verify only a single process at a time, we extend its functionalities via several mechanisms to support multi-process networked applications.

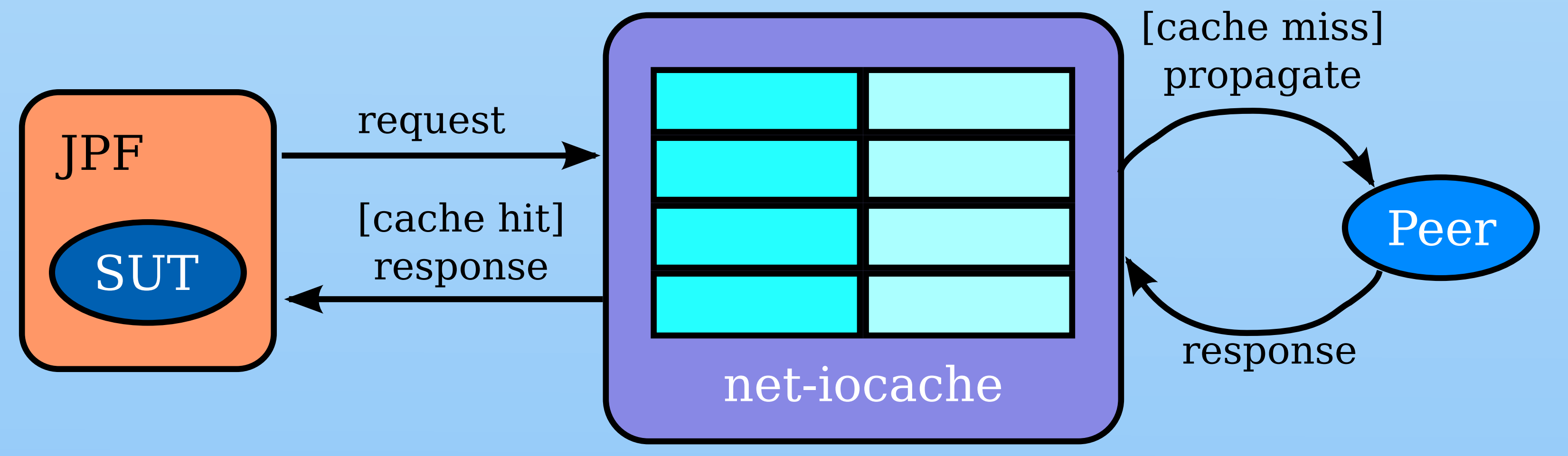


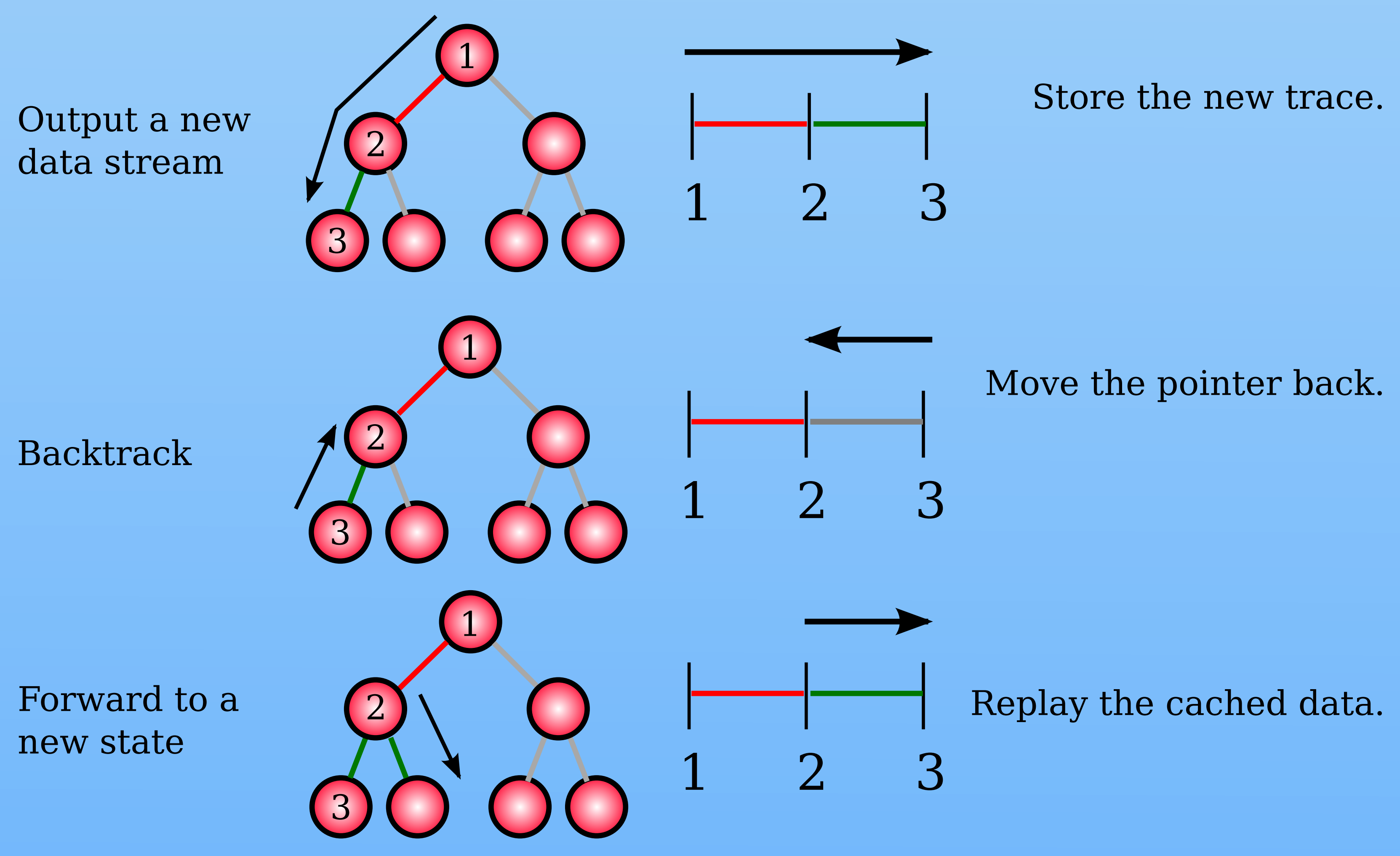
Testing covers only one of the possible schedules for each run.

Model checking executes and covers all possible schedules.

Concept of Cache

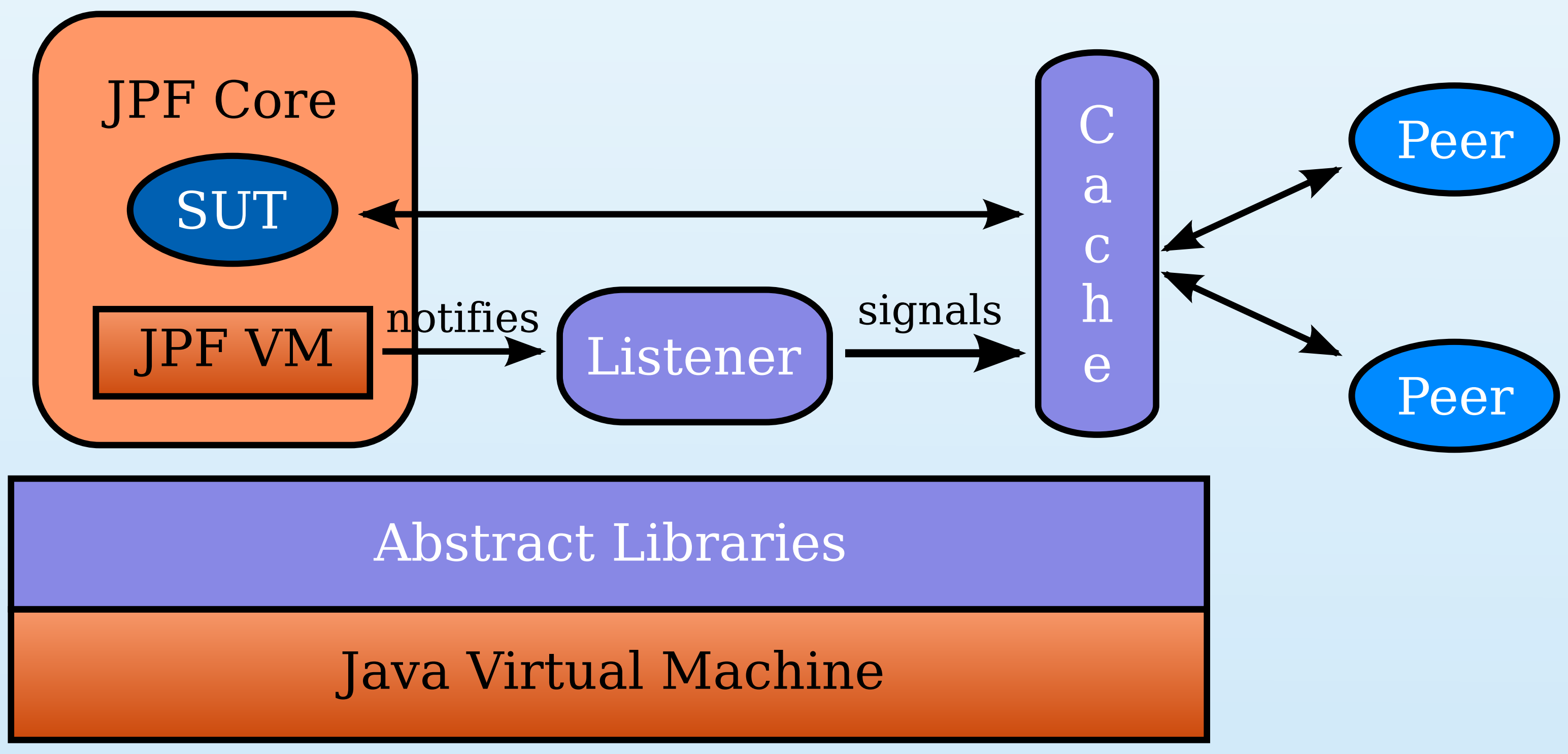
Cache can be used as a proxy to the real external process. *Net-iocache*, our JPF extension, makes use of requests and responses in the past and sends already known responses back to the target application instead of dispatching request messages to the peer process. As a result, peer processes do not become aware of the target application being driven by the model checker. If the request is not cached, the I/O-cache will physically send the request to the peer, wait for a response, and remember it.





Architecture

* Implemented as a JPF extension.
* Network-related classes are rewritten as abstract classes.
* Abstract output stream redirects outputs of the SUT to the cache.
* Abstract input stream accepts iutputs from the cache.
* Listener signals the cache on the state transition event.
* Cache saves/restores pointer positions and the number of active connections.



Experiments

* Platform
    * 8-core Mac Pro workstation
    * 16GB physical memory
    * Ubuntu 8.04 and JPF 4 (revision 1109)
* Tested programs
    * Alphabet server/client
    * HTTP server/client
    * Multipart file downloading tool
* Partial results
    * Alphabet client: 4 threads/2 characters, ~33 min/1.6M states
    * Alphabet server: 7 threads/1 character, ~50 min/2.2M states

Error Trace of a WebDAV Client

```
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
NullPointerException: calling 'hyber()V' on null object
at StreamDemultiplexor.init(StreamDemultiplexor.java)
(stack trace omitted)
transition #3 thread: 0
ThreadChoiceFromSet {>main,SocketTimeout}
StreamDemultiplexor : timer = t.setTimeout();
(some transitions omitted)
transition #153 thread: 1
ThreadChoiceFromSet {main,>SocketTimeout}
StreamDemultiplexor.java:435 : timer = null;
StreamDemultiplexor.java:438 : demuxList.remove(this);
transition #154 thread: 0
ThreadChoiceFromSet {>main,SocketTimeout}
StreamDemultiplexor.java:138 : timer.hyber();
HTTPConnection.java:2268 : requestList.remove(req);
(some transitions omitted)
snapshot #1
thread index=1,name=SocketTimeout,status=RUNNING
call stack:
at StreamDemultiplexor.close(StreamDemultiplexor)
at StreamDemultiplexor.markForClose(StreamDemultiplexor)
at SocketTimeout.run(StreamDemultiplexor)
----------------------------------- search finished
```

NullPointerException Call a method on a null object.

timer is assigned.

timer is null.

Call method on null object.

Call stack

The client starts with two threads, *main* thread and *SocketTimeout* thread. Any inactive stream will be automatically closed by the timeout thread after a certain period. In transition #3, the main thread executes method init and starts counting time by method *setTimeout*. In the error scenario, before method *hyber* is called, thread SocketTimeout gets its turn and continues running until time runs out. The countdown thread closes the corresponding stream and socket, making variable timer become null (transition #153). This causes hyber method call on timer to fail at transition #154.

Tool Download

* Java PathFinder subversion repository
* *https://javapathfinder.svn.sourceforge.net/svnroot/javapathfinder/trunk*

References

[1] C. Artho, W. Leungwattanakit, M. Hagiya, and Y. Tanabe. Efficient model checking of networked applications. In Proc. TOOLS EUROPE 2008, volume 19 of LNBIP, pages 22–40, Zurich, Switzerland, 2008. Springer.
[2] C. Artho, W. Leungwattanakit, M. Hagiya, and Y. Tanabe. Tools and techniques for model checking networked programs. In Proc. SNPD 2008, Phuket, Thailand, 2008. IEEE.
[3] E. Clarke, O. Grumberg, and D. Peled. Model Checking. MIT Press, 1999.
[4] Y. Y. Goland. Webdav: A network protocol for remote collaborative authoring on the web. In Proc. of the Sixth European Conf. on Computer Supported Cooperative Work, pages 12–16, 1999.
[5] B. Holmes. A Simple PROPFIND/PROPPATCH Client, 2000.
[6] K. Havelund and T. Pressburger. Model checking Java programs using Java PathFinder. International Journal on Software Tools for Technology Transfer, 2(4):366–381, 2000.
[7] NASA Ames Research Center. Java PathFinder 4 documentation, 2006.

University of Tokyo Watcharin Leungwattanakit, Masami Hagiya, Yoshinori Tanabe
RCIS/AIST Cyrille Artho **Chiba University** Mitsuharu Yamamoto