Network Simulation Tools for Prototyping Scalable P2P Applications

NATO IST RTG-12 WORKSHOP

Dr Ian Taylor

Lecturer & Coordinator, Triana-Grid Group, Cardiff University, UK
Contractor, Ball Aerospace
On-going work with SRSS group (Brian Adamson & Rick Jones)
Navy Research Lab, Washington DC
# Network Simulation Tools for Prototyping Scalable P2P Applications

## 1. REPORT DATE
01 DEC 2007

## 2. REPORT TYPE
N/A

## 3. DATES COVERED

## 4. TITLE AND SUBTITLE
Network Simulation Tools for Prototyping Scalable P2P Applications

## 6. AUTHOR(S)

## 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Cardiff University, UK

## 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

## 11. SPONSOR/MONITOR'S ACRONYM(S)

## 12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

## 15. SUBJECT TERMS

## 16. SECURITY CLASSIFICATION OF:

<table>
<thead>
<tr>
<th>a. REPORT</th>
<th>b. ABSTRACT</th>
<th>c. THIS PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>unclassified</td>
<td>unclassified</td>
<td>unclassified</td>
</tr>
</tbody>
</table>

## 17. LIMITATION OF ABSTRACT
UU

## 18. NUMBER OF PAGES
28

--

Standard Form 298 (Rev. 8-98)
Prepared by ANSI Std Z39.18
Coalescing Binary Search
SRSS Project

Stands for ...
Scalable Robust Self-organizing Sensors ...

- Simulate mobile sensor networks using NS-2
- Investigate Publish/subscribe/P2P discovery mechanisms e.g. Unicast, Multicast etc
- Conduct simulations to trade-off performance of application-level and/or network level discovery mechanisms in sensor environment.
- Running within MANET networks - wireless connectivity
- For NS-2, this involves:
  - Integrating data transmission between NS-2 nodes (via Protolib)
  - Building infrastructure to allow different middleware to be easily integrated into this architecture.
Overview of SRSS Architecture

Application - dynamic self organizing sensor nets

P2P Middleware (discovery, communication mechanisms)

Protolib Application Interface (PAI)

Protolib (NRL)

Network Wireless (or wired..) | NS-2 MANET simulations
The SRSS Environment

What is MANET?
- Mobile Ad hoc Networks
- Wireless transmission
- No centralized administration / control
- No existing network infrastructure
- A node can be a source, a sink or a transit
- All nodes participate in the discovery of a route

Applications
- Cooperative mobile data exchange
- Rapidly deployable communication with efficient networking
- Communication where no infrastructure exists

and the sensors

- Flexible for experimentation, but demonstration-worthy form factor.
- Linux on PC-104 or similar platform is a likely candidate.
NS-2: A Network Simulator

- Discrete event simulator
- Packet-level
- Link layer and up
  - i.e. network, transport, session, presentation and application
- Wired and wireless simulations
- Platforms
  - Most UNIX and UNIX-like systems
  - Window 95/98/NT
  - (Emulation only for FreeBSD for now)
NS-2: OTcl and C++

Agents/communication
- Pure C++ objects

Simulations/scenarios
- Pure OTcl objects

C++/OTcl split objects

C++

OTcl

ns

UNCLASSIFIED - APPROVED FOR PUBLIC RELEASE
NAM - Example

- Three TCP slow-start restart algorithms
- Test - improving restart of idle TCP connections
Protocol Prototyping library (ProtoLib) - C++ class library

Cross-platform - works on Windows and Unix using native implementations

Provides networking capabilities -
- Currently supports UDP communication
  - Unicast and Multicast
- Communication works across networks or between NS-2 nodes, by:
  - overriding basic NS-2 UDP protocol implementation
  - can communicate data across NS2 nodes
  - Can simulate real networked applications passing real data
  - We are doing this for the P2P world ..

UNCLASSIFIED - APPROVED FOR PUBLIC RELEASE
Protolib Overview

Application

Protolib

Communication

UDP

TCP*

Timers

Protocol Timer

Event Dispatcher

Event Notifications

Underlying Fabric

Network

NS-2

(*) Will be implemented next phase

UNCLASSIFIED - APPROVED FOR PUBLIC RELEASE
Simple Protolib Scenario

- 100 Millisecond Timer
- Event Dispatcher
- UDP Send
- Protocol
- Network
- NS-2
- UDP Receive
- Event Dispatcher
- Application
- Protolib
- Callback Functions

UNCLASSIFIED - Approved for Public Release
Protolib Application Interface (PAI)

- Abstracts reliance on specific networking/timing mechanisms in Protolib/others
  - Middleware/Applications use PAI and change environment to choose configuration e.g. Network or NS-2 etc
  - Provides generic classes for creating sockets/timers
  - Support multiple sockets/timers + listeners e.g. for timeouts or UDP receive data events - multithreaded event dispatching
  - Provides a concise C++ interface for Java JNI integration

![Diagram](https://example.com/diagram.png)
PAI Structure, Factory Method Design

Notifications e.g. Socket Received Data, Timer time outs

Application

PAI Interface

PAI Engine

PAISocket
PAITimer
PAIDispatcher

UDP Socket
NS2 UDP Socket
Timer
NS2 Timer
Event Disp.
NS2 Event Disp.

PAI Environment

PAI Factory

Calls e.g. send/receive

UNCLASSIFIED - APPROVED FOR PUBLIC RELEASE
**PAI Example**

**Example Main Program:**

```cpp
pai.getEnvironment()->setBinding(PAI_NETWORK);
pai.getEnvironment()->setNetworkProtocol(PAI_UDP);

timer = pti->addTimer(1.0, 5);
sock = pci->addSocket(5004);

pci->addListener(sock, this, (CallbackFunc)&PAI_Example::OnTxTimeout);
pti->addListener(timer, this, (CallbackFunc)&PAI_Example::OnSocketRecv);
pti->runTimers();
```

**When Timer times out:**

```cpp
void PAI_Example::OnTxTimeout() {
    .....  
    pci->send(sock1, "127.0.0.1", buffer, len);
}
```

**When Data is Received:**

```cpp
void PAI_Example::OnSocketRecv() {
    .....  
    char *buf = pci->recv(sock1, &addr, &len);
}
```
P2P Middleware

P2P Middleware Requirements?
- Dynamic Discovery Mechanisms e.g. Unicast, Multicast
- Communication - support different transport protocols UDP, TCP etc
- Lightweight

P2P: Which middleware to use?
- First promising choice: JXTA - Summer 2002
- Problems:
  - Lightweight? Only by limiting functionality on Edge peers
  - Scalability problems - discovering multiple pipes unreliable in tests
  - Difficult to extend code base
    - JXTA uses Endpoint Implementations to represent different network communication protocols e.g. TCP, Bluetooth etc BUT:
    - Different implementation for discovery - complex to plug in new layer

UNCLASSIFIED - APPROVED FOR PUBLIC RELEASE
Gridlab GAT Architecture

Application (e.g. Triana)

GridLab GAT (www.gridlab.org)

Adapter
OGSA Service

Adapter
Web Service

GAP Adapter
JXTA
P2PS

GAP is our P2P Application-level Interface
Triana and the GAP Interface

- Interface between Application and Peer-to-Peer Middleware
  - Provides an insulation layer for P2P applications
  - loosely coupled, dynamically late bound modules

- Simple
- Generic
  - Not Triana Specific
  - Contains common calls e.g. advertise_service, discover_service, create_pipe etc

For more info on Triana Distributed Implementation, see www.trianacode.org
Lightweight P2P Middleware:

- Language independent specification -
  - reference implementation is in Java - C++ version in planning
  - Communication is language independent - use XML adverts and data structures

- Pluggable transport layer - currently implemented UDP, TCP/IP
  - Dynamic Discovery - using Unicast and Multicast
  - Factory design, using resolvers

- Decentralized structure

- Uses Rendezvous nodes (self nominated) for caching adverts/data
  - (centralized-decentralized) network structure for scalability

- Implements Relays - traversing firewalls

- 1/100th size of JXTA ...

- Release (www.trianacode.org) soon .. for open source development
Discovery Service
- Broadcast/locate adverts + queries in discovery subnet

Pipe Service
- Connect pipes using endpoint resolvers

Rendezvous Service
- Send/receive queries from known rendezvous peers
Discovery Service

- All peers have Discovery Services
- Caches local adverts + queries
- Broadcast adverts + queries to all peers in its discovery subnet
- Responds to received adverts + queries
- Discovery subnet scope determined by resolvers
  - e.g. UDP Multicast scope
Discovery Service Scenario

1. Peer A creates pipe
   - Broadcasts advert
   - Caches advert locally (not cached at Peer B)
2. Peer B queries for all pipes
   - Broadcasts query
3. Peer A receives query
   - Matches query with locally cached pipe advert
   - Sends pipe advert direct to Peer B
4. Peer B discovers pipe
Peers can optionally become rendezvous peers
- Direct pipe connections to other rendezvous peers
  - usually in other discovery subnets
- Cache all received adverts + queries
- Forward queries to known rendezvous peers
  - Note: Adverts are not forwarded outside discovery subnet
NS-2 GAP Integration

SRSS Application

C++ Interface (JNI)

GAP Layer

JXTA P2P

Web Serv.

P2PS

P2PS Comms

Network

JNI

PAI

Protolib

Network

UNCLASSIFIED - APPROVED FOR PUBLIC RELEASE
Status Of Implementation

- Application - dynamic self organizing sensor nets
- JNI
- GAP with P2PS Binding
- Protolib Application Interface (PAI)
- Protolib (NRL)
- Network
- NS-2

UNCLASSIFIED - APPROVED FOR PUBLIC RELEASE
Scenarios and Conclusions

Resulting system has many applications/uses:

- Mobile Sensors - test discovery in simulated dynamically changing environments.
- Triana - simulate P2P environment - see if P2P middleware actually scales without having to run Triana on 1000's of nodes ...

The new GAT/GAP EU proposal - The P2P Gap interface will be generalized further in Gridlab2 with input from applications - NRL, GW@Home (AIP) and ..