JXTA-Sim2: A Simulator for the core JXTA protocols

by

Paul Dolan, B.Sc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

September 2010

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Paul Dolan

September 13, 2010

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Paul Dolan

September 13, 2010

Acknowledgments

My sincere thanks are extended to my family and friends, all of who provided much required support throughout the development of this project.

My thanks are also extended to my supervisor, Dr. René Meier, for his guidance both in this project and beyond.

Paul Dolan

University of Dublin, Trinity College
September 2010

JXTA-Sim2: A Simulator for the core JXTA protocols

Paul Dolan, M.Sc.
University of Dublin, Trinity College, 2010

Supervisor: Dr. René Meier

Peer-to-Peer networks are becoming increasingly popular for providing distributed solutions for industry, acadamia and the individual with solutions being developed for a wide range of applications - filesharing, distributed computation, content delivery, VoIP etc.

JXTA, released in 2001, offers a suite of protocols addressing a number of the key issues related to designing such networks - addressing, routing, scalability, endpoint communication.

However, while JXTA has had reasonable uptake within industry and academia, many aspects of its behaviour remain unknown - particularly with regards to large deployments containing hundreds and thousands of peers.

One such project was JXTA-Sim, a simulator for the evaluation of the JXTA lookup algorithm. Built on top of a P2P simulation framework, JXTA-Sim allows researchers a means to quickly see how networks with large numbers of peers react using the JXTA

hybrid search algorithm for advertisement discovery and how these algorithms reacts to varying some core underlying parameters.

This project presents JXTA-Sim2, an extension of the functionality provided by JXTA-Sim to support all network node types, build a more realistic network for simulation and implement the JXTA protocol for the generation of a distributed routing table.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Originally conceived by Sun Microsystems in 2001, JXTA was developed in order to allow for the rapid and relatively easy development of scalable and robust P2P services by abstracting the majority of low level P2P functionality.

While JXTA is currently used extensively in both academic and industrial applications, many aspects of its behavior, especially with regards to scalability, remain largely unknown.

JXTA-Sim was developed in order to begin addressing these knowledge gaps. JXTA-Sim allows for the simulation of the JXTA Lookup Algorithm, implementing two of the six protocols provided as part of the JXTA suite - Rendezvous and Peer Discovery.

Although version 1 of JXTA-Sim succeeded in its goals of providing a relatively flexible simulation platform on which to evaluate a single aspect of the JXTA protocol suite, there is still much scope remaining for further development under the following headings:

Improvements to simulation While the version of the rendezvous protocol has been implemented as part of JXTA-Sim, the algorithm utilised does not accurately re-

produce all aspects of the protocol. Specifically, edge peers do not periodically probe rendezvous (directly or via a relay) in order to verify activity and accessibility in order to react to adverse network conditions related to a high churn rate. Currently the workaround provided as part of JXTA-Sim is for rendezvous peers leaving the network to migrate their associated edge peers to another active rendezvous. This behavior is not analogous to a real-world situation where a failing node may not be able to preform a handoff before losing network connectivity.

Addition of real-world simulation parameters Further improvements to the realism of a simulation could be added. Latency, retransmissions caused by the introduction of errors or the introduction of local NATing firewalls would affect a simulations results, pushing towards a more meaningful real-world analogy.

Complete JXTA protocol support JXTA-Sim version 1 only supports protocols for Discovery and Routing. The addition of all protocols within the JXTA-Suite to JXTA-Sim would provide a complete evaluation tool allowing for more complex simulations involving all aspects of potential system interaction.

JXTA Application Simulation With complete protocol support as described above, it would become possible to investigate application level simulations utilising one or all of the core JXTA protocols.

1.2 Project Aims

The aims of this project fall into two broad categories:

Research Building on [1], additional analysis will be preformed on the protocols available within the JXTA suite. Distributed algorithms utilised will be identified and documented.

Development Additional functionality will be provided for the JXTA-Sim application based on the results from the above research. These additions will come under two headings mentioned in Section 1.1 - Improvements to Simulation and Additional JXTA Protocol Support.

1.3 Project Contributions

Again, closely following the methodology adopted by [1] the foreseeable steps will involve the following steps:

1. Identification of JXTA algorithms and the protocols implementing these algorithms
2. Documentation of protocols utilised in the previous step
3. Architecture of software components to extend functionality of JXTA-Sim in order to implement both a subset of the protocols/algorithms identified in the previous steps and offer additional simulation parameters.
4. Simulation and Evaluation using Version 2 of JXTA-Sim as a tool to draw further conclusions on the Discovery Protocol simulation component specified in [1] with additional simulation parameters and an additional distributed algorithm identified in the previous steps.

1.4 Dissertation Outline

This research project is organised as follows:

Chapter 2 provides a background on the JXTA protocol and an introduction to key concepts relating to the generation of the JXTA overlay including addressing, peer

types and advertisements. A distinction is provided between the two main protocol classes in JXTA and a description of each protocol.

Chapter 3 presents some background on the simulating peer-to-peer protocols in general and the tools used for preforming such simulations. Both PlanetSim and JXTA-Sim1 are introduced.

Chapter 4 offers an overview of the functionality, architecture and functionality provided by JXTA-Sim1 including some notes on running a simulation and the associated outputs.

Chapter 5 is concerned with JXTA-Sim2 architecture, code constructs such as nodes and messages and the protocols implemented.

Chapter 6 evaluates JXTA-Sim2 and the results which may be gained.

Chapter 7 offers a summary of the work completed in this project and potential avenues of future research which may be related.

Chapter 2

Background

2.1 Introduction to JXTA

JXTA is a suite of protocol and message definitions designed to allow nodes within disparate physical networks a means to creation and communicate via an ad-hoc, peer-to-peer network overlay. The definitions laid out by JXTA provide a standardised mechanism for such nodes to discover each other, organise themselves into logical groups and discover and advertise services. In doing so, node to node communications can be achieved without regard to the underlying connectivity available to each node. The JXTA definition does not require specific use of any particular type of programming language, operating system or network transport protocol.

Broadly, the JXTA protocols are organised into two categories - *Core* and *Standard* Services. The *Core* service protocols are required for any device to be considered JXTA compliant and cover basic concepts relating to discovery of 'advertisements' (discussed further in Section 2.2.4) and endpoint routing. *Standard* service protocols offer a more "high level" abstraction to developers and are normally reserved for devices with higher available resources. Such services revolve around the various mechanisms to create 'pipes' (again, discussed further in Section 2.2.5), service discovery, the ability

to query information associated with a particular peer and the dissemination of network related meta-information throughout the overlay.

Again, it is worth mentioning that only the *Core* protocols need be implemented for a node to have the capability to join a JXTA network overly.

2.2 JXTA Concepts

This section provides a high-level overview of the JXTA specification.

2.2.1 Addressing

All objects within the JXTA overlay - peers, peer groups, pipes etc. need to be assigned a globally unique identifier for direct reference. The JXTA specification does not directly indicate a specific length format for IDs, however the main reference implementations indexed from the JXTA portal [2] utilise a 128bit addressing scheme.

Within JXTA, IDs are normally presented as URNs of the **jxta** namespace in the form:

urn:jxta:uuid-1234567890ABCDEF1234567890ABCDEF

2.2.2 Peer types

Within JXTA peers are divided into three categories. It is important to note that within a JXTA network, certain nodes may take on multiple peer roles.

Edge Peers

Edge peers are the outermost leaf nodes on a JXTA network graph. Are usually transient clients to the JXTA overlay with a potentially high churn rate. Edge peers

connect to the JXTA network directly (by querying a Rendezvous peers for neighboring peers), via a Relay Peer or a combination of both.

Edge peers do not strictly participate in the P2P nature of the JXTA overlay, but simply act as consumers or endpoints to application layer traffic - edge peers publish their presence and may query the overlay for remote nodes or services, but do not forward system traffic throughout the network.

Rendezvous Peers

Providing a superset of the functionality provided for by the edge peer, Rendezvous nodes are the backbone of a JXTA overlay.

Rendezvous peers coordinate communications between all nodes within the JXTA overlay by facilitating the propagation of discovery requests and advertisements within a particular peer group.

A system-wide view of all Rendezvous peers is built by the JXTA Overlay with the dissemination of each local Rendezvous peer's *Rendezvous Peer View* - an ordered list of all Rendezvous peers currently associated with each node.

All Edge and Relay peers are connected to a single Rendezvous peer at any time with all queries for discovery being forwarded to this Rendezvous. If a particular Rendezvous peer loses connectivity to the JXTA overlay, all children associated with this Rendezvous must located a new parent or promote themselves to undertake the required roles provided by the failed Rendezvous.

Relay Peers

Relay peers service two broad functions within the JXTA network.

Firstly, Relay peers allow Edge and Rendezvous peers located behind firewalls or NAT gateways to participate within the JXTA overlay. This is achieved with each peer requesting a lease for communications from a Relay peer which then acts as a message

router, receiving and responding as a proxy on behalf of the leasing peer.

Secondly, Relay peers provide a mechanism to store messages destined for an Edge or Rendezvous peers that has temporarily lost connectivity to the JXTA overlay or is otherwise unavailable.

2.2.3 PeerGroups

Peergroups provide a mechanism to limit scope and define security policies for message propagation and membership within a logical clustering of peers within the overall JXTA overlay.

Every JXTA peer is a member of the *NetPeerGroup* by default (allowing for non application specific message passing to allow the core JXTA protocols to function). Beyond that, peers usually organise themselves into a peergroup denoting a certain application context - file sharing, distributed computation etc.

Each peer may be a member of a number of distinct PeerGroups at any time and may also fulfill a separate role within each peergroup (Rendezvous, Edge or Relay).

2.2.4 Advertisements

All resources within the JXTA overlay, be it peers, peergroups, pipes, services etc. are represented by advertisements. Advertisements are represented by an XML document, following a certain pre-determined schema (based on the type of advertisement being represented).

All advertisements share a number of important attributes, mostly importantly *URN*, *lifetime* and *expiration*.

As discussed above in section 2.2.1, a URN is used to uniquely identify the resource being referenced. The *Lifetime* attribute is to define the length of time this advertisement is to exist within the network (as it is propagated from peer to peer) while the *expiration* specifies the duration in which this advertisement may be cached at a

remote peer. These attributes are analogous to DNS lifespan values - "expire" within a zones SOA refers to the age a zone may exist while "TTL" determines the length of time a cache may be deemed "fresh".

An example of a Peer Advertisement is shown in Listing 2.1.

Listing 2.1: Example Peer Advertisement

```
1  <?xml version=" 1 . 0 " encoding="UTF 8"?>
2  <!DOCTYPE jxta:PA>
3  <jxta:PA xml:space="default" xmlns : jxta="http://jxta.org">
4    <PID>
5      urn:jxta:uuid-1234567890ABCDEF1234567890ABCDEF
6    </PID>
7    <GID>
8      urn:jxta:uuid-ABCDEF1234567890ABCDEF1234567890
9    </GID>
10   <Name>
11     Test Node 1
12   </Name>
13   <Desc>
14     This is a test Node
15   </Desc>
16 </jxta:PA>
```

2.2.5 Pipes

Pipes are an abstraction used for inter-peer communications allowing for the creation of a virtual communication channel between endpoints (irrespective of the physical path between said endpoints or the network protocol being utilised over this physical path).

Due to the volatile nature of the JXTA overlay (and P2P networks in general) pipes offer an unreliable, asynchronous means of communication.

2.3 JXTA Protocols

This section provides a high-level overview of the 6 main protocols that make up the JXTA protocol suite.

2.3.1 Core Protocols

The minimal set of protocols a node must implement before being allowed to join the JXTA overlay as an edge peer.

Peer Resolver Protocol (PRP)

The *Peer Resolver Protocol* provides each peer within the JXTA overlay a basic query/response interface with which to query for peers within a particular peergroup.

A particular node will issue a uniquely identified query message containing this peers source ID, addressed to a specific named "handler" . This message will then be passed onto a particular "peergroup" of which the source node is a member. A response message will be generated on one or more peers within this peergroup dependent on the particular handler in question being implemented on the remote node.

While not a requirement for a JXTA peer, as part of the PRP peers may participate in the Shared Resource Distributed Index (SRDI). SRDI provides a mechanism to distributed an index of available resources throughout participating peers in order to reduce lookup load on Rendezvous peers by utilising the Peer Discovery Protocol.

Endpoint Routing Protocol (ERP)

Due to the transient nature of peer membership within the JXTA overlay, message routing is nondeterministic - with the potential of peers leaving and joining the network frequently. Peers that are not physically connected to the JXTA network may only have a uni-directional route for communications via a firewall or NAT gateway. Such

scenarios are handled with the use of a Relay peer (covered in Section 2.2.2) and the ERP.

A peer can thus build a route to an endpoint by making use of a local cache of routes coupled with the ability to query this peers registered peer router (i.e. a Relay peer).

The ERP will be covered in more detail in Section 5.4.

2.3.2 Standard Protocols

While the Core Protocol specification defines required behaviors for all JXTA peers, the Standard Service protocols provide greater interoperability and broader functionality. Large scale distributed functionality may only be achieved with multiple Rendezvous peers implementing the Rendezvous Protocol with advertisements being disseminated via the Peer Discovery Protocol.

An overview of the Standard Service protocols is given below:

Peer Discovery Protocol (PDP)

The PDP is used for discovering and publishing all available network resources in the form of advertisements. Resources may be peers, peergroups, pipes etc.

Using the resolver service (implemented with the PRP and the SRDI), advertisements can be queried and collated by a peers local discovery service. This is achieved with the use of a local cache for advertisements and a two tiered lookup system. Discovery is first attempted via a peers local cache (built using the SRDI mechanism) and then remotely via the PRP.

Peer Information Protocol (PIP)

The PIP is a simple protocol allowing for remote lookup of certain peer status information. Analogous to an SNMP lookup, lookups are preformed via a simple Request/Re-

sponse process.

PIP messages can be viewed as the addition of a payload to messages passed via the Peer Resolution Protocol with PIP itself being layered on top of the PRP.

Rendezvous Protocol (RVP)

The RVP is used solely for the dissemination of messages and advertisements to peers within a peergroup. Rendezvous nodes use the RVP in order to distribute messages to each other and to client peers in a controlled and efficient manner.

Pipe Binding Protocol (PBP)

Layered on top of the Endpoint Routing Protocol, the PBP is used by JXTA applications to communicate with each other via a virtual channel between two endpoints (a n input and output) referenced by a Pipe Advertisement.

Pipes may be viewed as a named message queue supporting a number of operations including create, close, send and receive.

There are three varieties of pipe:

Unicast The simplest pipe definition, Unicast pipes are unsecure and unreliable allowing messages to be passed from one source to one destination.

Secure Unicast An extension of the Unicast pipe that provides security via a virtual TLS connection.

Propagate A diffusion pipe that allows for one source to multiple destinations. Analogous to an IP multicast - any endpoint that "registers" (i.e. creates an associated output pipe) to an input pipe receives messages sent to that input pipe. Like the Unicast pipe above, messages are unsecure and unreliable.

Chapter 3

State of the Art

3.1 JXTA Performance and Evaluation

JXTA is a peer-to-peer platform composed of 6 core protocols designed for ad hoc, pervasive and muli-hop network computing. Although many applications have been developed on top of the JXTA middleware little investigation has been preformed on the scalability of all aspects of the JXTA protocol stack.

The majority of early evaluations, [3], [4], [5] and [6], have focused on benchmarking specific aspects of system functionality - specifically message throughput in pipe communications. These experiments, performed in a testbed environment, have also lacked the quantities of peers required to draw meaningful conclusions on the operation of a large scale system.

While [7] focused on benchmarking JXTA discovery and rendezvous protocols within on a testbed environment using a larger number of peers, the tests performed focused on the number of peers required for an inconsistent state to develop with regards to local peerview caches. This inconsistency is expected in a larger scale JXTA network with no claims made on the part of the JXTA middleware to provide a system wide consistent peerview - a difficult (if not impossible) ask in any distributed network dis-

playing a high rate of churn. Another important point to recognise, as specified by [1] with work on the simulation of the JXTA lookup protocol, is the inability to achieve meaningful results within a testbed environment for a single specific JXTA protocol due to the constant conflicting presence of other JXTA protocols or services.

3.2 Evaluation of peer-to-peer Protocols

With regards to peer-to-peer (P2P) systems, a number of mechanisms exist with which to evaluate system behavior, each with an associated benefits and drawbacks. Due to the inherent distributed nature of P2P systems, any form of rigorous evaluation may prove to be a major challenge. A breakdown of potential approaches that may be used to evaluate such systems is provided below:

Crawlers A crawler may be viewed as an autonomous software agent that resides on a particular node, collating and record information as it passes through the network. Various crawler agents deployed throughout a network may be utilised to monitor certain network metrics. However, two main drawbacks exist with this approach. Firstly it is impossible to gain a global view of such a monitored network as there is no direct information available for nodes with no crawler agent. Secondly, the act of recording and actively processing the information passing through a particular node can have a non trivial impact on this nodes performance as a network peer.

Simulators A simulator is an application designed to model a specific network system or algorithm. Such applications allow for specific input models and physical infrastructures, along with a number of predefined or adaptive network conditions, all within a virtual sandboxed environment. By defining a constrained, virtual version of a real world environment simulators may be used to automate the evaluation of network behavior under a variety of specified parameters. However

while levels of realism may be introduced to a simulation model with the addition of parameters to simulate latency, noise, errors etc. the main drawback of any formal network simulator is the inherent detachment from real-world results.

Emulators While a simulator models a network system or algorithm in software, an emulator duplicates a systems behaviour using resources provided by another existing system. While potentially allowing for more realistic system evaluation then that provided by a simulation, the execution of emulated systems are tied to the physical constraints of the system used for emulation, specifically message propagation delays between emulated peers.

Testbeds A testbed is a physical platform for experimentation of large-scale network projects. Usually results gained from testbed experiments yield the most applicable and realistic results. The main drawback of the testbed approach is that of cost, especially for the evaluation of distributed systems in a context of scalability. As such, large scale testbeds for academic research purposes are usually collaboratively operated in order to distributed the cost among many sites and maximise the available nodes for deployment of systems for evaluation. PlanetLab is possibly the most well known example of this approach.

Simulators allow the evaluation of P2P protocols without the complexity and expense of deploying emulation or testbed systems. Furthermore, results gained via varied initial parameters (e.g. number of network nodes, initial network conditions etc.) may be easily compared over various simulation iterations with output analysis being performed as part of each simulation.

3.3 Peer-to-peer Simulators

3.3.1 Network Simulators vs Overlay Simulators

Within the context of P2P protocol evaluation there are two methods of simulation - Network and Overlay.

Network simulators are concerned with providing a low level emulation of standard network and transport protocols (TCP, UDP, IP etc.) that behave in a manner analogous to that of the same protocols in real-world data networks. Such simulators provide a sandbox based on these transports with a selected protocol for evaluation being implemented on top of the transport protocols provided by the simulator. Network simulators are most suited to the evaluation of smaller scale network protocol operation and as such are not recommended for evaluations where scalability is a core experimentation metric. Simulators such as NS-2, Opnet, Narses and OMNET++ fall into this category.

Overlay simulators are less concerned with the lower level transport protocol network operation and are more focused on simulating the particular protocol intended for evaluation. Due to the scalability limitations inherent in network simulators mentioned above, the most apparent choice for evaluating a P2P protocol is that of an overlay simulator.

3.3.2 Desirable characteristics of P2P simulators

Given the decentralised, self-organising nature of P2P systems and protocols any selected simulator overlay must fulfill a number of requirements in order to provide a meaning simulation environment. The following list provides an overview of some of these desirable characteristics The following list shows some of the desirable characteristics a peer-to-peer simulator should have as identified within [8] and [9]:

Scalability A general requirement of any new P2P system is to support scalability to many thousands of nodes. A P2P simulator must therefore be capable of efficiently running simulations with very large numbers of peers (in the hundreds of thousands).

Usability and Documentation Such characteristics are related directly to how easy the simulator application is to learn, use and extend. Documentation and an active development/support community is also an important feature to consider when selecting a particular simulation technology.

Simulation Architecture As mentioned in 3.3.1 (page 16) network simulators tend to closely model lower level network/transport protocols making the interaction of such transports and higher level P2P protocols all the more realistic. Overlay simulators tend to focus more on verifying algorithms associated with the higher level P2P protocol and may offer less editable parameters related to lower level protocol operation. Depending on the user case of a particular simulation, either model may be of use. As such, an interchangeable underlying network model would be preferable.

Flexibility Related closely to the underlying Simulation Architecture, a particular simulator should be able to support the simulation of structured and unstructured networks while providing rich feature-set of editable parameters in order to affect various virtual network conditions.

Statistics A key attribute of any research tool is the ability to gather meaningful and useful results. Such results should be provided in a means that is easy to manipulate, quantify and perhaps represent in a clearly understandable form - graphs, charts etc. A mechanism should also exist in order to allow for taking snapshots of simulator state when certain parameters are reached for verification of repeatability.

While difficult to build a simulator that satisfies all the requirements specified above, a trade-off of realism vs. usability is often necessary. Specifically in order for a simulator to offer high scalability, the network layer will need to be modeled less realistically, offering only a wireframe model of core low level network functionality. This will reduce simulation overhead required to more fully simulate the low level network within the system concerned with each nodes communications stack and allow for more focus on simulating the higher level P2P protocol in a scalable environment.

3.4 PlanetSim

PlanetSim, as described in [10], is a "discrete event-based simulation framework tool for overlay networks and services". JXTA-Sim, to be discussed in Section 3.5. has been developed and built on top of the PlanetSim architecture.

PlanetSim consists of a three layered design. The application layer is concerned with application layer messages, and messages traversing to and from the Application and EndPoints.

The overlay layer represents a middleware between the simulated distributed network stack and application. There are four main components at this level:

Node A central class within the Overlay Layer, the Node class is used to simulate a node's behavior. The algorithm or algorithms to be emulated are specified within this class. A Node object is also responsible for sending and receiving messages to and from the Application and endpoint Objects that exist in the PlanetSim Application Layer.

Id Used to identify nodes on the network, this class is overided to hold a protocol specific identifier.

NodeHandle Representing a handler used for node manipulation, a NodeHandler is

composed of an Id and boolean variable to denote if a node is currently alive.

RouteMessage A RouteMessage defines internal system messages sent between End-Points (on the Application layer) and Nodes. Such messages contain routing specific information such as source, destination and next hop.

The Network layer represents a virtual physical network on which the Overlay layer is built. While this network may to be extended and modified to increase realism, for algorithm verification purposes this layer simply provides a means for Nodes to communicate with each other via RouteMessages.

3.5 JXTA-Sim Version 1

JXTA-Sim is a simulator described in [Garcia, 2008] that works on simulating the behavior of a single JXTA protocol - the JXTA Lookup Algorithm as described in [11]. Built on top of PlanetSim, JXTA-Sim was designed with three core principles:

Scalability As a major selling point for the JXTA system itself, scalability was a core requirement in the development of JXTA-Sim. As specified in Section 3.4 (page 18) PlanetSim can simulate up to 100,000 distinct nodes. As such, JXTA-Sim has an upper supported node count of 100,000 during a single simulation.

Extensibility JXTA-Sim undertook the simulation of a single core JXTA protocol. However, JXTA-Sim was designed in a modular fashion to allow for additions of further JXTA protocols.

Usability JXTA-Sim was designed to be easy to use at all stages of simulation - from the specification of initial simulation parameters to the formatting of simulation results.

Chapter 4

JXTA-Sim Overview

4.1 JXTA-Sim Architecture

Introduced by S.Garcia [1], JXTA-Sim was created to evaluate the lookup algorithm used by JXTA in discovering advertisements using the Rendezvous and Peer Discovery Protocols. JXTA-Sim, being built on top of the PlanetSim architecture need only implement a number of interfaces provided by the PlanetSim system in order to achieve a viable virtual environment in which to base its simulations.

In this Chapter an overview of JXTA-Sim1 will be provided, including high level class architecture, an overview of the implemented peers and message types as well as a synopsis of a general simulation. This chapter finishes with some limitations inherent with version 1 of JXTA-Sim.

4.1.1 Packages and classes

Figure 4.1 shows the interfaces provided by PlanetSim and the associated implementation classes provided by JXTA-Sim.

From this diagram we can see clearly the mapping between JXTA-Sim objects and the underlying PlanetSim interfaces.

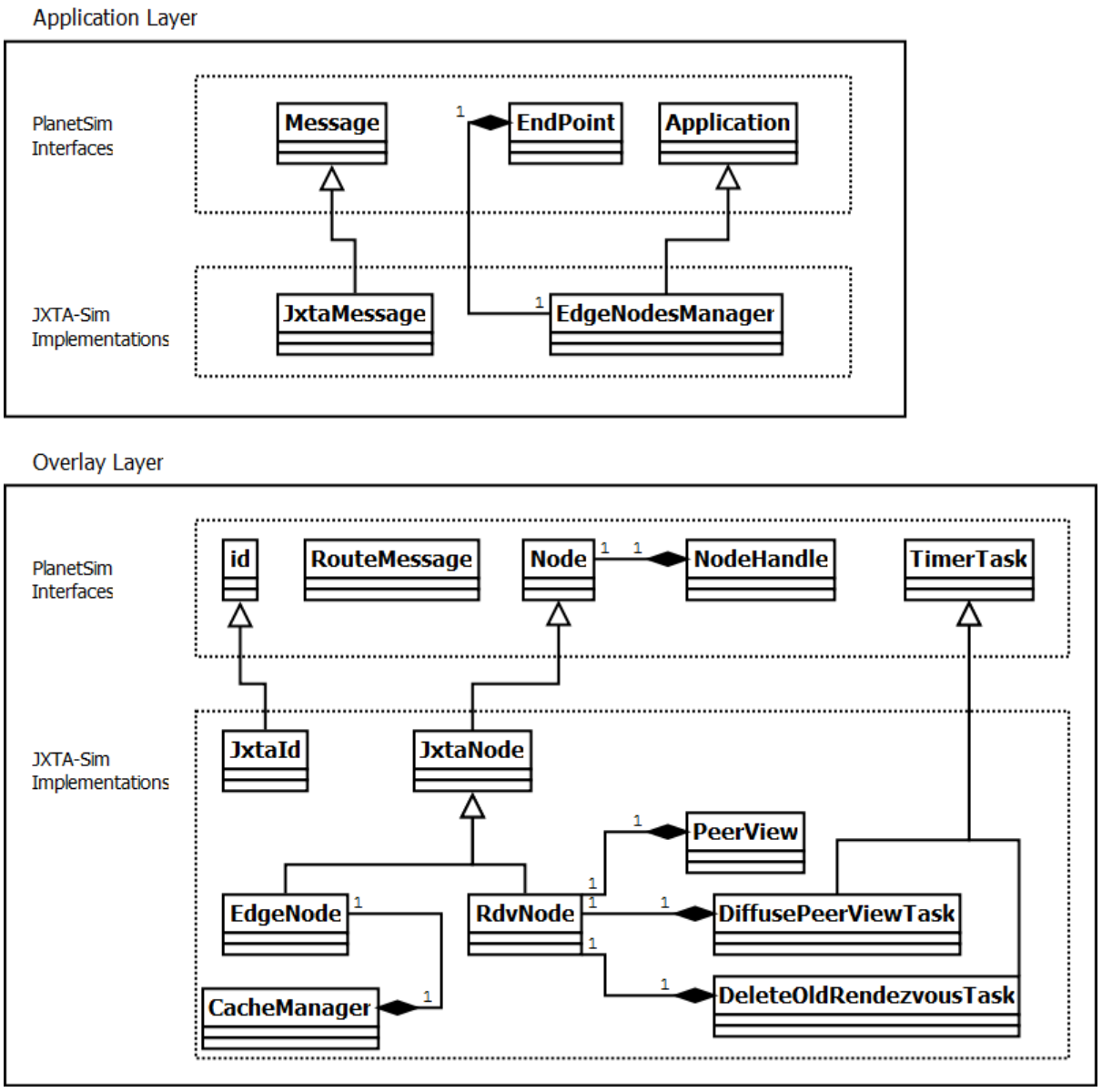

Figure 4.1: Relationship between JXTA-Sim1 and PlanetSim Classes

From Section 3.4 we recall that a PlanetSim application is concerend with the simulation of application messages between virtual nodes within the Overlay Layer. For JXTA-Sim this involves the definition of an *EdgeNodeApplication* that is associated with a number of *EndPoints*. This application then coordinates the passing of

JxtaMessage message types to and from each application EndPoint.

Two types of Nodes are defined (which may be later used as EndPoints within the EdgeNodesApplication) - RdvNode and EdgeNode, which represent the JXTA Rendezvous Peer and JXTA Edge Peer respectively (Section 2.2.2) .

Node types

RdvNode Implementing the functionality provided by a JXTA Rendezvous node, an RdvNode node is tasked with generating a PeerView (as seen in **??**) - an ordered list of Rendezvous peer identifiers and then disseminating this peerview throughout all RdvNodes within the simulation.

The task of managing EdgeNodes associated with a particular RdvNode is handled by the EdgeNodeManager application (an instance of which is associated with each RdvNode) with periodic refreshes and removal of outdated cache entries handled by two additional classes "DifusePeerViewTask" and "DeleteOldRendezvousTask".

EdgeNode Within JXTA-Sim1 an EdgeNode is only concerned with publishing a Peer Advertisement referencing itself and the ability to lookup other advertisements from the JXTA overlay. This lookup is achieved via the PDP (Section 2.3.2) and its two tiered lookup system - checking first with a local cache and then querying an EdgePeers associated RdvPeer.

JXTA Messages

During a standard JXTA-Sim simulation, nodes within the PlanetSim overlay will communicate by sending a variety of messages to each other. PlanetSim provides applications 3 modes of communication - Request, Reply and Refresh - with a Refresh message being a Request that does not require a response.

It is important to note that while all JXTA messages (advertisements, application layer messages etc.) are described using XML, JXTA-Sim abstracts this requirement for the purposes of simulation and represents these messages as either java objects.

As part of a JXTA-Sim simulation, 4 standard messages are used

Peer View Message Used by RdvNodes to send a copy of the PeerView to each other.

Edges List Message Used to send a list of Ids representing an unordered list of EdgeNodes.

Advertisement Message Used to send or receive information on an advertisement.

Discovery Message Used to lookup or publish a particular request. This message is main message type used when a PDP lookup request has been generated by a node within the JXTA Overlay.

4.2 Simulation

The following section provides an overview of running a JXTA-Sim1 simulation.

4.2.1 Configuring a simulation

PlanetSim provides a config file with which to affect a number of simulation parameters. JXTA-Sim, being concerned with the simulation and evaluation of the JXTA lookup algorithm, extends the options available for configuration via this file. Taken from the JXTA-Sim properties config file, a list of the configurable options is provided in Table 4.1.

Parameter	Description
JXTA_STABILIZATION _STEPS	The default number of stabilize steps
JXTA_CACHE_SIZE	Maximum Size of the cache to store advertisements
JXTA_BITS_PER_KEY	The default number of bits for JxtaIds
JXTA_EDGES_PER_RDV	Maximum Number of Edge Peers per Rendezvous
MAX_WALKER_HOPS	Maximum Number of Hops for the Walker
MAX_PEERVIEW_SIZE	Maximum size of the peerview of a rendezvous node
REPLICATION_DISTANCE	Distance to replicate the index in the close rdvs
SIMULATION_STEPS	Steps we want to simulate after the initialization of the network
NUM_SEARCHES	Number of published advertisements and lookups performed.
NUM_RDVS_IN	Number of Rdvs that will join the network during the simulated steps
RDVS_OUT	Number of Rdvs that will leave the network during the simulated steps
PEERVIEW_REFRESH _INTERVAL	Interval of time to send the some of the RPV to random rdvs
PEERVIEW_ENRIES _FLUSH_INTERVAL	Interval of time at which to remove the entries of the RPV which have expired
RPV_ENTRY_DEFAULT _EXPIRATION	Default expiration of an entry of in the RPV

Table 4.1: Configurable options for JXTA-Sim1

4.2.2 Building a network

When a JXTA-Sim simulation begins, the network is initial populated with RdvNodes. These RdvNodes can be specified within an event file, or programatically.

Upon completion of the RdvNodes being added, JXTA-Sim then iterates through all RdvNodes within the Overlay and adds a number of EdgeNode to each (determined by the JXTA_EDGES_PER_RDV parameter as specified in Table 4.1).

4.2.3 Outputs and results

JXTA-Sim provides a number of visualisation mechanisms during simulation to allow for verification of system operation against protocol specification.

Figure 4.2 shows message types sent during the initial steps of all nodes joining a rendezvous and a number of rendezvous nodes choosing to part.

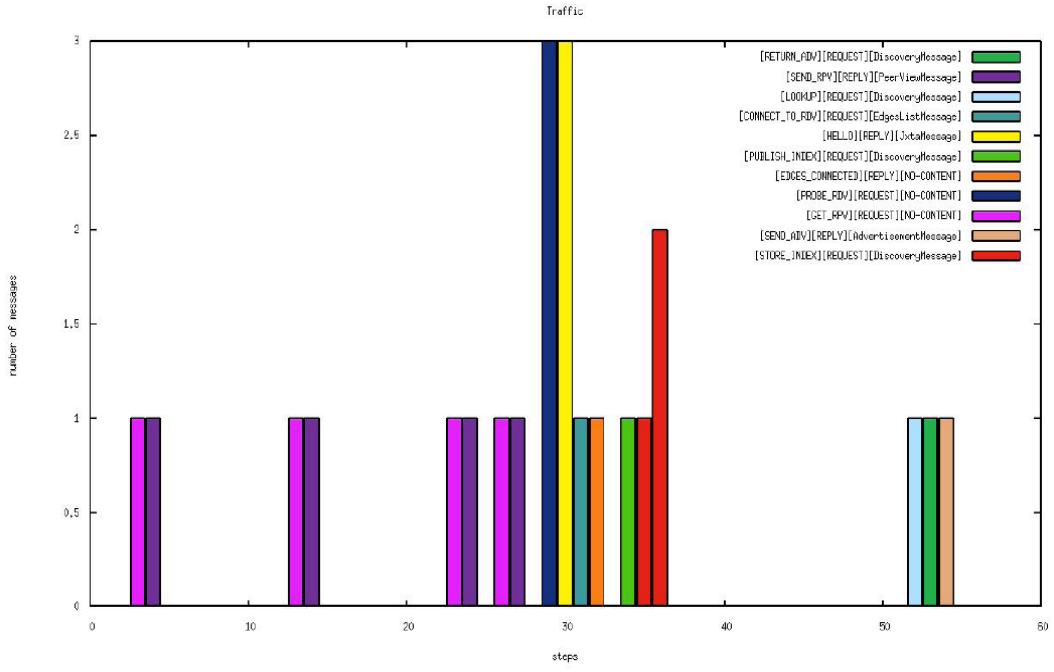

Figure 4.2: JXTA-Sim message counts for a 5 rendezvous network

Figure 4.3 shows a network diagram of the network used in the generation of mes-

sages counted in figure 4.2 above, consisting of 5 rendezvous nodes and a number of edge nodes.

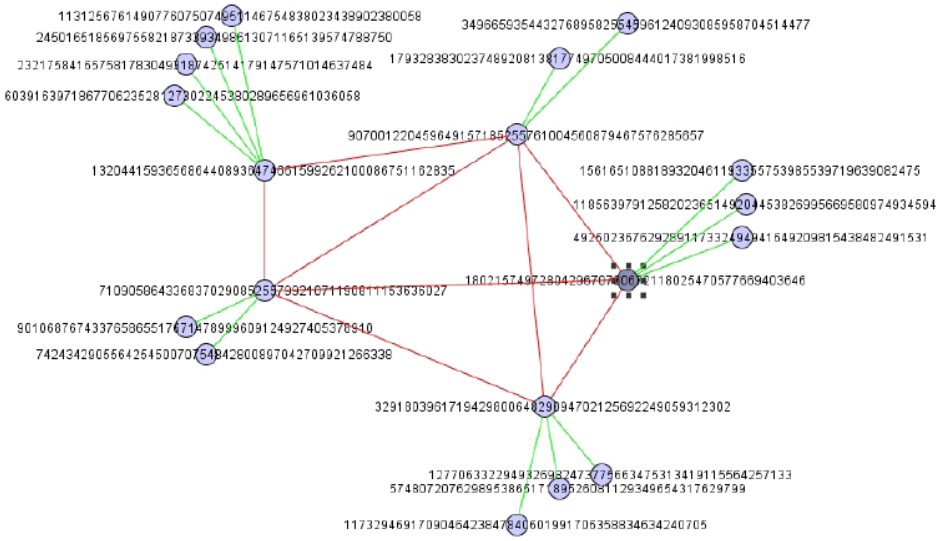

Figure 4.3: JXTA network with 5 rendezvous nodes with a number of associated child nodes

Figure 4.4 shows a network consisting of 100 rendezvous nodes and 123 edge nodes distributed among the 100 rendezvous nodes. While PlanetSim has already offered a researcher the ability to take a snapshot of node information to pass to Gnuplot for visualisation purposes, JXTA-Sim introduces the concept of message type for graphing purposes. As such, JXTA-Sim includes additional meta-information in a graph specification to allow easy visualisation of traffic types.

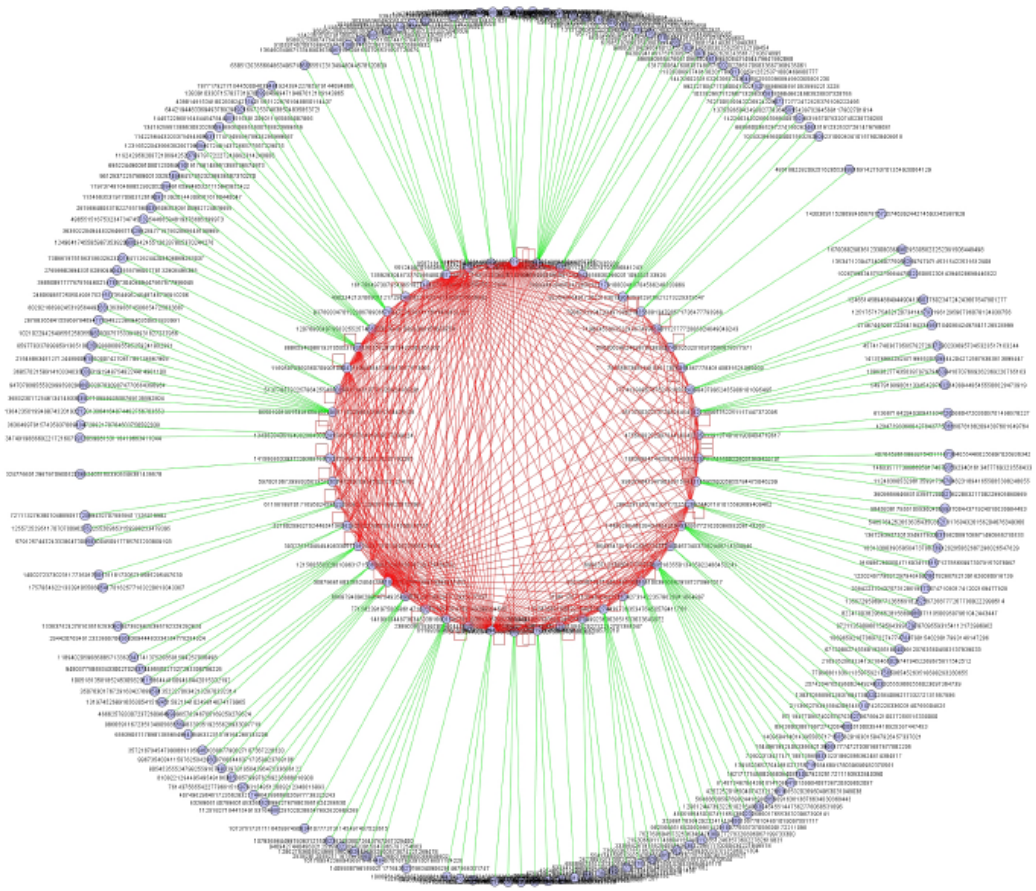

Figure 4.4: JXTA network visualisation comprising 223 nodes

Chapter 5

JXTA-Sim2

JXTA-Sim2 is an extension to JXTA-Sim providing additional peer and protocol support. JXTA-Sim2 implements support for Relay peers, allowing for more complex nested subnets within the JXTA-Sim simulation. In order to support communication via these Relay peers, JXTA-Sim2 also implements the Endpoint Routing Protocol a distributed routing table to be built as part of the overlay and for endpoints to discover each other.

This chapter details the additions to JXTA-Sim.

5.1 Packages and Classes

Figure 5.1 below shows an updated view of the high level class relationships within the planet.jxta package. Changes from JXTA-Sim1 are clearly labeled.

From this class diagram we can see the addition of the RelNode, an implementation of the JxtaNode interface. Each RelNode has an associated "RouteView" (named for JXTA-Sim2, this term is not referenced from the JXTA protocol specification) and each RouteView is comprised of many Route Advertisements.

The EdgeNodeApplication used to implement the lookup algorithm in JXTA-Sim1

is also extended to provide functionality for our RelayPeers. As there is scope for some code reuse, both RelNode and RdvNode will register with the EdgeNodeApplication to leverage the required protocol logic.

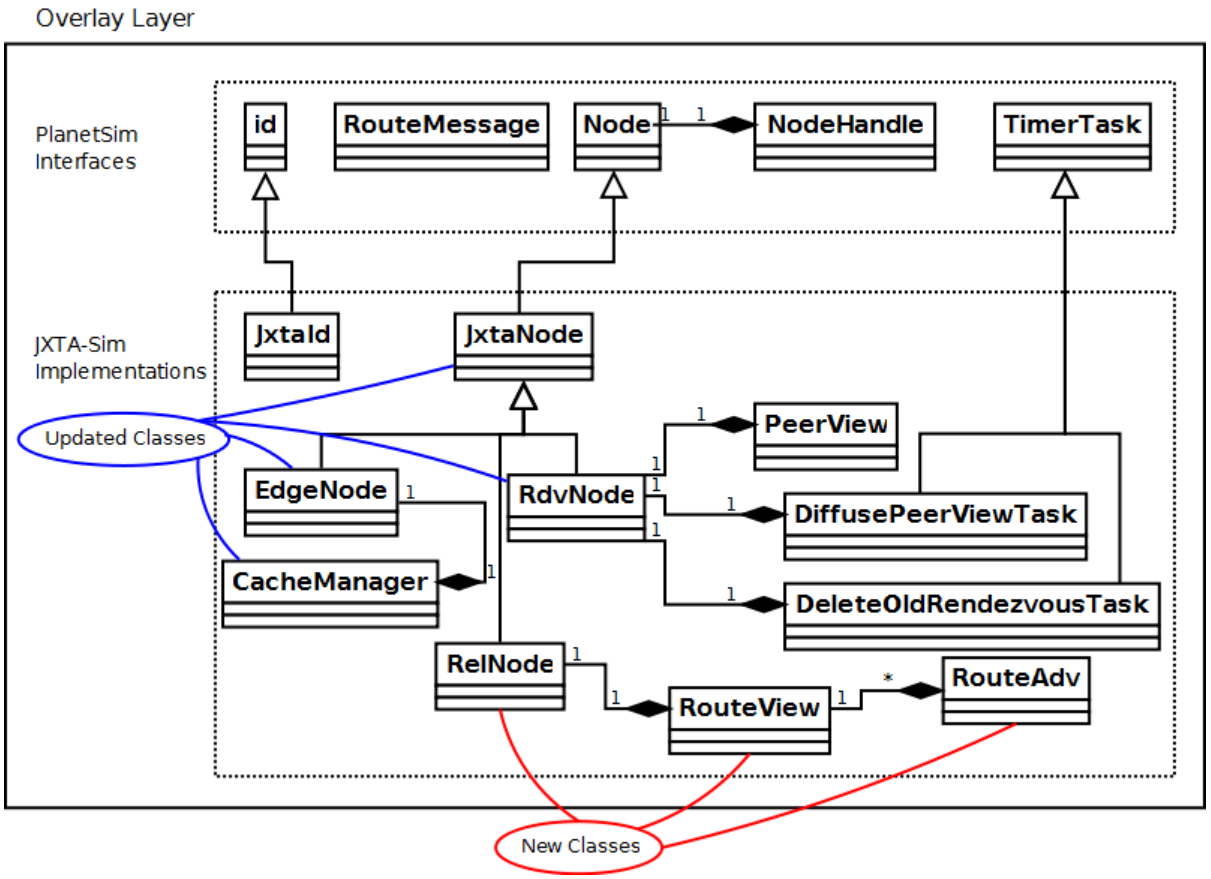

Figure 5.1: Relationship between JXTA-Sim2 and PlanetSim Classes

5.2 JXTA Network nodes

This section will introduce the new JXTA-Sim node type, RelNode and also detail changes made to the EdgeNode and RdvNode classes as part of the addition of the ERP.

5.2.1 Relay Node

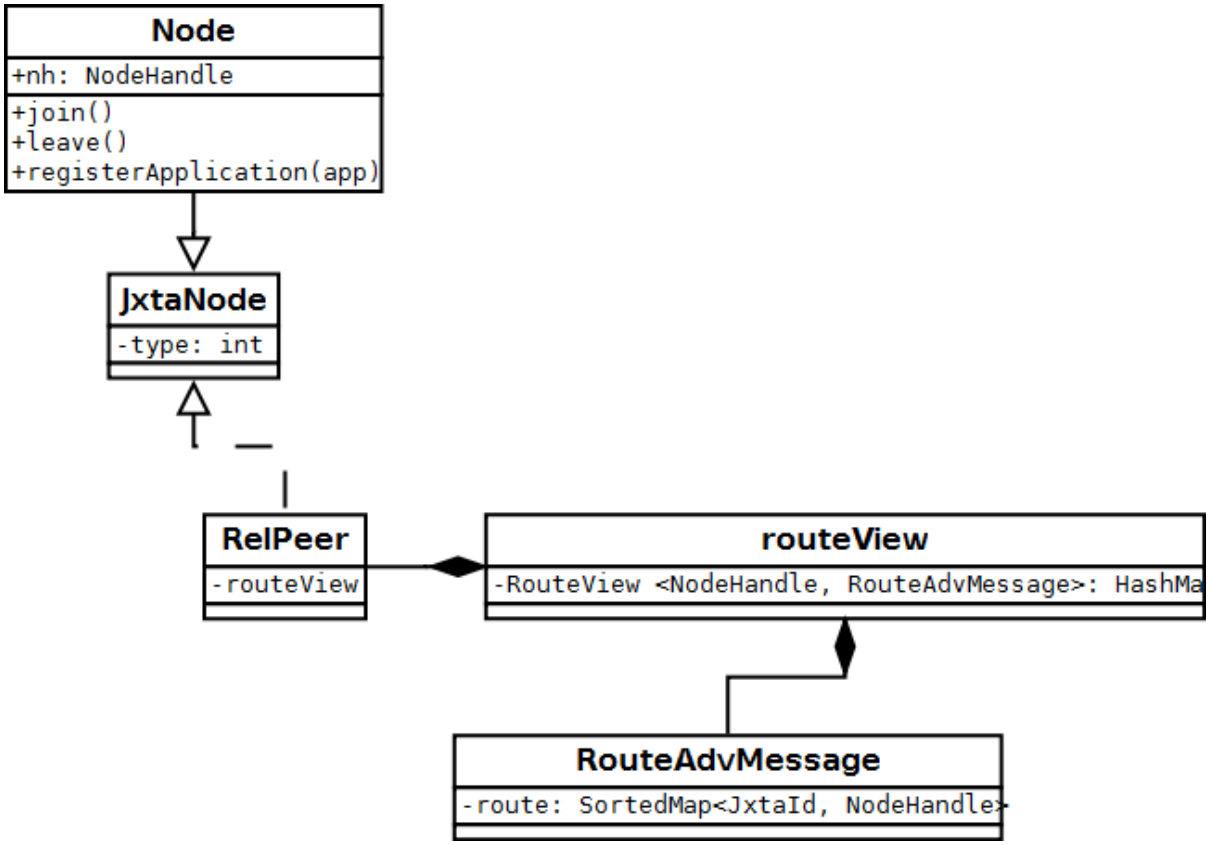

Figure 5.2: Classes relating to the Relay Node

The RelNode keeps an updated RouteView, a table of all current routes that have been registered with this Node.

Peers of type RelNode accept RouteAdvMessages from EdgeNodes and RelayNodes, preform some minor processing and forward the RouteAdvMessage to its immediate parent.

5.2.2 Edge Node

As an extension from the EdgeNode class in JXTA-Sim1, the JXTA-Sim2 edge node is updated to preformed a triggered event upon joining the network - the propagation of

a GET_CONNECT message to this nodes direct parent (either RelNode or RdvNode).

An EdgeNode may also preform a LOOKUP_ROUTE query against another *route peer* (in JXTA-Sim2, all route peers are RelNodes or RdvNodes) in order to receive an enumeration of hops to a desired endpoint.

5.2.3 Rendezvous Node

As an extension from the RdvNode class in JXTA-Sim1, the JXTA-Sim2 rendezvous node has the ability to accept GET_CONNECT messages from any child and will store the contained RouteAdvMessage within a local RouteView.

5.3 JXTA Messages

The following display the additional message types implemented as part of JXTA-Sim2

5.3.1 GET_CONNECT

The GET_CONNECT message is sent when a "child node" type (either EdgeNode or RelayNode) initial joins the simulation network. Thie message is propagated upwards, to the source nodes immediate parent.

Upon reception of a GET_CONNECT, the node will propagate it upwards until the local node is an RDV node.

The GET_CONNECT message will always contain a single RouteAdvMessage.

RouteAdvMessage

A RouteAdvMessage, an extension of JxtaMessage, is a simple message type containing an ordered list of JXTA IDs, known internally to JXTA-Sim2 as a RouteView.

5.3.2 LOOKUP_ROUTE

A LOOKUP_ROUTE message can be sourced from any JxtaNode. This message is trigger and dispatched to an appropriate parent node (either a node's bootstrap or an ancestor of said node). This message allows for routing information to be gained in order for message delivery to a particular endpoint node.

5.4 Endpoint Routing Protocol

The Endpoint Routing Protocol, introduced in Section 5.4, is a mechanism to allow JXTA peers with no direct physical connection to communicate via one or more Relay Peers. This is achieved by two mechanisms - firstly, a means to build a routing table distributed throughout the JXTA overlay and secondly, a means for any peer to locate a route for a certain endpoint.

For the purposes of the JXTA-Sim2 simulation, we may think of all Relay and Rendezvous peers as having the ability to respond to route lookup messages - i.e. all Relay and Rendezvous peers may function as a "Peer Router" [12]. When a Peer Router receives a route query it answers the query by returning an ordered list of hops to final destination endpoint. Messages may then be routed based on this enumeration of hops towards the preferred destination.

It should be noted that at any point during routing an intermediate hop may lose connectivity, thus rendering the route obsolete. At this point the current Peer Router must be able to discover a new route in order to complete message delivery or hold the message pending a route becoming available (based on a timeout value associate with the message in question).

5.4.1 Route Generation Algorithm

Figure 5.3 provides a simple flowchart for the lifecycle of a JxtaNode as it joins the JXTA-Sim2 overlay network.

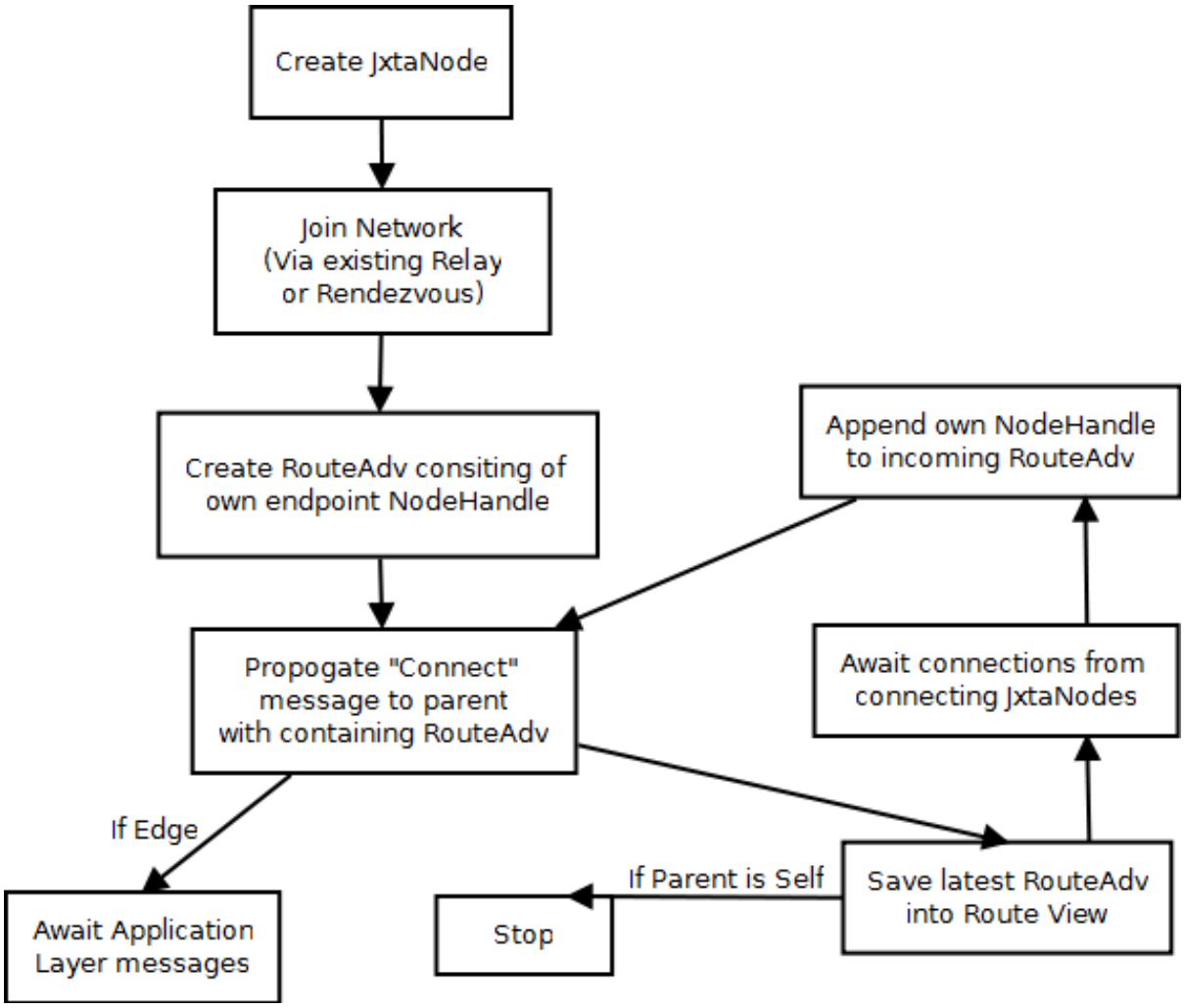

Figure 5.3: Endpoint Routing Flowchart

From this Figure we can see two potential termination points - If the current node is an Edge node post propagation of a GET_CONNECT message or if the current node's parent is itself - i.e. if this node is a Rendezvous node.

Parameter	Description
JXTA_EDGES_PER_RDV _REL	Maximum Number of Edge Peers per Rendezvous and Relay
JXTA_RELAY_DEPTH	Maximum depth of nested subnets separated by Relay Peers
NUM_ROUTE_LOOKUPS	Number of route lookups preformed post simulation

Table 5.1: Additional configurable options for JXTA-Sim2

5.5 Simulation

In this Section an overview of JXTA-Sim2 in operation is provided, from configuration to generating results.

5.5.1 Additional configuration options

JXTA-Sim2 is configured in the same manner as JXTA-Sim1 with some additional configuration options available to the researcher. Options are as defined in Section 4.2.1 with the additions laid out in Table 5.1:

The options JXTA_EDGES_PER_RDV_REL and JXTA_RELAY_DEPTH refer to initial network creation while NUM_ROUTE_LOOKUPS places a limit on the number of route lookups preformed by Edge and Relay peers post simulation.

5.5.2 Running a Simulation

As with JXTA-Sim1, a simulation can be run programatically or via an events file. In JXTA-Sim2 an events file contains an initial list of Rendezvous peers for network creation with Relay and Edge nodes being created and joined to the network by the simulator itself.

A code sample for initiating a JXTA-Sim2 simulation is given in Listing 5.1.

Listing 5.1: Sample code for starting a simulation

```
1 Vector events =EventParser.parseEvents(Properties.simulatorEventFile);
```

```
2  Scheduler timer = new Scheduler();
3  timer.addEvents(events);
4
5  NetworkSimulator sim = new NetworkSimulator(timer);
6
7  for(int i=0; i<events.size();i++)
8    sim.simulate();
9
10 for (int i=0 ; i<((JxtaProperties)Properties.overlayPropertiesInstance
       ).maxRelsDepth; i++)
11   addRelayNodesToRelsRdvs(sim.getInternalNetwork());
12
13 addEdgeNodesToAllRdvsRels(sim.getInternalNetwork());
```

Line 1 selects an event file as specified in the jxta.properties config file.

Lines 2 & 3 associate the events with a scheduler for addition to the PlanetSim overlay.

Line 5 Creates a new PlanetSim simulator instance and associates the events.

Lines 7 & 8 preform the required number of steps for the PlanetSim simulator to consume the events file.

Lines 10 & 11 build a layered network of relays based on parameters specified in the jxta.properties config file.

Line 13 adds EdgeNodes to all Rendezvous and Relays.

5.5.3 Simulation Lifecycle

This section will provide a visualisation of the steps required for network creation and for the JXTA-Sim2 route view to be propagated from an Edge peer to a central Rendezvous peer.

Applicable configuration options used for this run of JXTA-Sim2 are shown in listing

5.2 with an eventfile containing a single Rendezvous node.

Listing 5.2: Configuration options used for demonstration simulation

```
1 JXTA_EDGES_PER_RDV_REL = 1
2 JXTA_RELAYS_PER_RDV_REL = 1
3 JXTA_RELAY_DEPTH = 2
```

Figure 5.4 shows the network post the addition of all Rendezvous nodes - in this circumstance, only one Rendezvous node was used. At this point a global Rendezvous Peer View exists within this single RdvNode. The red edge connected to itself signifies this peer, *907001220459649157185255761004560879467576285657*, is a Rendezvous Node.

Figure 5.4: Network Generation Step 1 : Rendezvous node added

Figure 5.5 shows us the network post the first round of relay additions. At this point, a RouteView exists on the Rendezvous Node containing two entries of single hop destinations for both Relay Nodes.

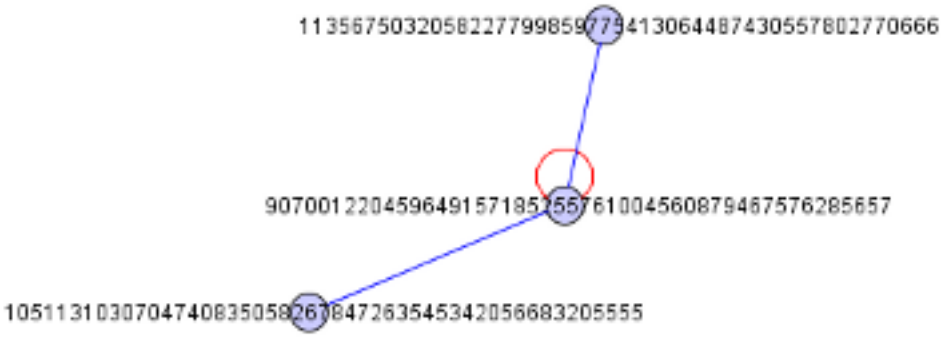

Figure 5.5: Network Generation Step 2 : First round of relay peers added

Figure 5.6 shows us the network post the second round of relay additions. At this point, two GET_CONNECT message has been passed from the outermost relay edges to *1051131030704740835058267847263545342056683205555*. This relay has then appended its own ID onto the top RouteAdvertisements of both edge relays and has forwarded a GET_CONNECT message to the Rendezvous.

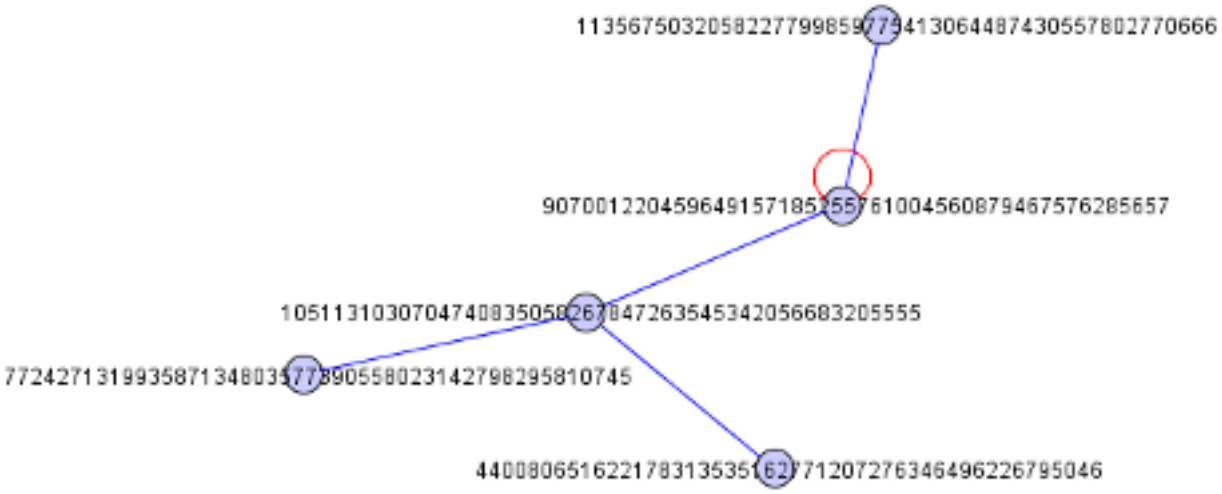

Figure 5.6: Network Generation Step 3 : Second round of relay peers added

Figure 5.7 shows the network post addition of Edge Nodes to all Relays and Rendezvous. As in step 3 above, all intermediate Relay Nodes have now appended their own IDs onto each Edge Node's RouteAdvertisement and have propagated the message to the Rendezvous.

5.6 Gathering results

JXTA-Sim2 provides 3 ways to gather information pertaining to the simulation.

Gathering statistics during simulation The PlanetSim architecture lends itself well

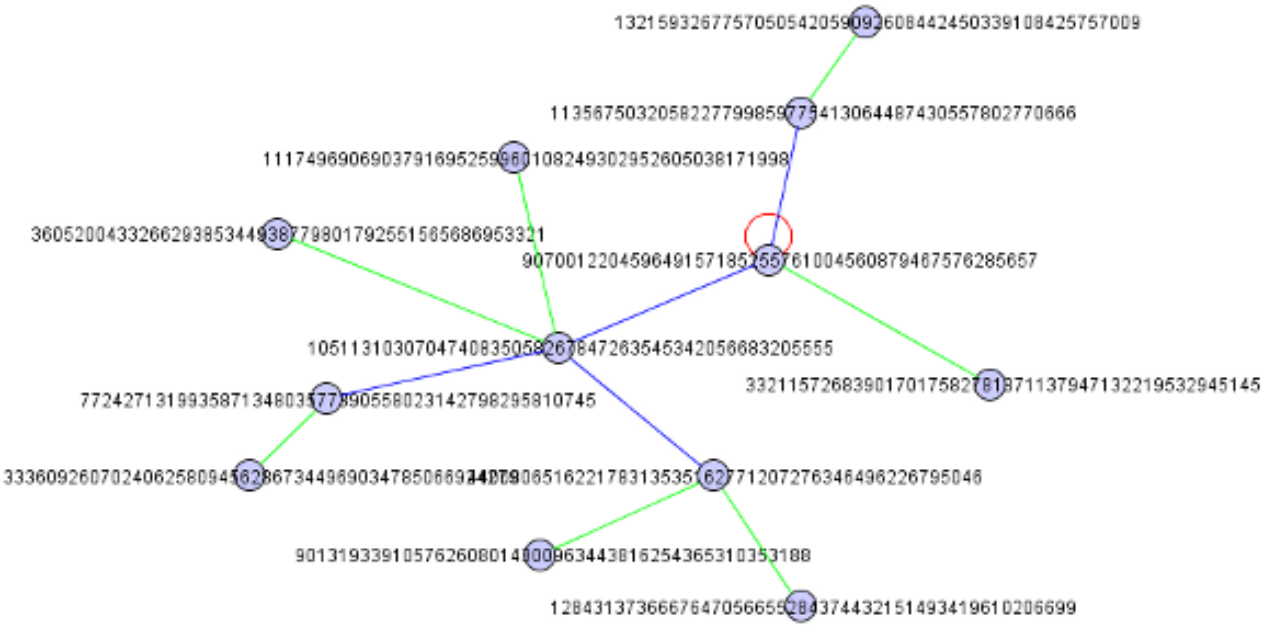

Figure 5.7: Network Generation Step 4 : Edge peers added

to using *Aspect Oriented Programming* techniques for extracting statistical information from a running simulation.

JXTA-Sim1 modified an existing aspect for recording all messages as they are sent via the PlanetSim overlay. Utilising this aspect we can gain information on messages passed during the ERP operation.

Gathering protocol information post simulation JXTA-Sim2 provides a mechanism to directly query a Relay or Rendezvous for a copy of this peers RouteView, represented as a table of Route Advertisements.

Generating a graph of the final network PlanetSim provides a mechanism for which to generate graphs of the current network overlay at any stage throughout the simulation.

While JXTA-Sim2 makes use of this functionality to provide graphing at key

stages throughout the network generation process along with the ability to query nodes directly for information regarding local caches, Route Views etc. an option is also given post simulation to generate a final view of the network.

Figure 5.8 shows a network graph post a simulation consisting of 10 Rendezvous, 10 Relay x 4 Relay Depth each having 5 associated Edge Peers (totaling 23201 unique peers).

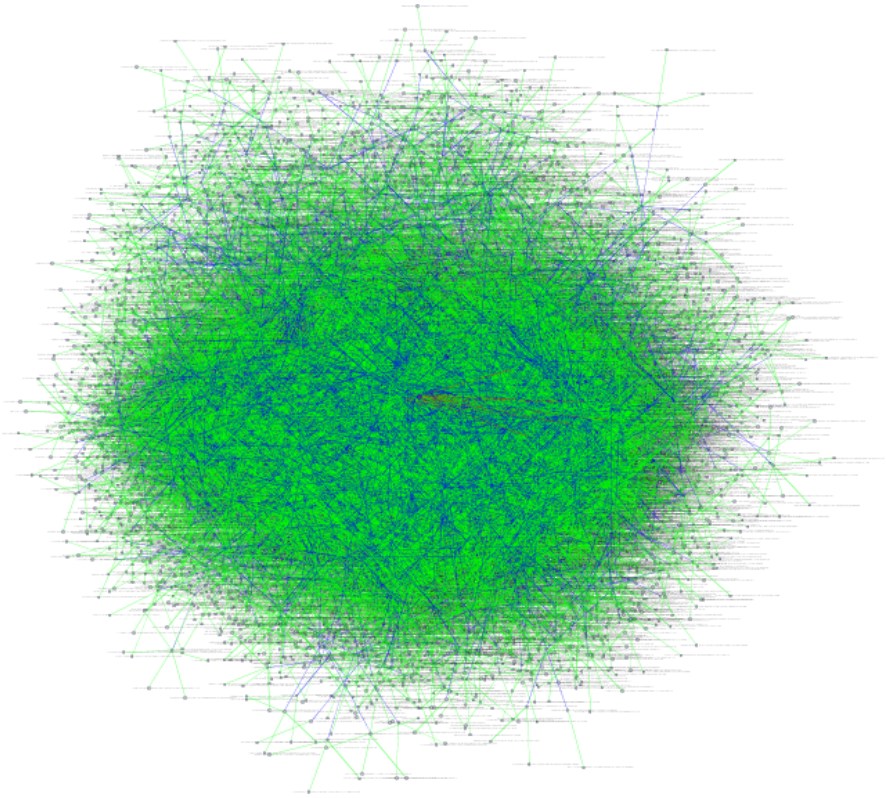

Figure 5.8: Extremely dense final network graph, organic layout

Chapter 6

JXTA-Sim2: Results

After the implementation of JXTA-Sim2, a number of test evaluating the correctness of the simulation are completed.

Towards this end, a number of statistics were gathered. Traces of messages sent as part of the ERP were collected during simulation along with graphs of network layout.

Post simulation, graphs were loaded into a graph visualiation toolkit with routes manually verified with complete RouteView tables from all core JXTA peers.

For the purposes of this section, the JXTA-Sim2 specific simulation parameters in Listing 6.1 with an eventfile containing 10 rendezvous peers seeding the initial network.

Listing 6.1: Configuration options used for demonstration simulation

```
1 JXTA_EDGES_PER_RDV_REL = 3
2 JXTA_RELAYS_PER_RDV_REL = 4
3 JXTA_RELAY_DEPTH = 3
```

6.1 Validation

After a simulation run with 1096 steps, Listing 6.2 shows network size statistics.

Listing 6.2: Node counts post simulation

```
1 Network size: 1918 nodes
2 Number of Edge Nodes:1374
3 Number of Relay Nodes:534
4 Number of Rdv Nodes:10
```

A visualisation of this network graph post simulation is provided in Listing 6.1.

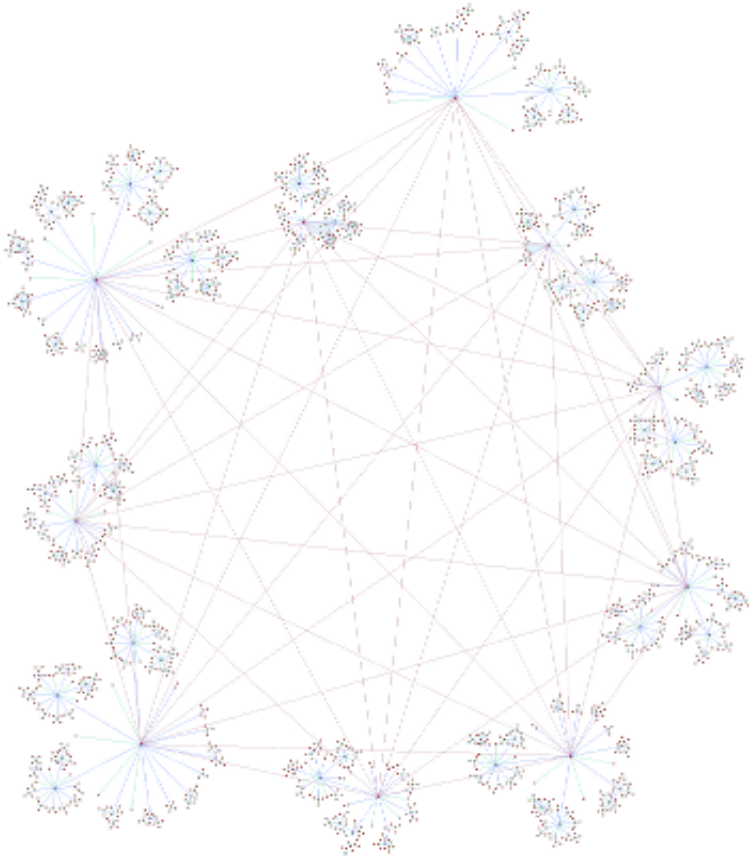

Figure 6.1: JXTA-Sim2 post simulation graph for 1918 nodes

From a trace provided during simulation (Figure 6.3), 469 route advertisements were generated in the construction of this networks.

Listing 6.3: Configuration options used for demonstration simulation

```
1  Current RouteAdv contains 4entries
2  Entry added to RouteAdv at
       116374408916589761534358376310143866113185201641 2 sourced from
       592045117257534190812679479089422797908861241074
3
4  Current RouteAdv contains 4entries
5  Entry added to RouteAdv at
       116374408916589761534358376310143866113185201641 2 sourced from
       457443236903550747675534568254051698412221590 63
6
7  Current RouteAdv contains 4 entries
8  Entry added to RouteAdv at
       116374408916589761534358376310143866113185201641 2 sourced from
       166762796472287003952708123306105347157271949892
9
10 Current RouteAdv contains 4 entries
11 Entry added to RouteAdv at
       116374408916589761534358376310143866113185201641 2 sourced from
       592045117257534190812679479089422797908861241074
12
13 Current RouteAdv contains 5 entries
14 Entry added to RouteAdv at
       116374408916589761534358376310143866113185201641 2 sourced from
       592045117257534190812679479089422797908861241074
```

By confirming these results, JXTA-Sim2 has been proven to behave as expected.

Chapter 7

Conclustions and Future Work

7.1 Conclusions

The goal of this project was to extend the functionality of the JXTA-Sim simulator in order to support disparate nested networks using the JXTA Relay Peer with connectivity being achieved using the Endpoint Routing Protocol. Furthermore, verification of the ERP to prove support for edge routing under "wide" graphs, diameters of up to 40 or 50 nodes.

The functionality provided as part of JXTA-Sim1, specifically that related to the evaluation of the lookup algorithms, remained intact, even with communications now being proxied through, potentially, many intermediate disconnected subnets. The RdvNodes have also continued to support a PeerView distribution with the addition of a RouteView for peers not directly accessible from the "core" JXTA network overlay.

However, the number of routing messages suffers an exponential growth with regards to the depth and quantify of nested subnets within the overall JXTA overlay. To this end it might be suitable to implement an additional route logic as part of the core ERP algorithm - specifically, the addition of static routes.

Given reasonable assumptions on the nature of nested subnets - that of a large

institution providing a number of gateways with reasonably high uptime to peers existing within a private network outbound to the Internet with limited further internal subnets - there could be a substantial message saving if static routes were held for these relays and not propagated further.

JXTA provides a mechanism to developers to implement more intelligent routing for route peers [12] involving the implementation of a routing service on the application layer of a JXTA system providing hints to the underlying routing nodes. Dependant on application and rate of churn inherent in a particular JXTA deployment, it would be reasonable to assume message savings could easily be achieved beyond the core ERP implementation to lessen traffic further.

7.2 Future Work

There are a number of different study avenues that may be undertaken from this point.

One line of study would be the addition of "churnable" node behaviour for Rendezvous and Relay nodes. Currently in JXTA-Sim2, once a Rendezvous or Relay node is added to the network, this node never leaves.

There is currently no support for measuring latency or timeout values analogous to real world networks. PlanetSim developers have been working on two avenues to address this - extensions to simulate realworld network layer communications or the ability to leverage the network layer from an existing third party network level simulator such as NS2.

A core assumption of JXTA-Sim (both versions) is the lack for any kind of concept relating to PeerGroups. As JXTA supports a node belonging to multiple peergroups with varying search scopes, extending JXTA-Sim to be cognitive of this fact may yield interesting results.

With the addition of support for the ERP for simple message passing, the way is

open for the support of Pipes within JXTA-Sim. Supporting this higher level communications abstraction would require the complete implementation of a further JXTA protocol (specifically the Peer Endpoint Protocol).

The above suggestion with regards to supporting Pipe based communication may be extended further with the ability for JXTA applications to be deployed on JXTA-Sim directly. Unmodified JXTA applications running on a reasonably complex simulated network with the ability to trace communications, receive snapshots of system services at any time at any node would be of great benefit for any potential peer-to-peer developer.

Finally, extending the use of *Aspect Oriented Programming* and graphing ability within PlanetSim could be used to develop a step-by-step visualisation engine allowing for a non-realtime visualisation of network creation and message passing. Such a system would be very usefull as both a learning tool and for dissection of the various JXTA protocols.

Bibliography

[1] S. Garcia Esparza, "JXTA-Sim: A simulator for evaluating the JXTA Lookup Algorithm," Master's thesis, Trinity College Dublin, 2008.

[2] https://jxta.dev.java.net/, *JXTA Community Portal, (fetched September 2010).*

[3] E. Halepovic and R. Deters, "The jxta performance model and evaluation," *Future Generation Computer Systems*, vol. 21, no. 3, pp. 377–390, 2005.

[4] E. Halepovic, R. Deters, and B. Traversat, "Performance Evaluation of JXTA Rendezvous," *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pp. 1125–1142, 2004.

[5] E. Halepovic and R. Deters, "JXTA performance study," in *2003 IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2003. PACRIM*, vol. 1, 2003.

[6] E. Halepovic and R. Deters, "The costs of using JXTA," in *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, p. 160, Citeseer, 2003.

[7] G. Antoniu, L. Cudennec, M. Jan, and M. Duigou, "Performance scalability of the JXTA P2P framework," in *IEEE International Parallel and Distributed Processing Symposium, 2007. IPDPS 2007*, pp. 1–10, 2007.

[8] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers, "The state of peer-to-peer simulators and simulations," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, p. 98, 2007.

[9] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *IEEE Global Internet Symposium, 2007*, pp. 79–84, 2007.

[10] J. Pujol-Ahulló, P. García-López, M. Sànchez-Artigas, and M. Arrufat-Arias, "An extensible simulation tool for overlay networks and services," in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 2072–2076, ACM, 2009.

[11] B. Traversat, M. Abdelaziz, and E. Pouyoul, "Project JXTA: A Loosely-Consistent DHT Rendezvous Walker," *At http://research. sun. com/spotlight/misc/jxta-dht. pdf*, 2003.

[12] M. Duigou, "JXTA v2. 0 protocols specification," *IETF Working Group Draft Specification*, p. Sec 3.2., 2006.