Modelling and Verification of JXTA peer-to-peer Network Protocols

Yannick L. Kala Konga
F'SATI
Tshwane University of Technology
Pretoria, South Africa
yann.lokombo@gmail.com

Karim Djouani
F'SATI
Tshwane University of Technology
Pretoria, South Africa
djouani@gmail.com

Guillaume Noel
F'SATI
Tshwane University of Technology
Pretoria, South Africa
noelg@tut.ac.za

Recent advances in peer-to-peer computing have allowed its evolution as a reliable alternative to traditional centralised computing methods. The JXTA project is a popular open source describes a platform formed by six protocols purposed to enable interoperable, ubiquitous and reliable peer-to-peer networking. We present a formal model of integrated JXTA protocols using Promela. We subsequently verify the model with the SPIN model-checker for internal consistency. Because the integrated model proves to be too large formal verification due its size and complexity, we verify the protocols separately. Number of non-progress cycles and an invalid end state are detected and we provide possible solutions approaches for these errors.

Formal verification, Formal specification, Finite-state machine, JXTA, Model-checking, Peer-to-peer, SPIN.

1. INTRODUCTION

Peer-to-peer networks are based on the distribution of resources. Network participants are not the traditional server and clients but rather peers operating in a decentralised and often independent manner. As an alternative to traditional computing models such as the client-server model, the peer-to-peer concept is not new. Peer-to-peer networks gained high exposure with Napster's support of music files sharing (Milojicic et al., 2003). Their benefits include improved scalability, resources aggregation, lower cost of ownership and potential superior efficiency; as such their deployment was envisioned to be pervasive.

In spite of these benefits, the deployment peer-to-peer networks have is not as ubiquitous as anticipated. In fact, peer-to-peer networks are still mainly deployed in casual file sharing and instant messaging applications. Use and acceptance, security and interoperability constitute their most important challenges (Milojicic et al., 2003). However, steady progress in distributed computing now allows peer-to-peer systems to be considered for building critical and dependable services and applications. JXTA (for "juxtapose") (Sun Microsystems Inc., 2007), an open-source effort from Sun Microsystems represents one such progress (Domingo-Prieto, Arnedo-Moreno and Herrera-Joancomartí, 2010).

JXTA is a platform, designed as set of six protocols intended to provide the basis of building a wide range of peer-to-peer services and applications. These protocols allow the development of more specialised, robust and reliable peer-to-peer applications (Spaho et al., 2010; Hossain et al., 2011). JXTA protocols are generic enough to enable implementation in any programming language, operating system or network transport (Gong, 2001). They enable ad hoc, pervasive and multi-hop peer-to-peer networking in almost any application scenarios.

Besides, benefits derived from the application of formal methods to a system or a concept abound. For instance, the deeper understanding a designer gains from the formally specifying the system under test allows to increase confidence in design choices and to highlight improvements areas (Clarke and Wing, 1996). Also, formal verification analyses the specified system for desired properties. Furthermore, the complexity of some systems increases the likelihood of the existence of subtle errors. Communications systems are notorious for their complexity and the difficulty of finding errors the complexity incurs. Errors can go undetected, passing even the most rigorous simulation tests and causing significant losses. In fact, the software bug in a single switching station that caused the failure of AT&T's entire long-distance telephone switching on 15 January 1990 is illustrative of this reality.

Particularly, using the transition-based specification PROMELA (Protocol Meta Language), it is possible to adequately model the behaviour of finite-state concurrent reactive systems such as communication protocols. In addition to that, the behaviour of the modelled system can be formally verified with the SPIN (Simple Promela Interpreter) model-checker (Holzmann, 2003). SPIN accepts Promela-based models and Linear Temporal Logic (LTL) claims to perform formal verification. It has been used to adequately verify communication protocols and systems (Jing and Jinhua, 2009, De Renesse and Aghvami, 2004). However, the verification of an integrated model of the JXTA protocol suite exposes a significant open problem in formal verification: the state space explosion.

Hence the purpose of this paper is to present a formal specification and verification of JXTA protocols based on their informal specification in (Sun Microsystems Inc. 2007). The rationale behind this work is:

- To use the developed model as the basis for model-checking automated verification;
- To uncover possible design errors, incompleteness, logical inconsistencies;
- To pinpoint improvement areas, and;
- To present issues around resolving the state space explosion problem with respect to an integrated model.

The formal specification of JXTA could thus provide a reliable basis from which improvement can continually be made to the platform. Also, improvement of reduction and abstraction techniques, compositional verification and the SPIN algebra could be centred on the formal verification and the resulting state space explosion problem. In the end, while technical problems will be solved at a faster pace, JXTA-based peer-to-peer systems and the peer-to-peer computing paradigm could gain more acceptance from the use of the formal methods. This paper is a first step towards these objectives. Full formal specification of the entire platform is performed and initial verification of done. The results presented here will therefore provide the basis and direction for more advanced research, which would ultimately permit to reach the set goals behind the entire effort.

The rest of the paper is organised a follows. Section 2 provides an overview of the JXTA platform. We briefly present PROMELA and the SPIN model-checker in section 3. Section 4 deals with the modelling decisions and assumptions while section 5 and 6 discuss the formal models and the results of the verification using SPIN. Section 7 concludes the paper and discusses possible further work.

2. THE JXTA PROTOCOL SUITE FOR PEER-TO-PEER NETWORKS

2.1. The JXTA Platform

JXTA is an open network programming and computing platform for the peer-to-peer paradigm. The platform is specified as a set of protocols purposed to allow the collaboration of all connected devices as peers (Sun Microsystems Inc. 2007). JXTA intends to enable the development and deployment of interoperable services and applications, thus advancing the adoption of peer-to-peer computing in mainstream computing systems. All design considerations in the JXTA framework are governed by 3 essential objectives (Gong 2001):

- **Interoperability**: JXTA-based services should be able to seamlessly interact with each other by creating a common infrastructure with standard software and primitives used by all peer-to-peer systems. This approach breaks away with initial peer-to-peer services such as Napster or AIM that were locked within one specific service, incompatible with others.
- **Platform Independence**: JXTA is designed to be agnostic of the programming language, the development environment or the deployment platform used. With JXTA, a peer should be able to use any application, irrespective of the implementation behind the peer or the application.
- **Ubiquity**: JXTA technology should be able to run on any digital device ranging from a simple sensor, a network switch to a supercomputer. By remaining neutral of hardware capabilities or configuration of peers, JXTA peer-to-peer networks could have a ubiquitous presence.

Based on these objectives, the JXTA platform defines seven conceptual elements fundamental to the behaviour of the platform (Sun Microsystems Inc. 2007): identifiers, peers, peer groups, pipes, advertisements, credentials and messages.

A JXTA **Identifier (ID)** is used to provide unique identity to entities and resources in JXTA so that they can be referred to unambiguously and canonically. Identifiers are opaque: the context in which they appear in a protocol is enough to infer their type.

A **peer** is any addressable networked device running at least JXTA's core protocols. A peer can be a mobile phone, smartphone, laptop, sensor, desktop computer etc. Although additional roles can be taken up, peers are first assumed equal in functionality and capabilities.

A **peer group** is a collection of peers with common goals and interests. They can be considered as logical groupings used to restrict access to resources. Peer grouping is one of the most important architectural aspects of JXTA as most features can only be accessed in a peer group setting. Hence two default peer groups are specified in all JXTA implementations: **netPeerGroup** and **WorldGroup** peer groups, which all peers join at start up. Peers can self-organise in peer groups and a single peer can participate in as many peer groups as it requires. JXTA Protocols do not command peer groups formation but simply provide the means to do so.

A **pipe** is the central data communication means in a JXTA network: a virtual communication channel used by peers to exchange data. The qualities of service they provide allow to differentiate between pipe types: unidirectional asynchronous, synchronous request response, bulk transfer, streaming and secure. However, JXTA protocols require only the unreliable unidirectional asynchronous pipe type for their operation. This choice significantly influences the behaviour of protocols and the model we develop.

Advertisements are neutral metadata structures used by the protocols to describe all resources, including peers. They contain the ID and other required information of the resource they describe. Their importance is critical in JXTA since a peer cannot access a resource without its associated advertisement.

Messages are the basic data exchange unit in JXTA. Protocols and peers interactions make use of messages to exchange data and communicate. The current specification of JXTA (Sun Microsystems Inc. 2007) implement messages as XML documents that can hold any type of data.

Credentials are tokens appended to a message body to identify the sender and verify its access rights. They are used to respond to the need for support of different levels of access in a dynamic peer-to-peer environment.

2.2 The JXTA protocol suite

JXTA has a 3-tier architecture shown in Figure 1:

- The *JXTA* **core** is at the heart of every JXTA network. It includes compulsory functions enabling a rudimentary level of operation: peer creation, basic communication and security primitives are thus incorporated here.
- The *JXTA* **services** make use of the core's functionalities to provide higher level of operation that is not essential but suitable including discovery and resources publications.
- The *JXTA* **applications** make use of JXTA services and core to implement functionalities such as distributed computing, real-time collaboration, etcetera.

The 3-level architecture (figure 1) is used to organise all six JXTA protocols for the development of higher-level services and applications. The protocols are designed such that they may operate independently. However, the core protocols always need to be implemented by a device before it is addressable in the network. Further, the full potential of JXTA can be utilised when all protocols are supported. In fact, interaction between protocols, including the optional ones, can allow a higher degree of operation. For instance, the pipe binding protocol can use the peer discovery protocol to discover an incoming pipe to bind. JXTA protocols are described next.

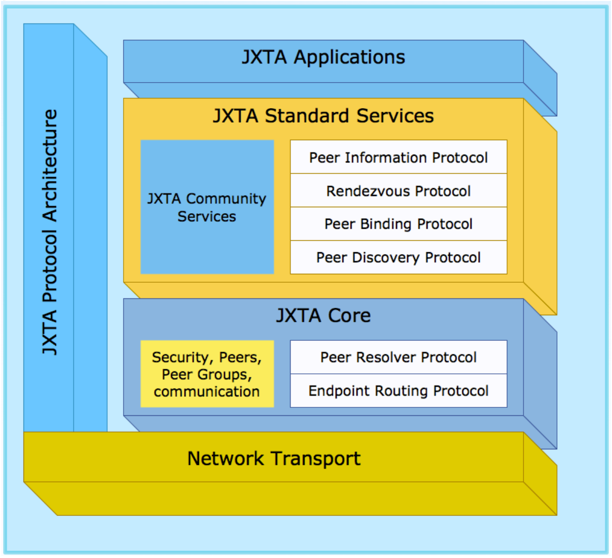

***Figure 1**: JXTA Protocols Architecture*

2.2.1. The Endpoint Routing Protocol (ERP)

The Endpoint Routing Protocol provides a simple routing primitive for JXTA peers. It is responsible for finding routing information for the relay of messages between source and destination peers. The ERP protocol is specified as an exchange of queries and responses to determine routing information. Routing information is determined even for peers behind a firewall or with a private IP address space because the ERP protocol is transparent to underlying transport or network configurations. The ERP relies on special peers called **peer router** for its operation.

Peer routers are self-elected peers that cache more routing information than required and make the information available to other peers. They handle new route queries for the peers they serve and have the ability to bridge different logical or physical networks. A peer therefore needs to be connected to at least one peer router before getting a new route. Peer routers provide low-level infrastructure for the basic routing in a JXTA network. When a peer does not have a route to send a message to another peer, it requests new information from its serving peer routers with a query resolved by the peer resolver protocol.

2.2.2. The Peer Resolver Protocol (PRP)
The Peer Resolver Protocol provides a mechanism for the resolution of queries and responses among protocols and services run by each peer and peer group. The resolver protocol issues a query on behalf of a peer to a peer group and later matches possible responses to the query. The protocol makes use of named handlers in its resolution tasks. These named handlers, with specific query strategies and policies, determine how a query is distributed and how the responses should be processes or handled.

Further, the PRP protocol performs authentication and authorisation based on the credentials contained in resolver messages. Security in JXTA is based on JXTA membership service with its default **Personal Security Environment** (Arnedo-Moreno & Herrera-Joancomarti 2009). This is one example of the importance of peer groups in JXTA. Thus, the credential in the resolver message allows granting of access rights to a service (represented by a named handler in PRP) or resource and confirmation of the identity beyond the simple peer ID. The ERP and PRP protocols constitute the core protocols. The remaining protocols form the standard services protocols.

2.2.3. The Peer Discovery Protocol (PDP)
This protocol enables resources publication with the dissemination of advertisements. Thus, a peer can discover resources within its peer group with the PDP protocol. Resources in JXTA can be a peer, a peer group, a pipe or any custom entity with an advertisement. Besides, the PDP as a default discovery mechanism, is the foundation on which more sophisticated discovery services can be built. Also, services leveraging PDP can improve its efficiency by providing additional information.

2.2.4. The Pipe Binding Protocol (PBP)
The pipe binding protocol is used by applications and services for peer-to-peer communication. In essence, the protocol is required to dynamically bind a pipe to endpoint at runtime. Because of its nature, it requires the ERP protocol to function. Furthermore, because unidirectional communication is assumed in JXTA, the PBP protocol actually binds an input pipe to an output pipe. Besides, the protocol also specifies a propagate pipe type that does not require binding to specific input pipes since all peers in a peer group can listen to it.

2.2.5. The Rendezvous Protocol (RVP)
The RVP protocol facilitates controlled propagation of messages in a peer group. It is also based on special peers, **rendezvous peers**, cooperating and performing additional tasks required to control the propagation of messages. The rendezvous protocol is divided in three sub-protocols:

(i) The Rendezvous Propagation protocol: for the actual propagation of individual messages in the peer group.
(ii) The Rendezvous Lease protocol: a subscription service used by non-rendezvous peers.
(iii) The PeerView protocol: a rendezvous management protocol used by rendezvous peers.

2.2.6. The Peer Information Protocol (PIP)
The PIP protocol enables peers to enquire on the status and capabilities of others peers. Layered on the resolver protocol, it is the simplest of all JXTA protocols. Status information include traffic information, uptime, load etcetera.

3. PROMELA AND THE SPIN MODEL CHECKER

We perform formal verification of JXTA protocols using the SPIN model checker (Holzmann 1997). Model-checking in SPIN is uses finite-state machines modelled as processes in the Promela language. Because model-checking is performed on an abstracted system, Promela is intended to provide system description at a relatively high level of abstraction (Holzmann 1991). Implementation-oriented aspects such as time and memory management are intentionally left out. In fact, Promela emphasises on modelling process synchronisation and coordination between processes. Furthermore, Promela targets the description of reactive systems and is a transition-based specification language (Lamsweerde 2000).

Consequently, operational semantics of Promela are expressed in terms of transition systems. This makes Promela particularly suited for describing the behaviour of distributed or concurrent systems (Merz 2001). For example, asynchronous communication protocols or synchronous variable programs can be optimally modelled in Promela and verified in Spin. Peer-to-peer networks including JXTA exhibit both concurrent and distributed behaviour. They are spatially separated and are based on sharing of resources. Besides, properties of transition systems can be conveniently conveyed in temporal logic, which Spin converts into Büchi automaton (never claims) for verification.

In fact, asynchronous processes and inter-process communication, message channels, synchronising statements and structure data as well as implicit and explicit correctness claims constitute the basis of the verification performed in SPIN. SPIN checks the model for logical inconsistency and flaws based on the presence of deadlocks, live-locks and improper terminations. This check is based on the Promela model containing explicit specification of correctness such as assertions or implicit specification of correctness like control structures.

Furthermore, automated verification of Promela models with full state space search is done using nested depth-first search or alternatively, breadth-first search(Holzmann 1997) (Holzmann 2003). For large models, partial order reduction and state space compression are used to mitigate the state explosion problem. Where the amount of memory on the system running the verification, is simply not sufficient for an exhaustive state space search, the supertrace or bitstate algorithm can be used instead. However, as we shall see, the state space explosion can persist and prevent the verification process to complete. In this case, other reduction and complexity management techniques need to be applied.

4. FORMAL MODELLING DECISIONS

By their nature, peer-to-peer systems can be large and complex. This complexity is often translated into the difficulty to arrive at an adequate model describing their behaviour (Velipasalar et al. 2006). Further challenges emanate from the state explosion problem when these models are formally verified. This further explains the importance of abstracting the system's behaviour to a relatively high level. However, abstraction should be carefully applied depending on the system under test. We discuss the modelling choices and abstraction for the JXTA peer-to-peers protocols in this section.

Throughout the Promela models of each protocol, we make use of the data type with low range like `byte`, `mtype` (for message type or symbolic constants) `bit` or a combination of these in custom types to represent data structures. Models of protocols may be small but their analysis may be exponentially more complex due to a number of factors (Merz 2001). Particularly, in Promela, complexity is influenced by variables, the number of processes, message queues (communication channels) and the size of message queues (Holzmann 1991). For example, we abstract a peer's route cache in the endpoint routing as an `mtype` for this reason. Also, we limit the number of processes by keeping different entities to the minimum number of required protocols participants. Message queues and their size can particularly increase the state space required. We reduce the number of communication channels and their buffer size. Hence we include in our model dynamic creation of processes and passing of channels identifiers between processes. This choice allows decreasing the number of asynchrony in our model, which results in the reduction of reachable composite state without reduction of scope. We therefore keep the buffer size of all communication channels to 1. We could have used zero as buffer size but this corresponds to a synchronous channel, which is not recommended as a default assumption for JXTA protocols.

In fact, the JXTA specification in (Sun Microsystems Inc. 2007) makes a number of assumptions that influence modelling decisions and the behaviour of the model. These assumptions define the requirements for JXTA protocols and their implementation and they prescribe the expected behaviour of JXTA networks.

Regarding transport of messages, JXTA requires that the worst-case scenario be assumed. In fact, JXTA makes abstraction of the underlying layer and assumes that the exchange of messages is always unreliable even when using reliable transport protocols such as TCP/IP. Also, messages can be dropped at any time. JXTA core protocols are also required to make no timing requirements and as such, they do not include any notion of time. However, standard protocols may include timing requirements in their exchanges. Finally, the connection status of a peer may change at any without notice due to the fact that the peer may disconnect, reconnect or change configurations at will.

Thus by virtue of they support of very arbitrary environment changes, JXTA networks are highly non-deterministic. The impact of these assumptions can be observed in the complexity of protocols such as the endpoint routing and peer discovery protocols (see sections 5 and 6). In addition to this, JXTA protocols are specified such that they may not maintain any protocol states.

Our model is first based on the integration of all JXTA protocols, assuming that a peer implements them all. Although they may function independently, significant dependencies exist. For example, the resolver protocol is designed to rely on the non-compulsory rendezvous protocol for actual propagation of messages. Likewise, the pipe binding protocol needs a discovered and published (by the peer discovery protocol) input pipe before binding it to an output pipe. The objective is to verify the behaviour of the network at system level, because the network may fail even if individual protocols are proved correct. We however apply protocols layering and structuring to lessen complexity. Thus, in a different approach we verify orthogonal behavioural functions of JXTA by checking the model of each protocol separately. We present the Promela models of the JXTA network next (Promela codes can be obtained by e-mail request to the first author).

5. PROMELA SPECIFICATION OF JXTA

Assumptions about the JXTA networks and the informational specification of protocols in (Sun Microsystems Inc. 2007) constitute the basis for the formal models we discuss here. We started with the specification of core protocols and incrementally incorporated the standard services protocols to create an integrated model of the JXTA peer-to-peer network environment. Considering the independence of JXTA protocols, we model the behaviour of each protocol entity as a Promela process. Inter-process communication is achieved by using communication channels. We describe the behaviour of each entity as a finite-state machine constructed from the Promela code.

We model the ERP protocol with two processes describing the behaviour of each entity: a served peer and a peer router. Figure 2 shows the finite-state machine formalising the behaviour of the served peer. We assume that served peer does not have any route and that its knowledge of at least one peer router is pre-configured. With transient connections and only unidirectional pipes available the model starts with a sequence that sets all initial conditions and addresses. The served peer can either forward a message when a route is known or search for a new route with the help of peer routers.

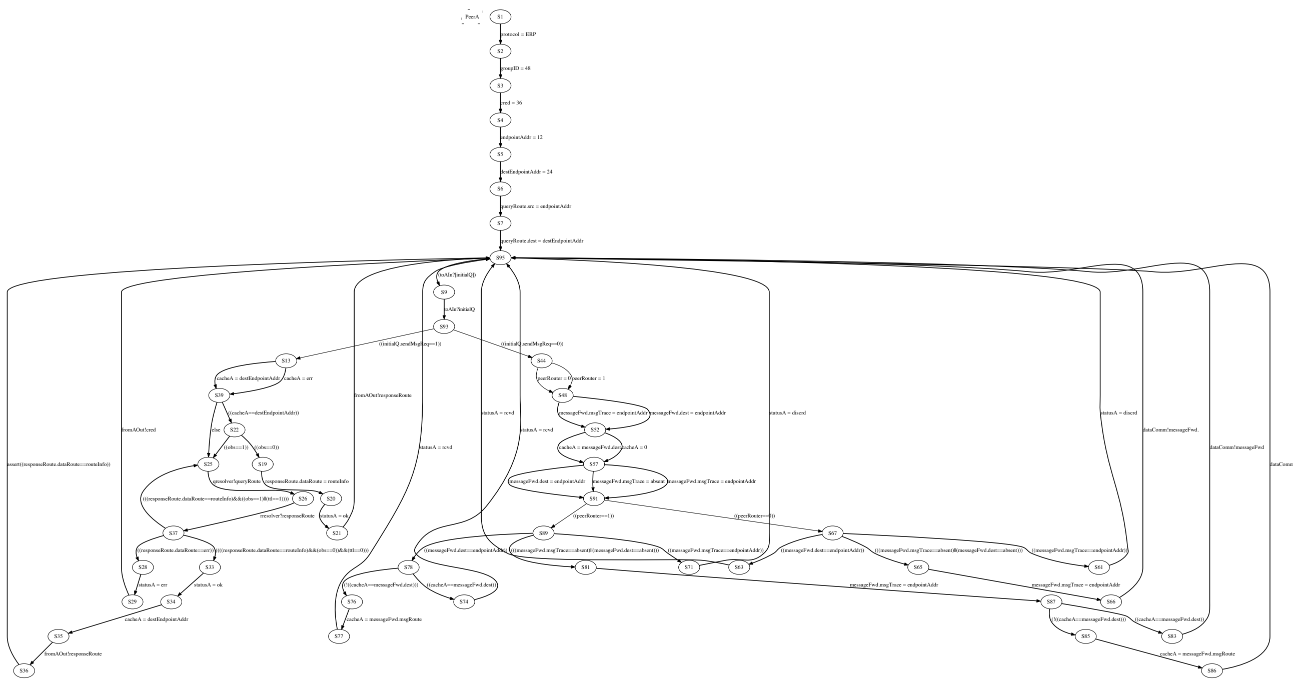

Figure 2: *State transition diagram of the finite-state machine of the served peer in ERP: most of the behavioural features in this protocol entity are related to relaxed communications channel characteristics and the non-deterministic nature of communication.*

Automata related to the new route queries are on the left and those related to the forwarding of a message on the left grouped on right of the state diagram in figure 2. As an entity of a core protocol, the served peer depends on the reception of requests from higher-level services. This dependence is highlighted in the state transition diagram (figure 3) with central input action (S95) leading the channel reception of the request. At reception of routing request the peer checks its cache for route and proceeds to message forward if the route is available and not obsolete. Otherwise, the peer sends a route query message to its peer router(s). Only one send operation is modelled but the behaviour is the same in presence of multiple peer routers.

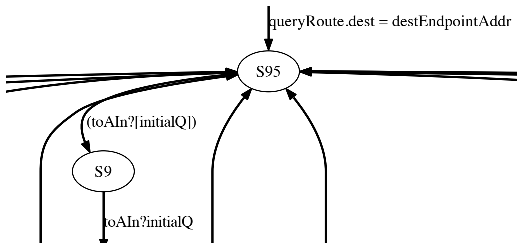

Figure 3: *Closer look on the central state of the served peer's behaviour (extract of the state diagram figure 2).*

In case of a message forward request, the message is discarded received or forwarded if the message is respectively duplicated, intended for the peer or intended for another peer. In the latter case, the peer appends its address in the message trace to prevent duplicate messages. In order to later check for correct behaviour we use control sequences in Promela to ensure that message forward can only be possible when a route is available in cache.

The state diagram in figure 4 shows the automata of the peer router's model. As can be observed by comparing the automaton of the peer router to the automaton the served peer, the behaviour of the former is significantly less complex than the latter's. This is explained by the fact that a peer routing only deals caching route and providing these to its served peers when needed. The peer router does not perform communication checks such as route obsoleteness and time-to-live checks. The communication layer of JXTA, where all messages transit is modelled in as a separate state machine (figure 5). It is simply modelled as a buffer for the required asynchrony. Unreliability is characterised by a non-deterministic decision either correct delivery or message drop. JXTA protocols do not implement any recovery messages such that non-resolvable messages (in error or corrupted) are also dropped. At the core, the layer is mostly accessed by the peer resolver protocol.

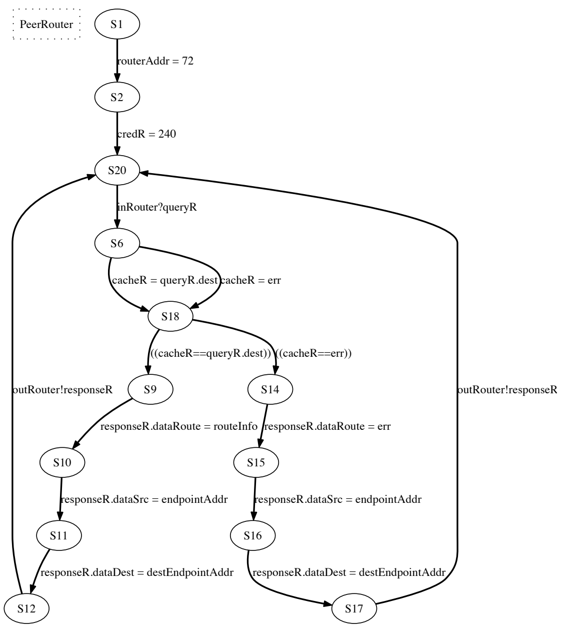
PeerRouter
S1
routerAddr = 72
S2
credR = 240
S20
inRouter?queryR
S6
cacheR = queryR.dest
cacheR = err
S18
((cacheR==queryR.dest))
((cacheR==err))
outRouter!responseR
S9
S14
outRouter!responseR
responseR.dataRoute = routeInfo
responseR.dataRoute = err
S10
S15
responseR.dataSrc = endpointAddr
responseR.dataSrc = endpointAddr
S11
S16
responseR.dataDest = destEndpointAddr
responseR.dataDest = destEndpointAddr
S12
S17

Figure 4: Peer router State Diagram: the behaviour of the peer router is less complex than the peer server's.

Like in all other JXTA protocols, entities of the ERP protocols rely on the peer resolver protocol to issue queries on their behalf and later match responses to these queries. The queries are issues within a peer group. Furthermore, all protocols entities communicating with a peer are required to register a named handler with the PRP protocol. Because all protocols require the service of the PRP protocol to resolve queries and responses, the size of models of PRP resolvers is large.

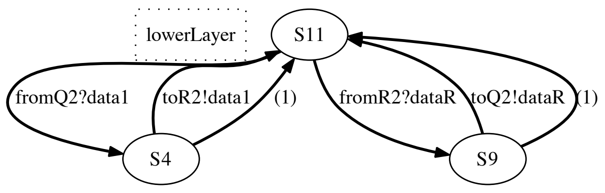

Figure 5: Default communication layer for JXTA protocols: unidirectional, asynchronous and unreliable.

The sending resolver and the receiving resolver are modelled separately. Figures 6 and 7 show their automata. In order to simplify the model, we assumed that only one instance of each service was running the PRP protocols. Hence each entity is modelled to register only one named handler. We abstracted the actual task of propagating queries and responses within a peer group and did not include it in the PRP entities. Consequently, the PRP uses either the communication layer above for propagation or rendezvous peers (from the rendezvous protocol), when supported, for controlled propagation.

A critical task performed by the PRP protocol is the generation of query identities (`queryID`) for every query it resolves in the sender model. These query identities are later matched in the receiver automata such that responses to the query can be processed accordingly. Again, in order to reduce complexity we model `queryID` simply as byte with a range of only 256 distinct values. For formal specification and verification purposes, this range is adequate as it is sufficient to express the concept. The exact data structure can be determined by an implementation of the protocol.

Furthermore, PRP is designed to manage a number of security aspects such as authentication and authorisation by processing credentials in each message. However, the informal specification of JXTA intentionally leaves out precise security solutions such as encryption techniques and signatures. It rather provides a framework with placeholders for implementation of particular security solutions. Hence we generically perform a check to ensure that credentials messages are authorised.

While the ERP provides the routing primitive, the PRP enables a correct interaction between protocols. After the two core protocols, we now continue with the modelling of standard protocols, which make extensive use of the core, as expected.

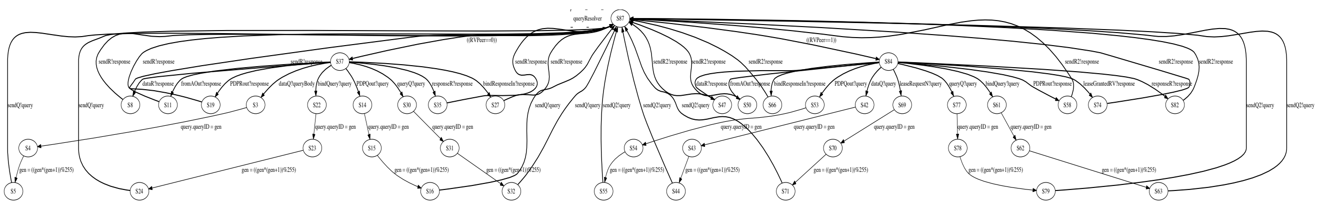

Figure 6: *Send Resolver state diagram: the large size of the model is due to the number of distinct services causes*

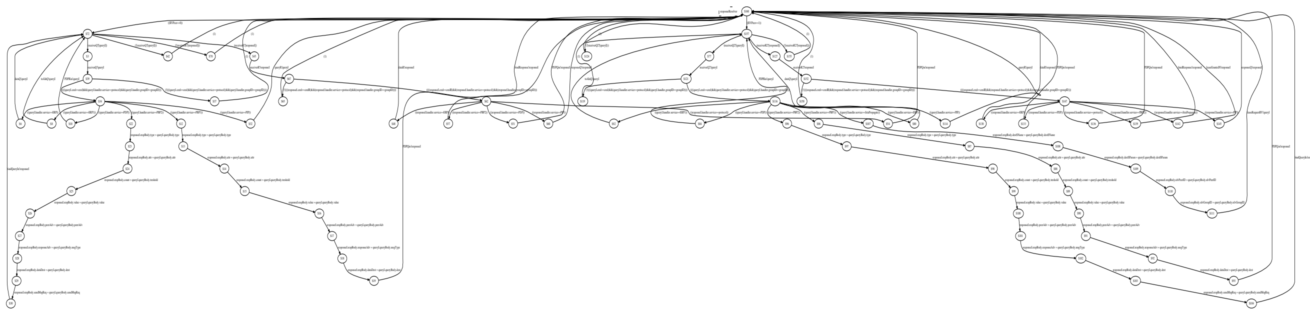

Figure 7: *Receiver state diagram: with at all named handlers required by the operation of other protocols, receiver's automaton is even larger.*

Peers use the peer discovery protocol to find published resources in the network. Based on resources advertisements, the PDP protocol explicitly distinguishes between peers, peers group and other advertisements. Other advertisements include pipe advertisements and application-specific advertisements. Although not a core protocol, the importance of the PDP protocol should be underestimated. In fact, due to the transient nature of JXTA networks where peers may appear and disappear uncontrollably, a mechanism for frequently searching for resources is of significant importance. We model the PDP protocol as two entities: a querier and a responder.

The PDP querier initiates the discovery of resources upon reception of request for a new resource. The behaviour of the querier in figure 9 is relatively simple compared to the behaviour of the discovery responder in figure 8. In this case as well, non-determinism is cause for the large size of the receiver. Indeed, because a peer may not know details of a peer but only the type of resources it seeks, the discovery protocol is capable of handling such imprecise resources discovery requests. In addition to this, to enable remote resource publication in JXTA, the responder may respond to unsolicited queries and even send a response not corresponding to any discovery query. This is exhibited in the responder's behaviour.

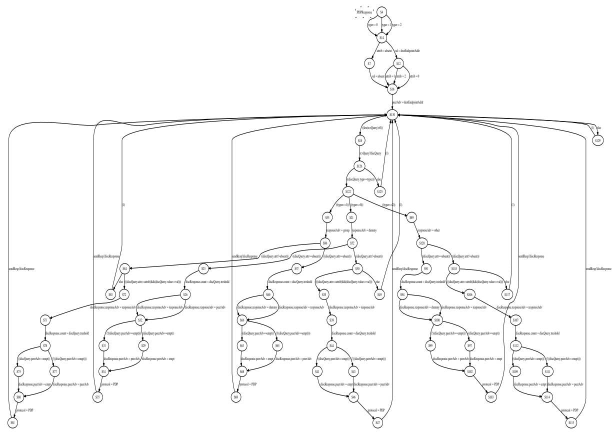

Figure 8: Discovery Responder state-diagram: because of missing information the response has the capability to non-deterministically determine data to include in the response. This increases the size of the model.

The model of the initiator of the pipe binding protocol is depicted in figure 10. This is the behaviour presented by a peer with an input pipe and using pipe binding services to bind to binding to a published output pipe. The state machine of the bind receiver is shown in figure 11. The models make abstraction of the actual communication in the pipe since this is managed by the endpoint routing protocol and network transport. Although different qualities of services can be possible with JXTA pipe we only modelled the unreliable unidirectional pipe, as it is the only one, which is generic enough to be used in all applications situations.

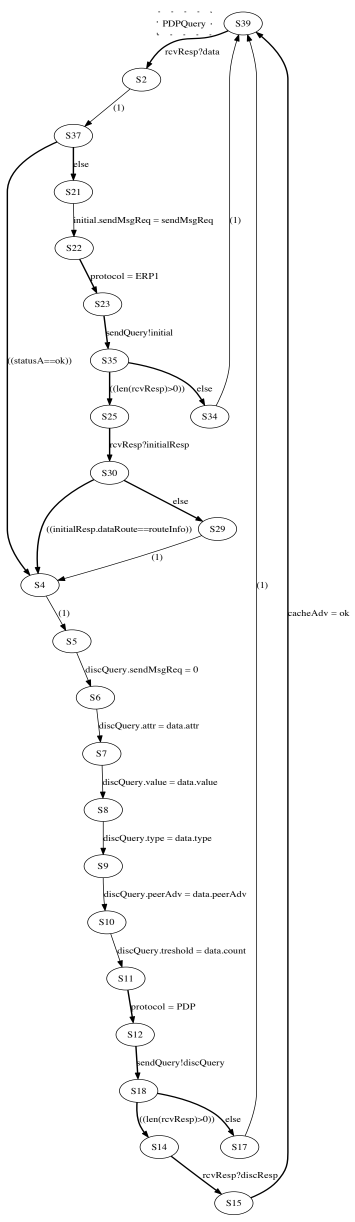

Figure 9: *The PDP querier behaviour*

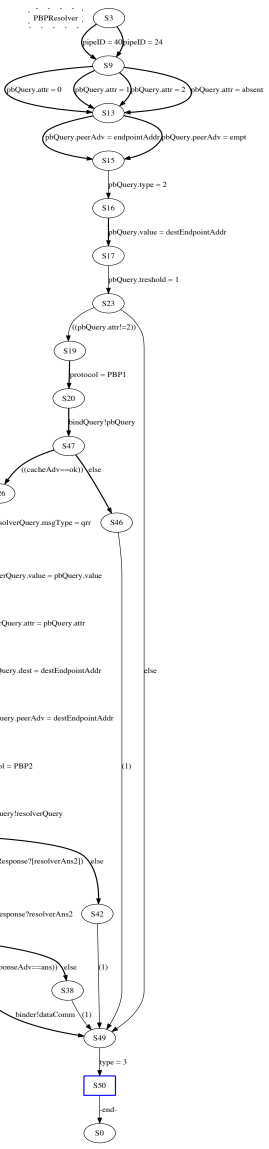

Figure 10: *State-diagram of the pipe binding resolver*

The rendezvous protocol especially leverages the peer group capability of the JXTA architecture. As mentioned in section 2, it is composed of three protocols. We model all protocols by including a normal peer, two rendezvous peers. The normal peer characterises the behaviour of a standard peer, shown in figure 13, that relies on special rendezvous peers for its participation in a peer group. The two rendezvous peers are required because only rendezvous peers can take part in the PeerView protocol, one of RVP protocols. Figure 12 presents the first rendezvous peer's automata. Not shown in this paper for brevity, the model of the second rendezvous peer simply shows its behaviour with respect to the PeerView context only.

When only the compulsory rendezvous propagation protocol is supported and the optional lease and PeerView protocol are left out, the models are simpler. Furthermore, because the RVP protocol heavily interacts with the PRP and ERP protocols, the composite model can be very large. Interaction between protocols is best showcased by the peer information protocol. At the topmost position of the JXTA architecture, the PIP protocol leverages services provided by other protocols to request and receive the status information updates from other peers. The behaviour of a requesting peer in the information protocol context is shown in figure 14. Information requested by the PIP protocol usually receives the lowest priority and is only provided when there is no security risk. After discussing JXTA protocols models, we present the results of the formal verification in SPIN next.

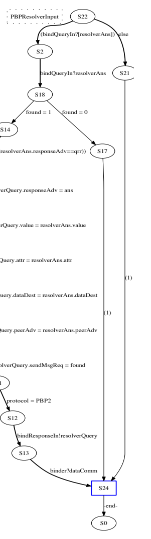
PBPResolverInput
S22
(bindQueryIn?[resolverAns]) -else
S2
bindQueryIn?resolverAns
S21
S18
found = 1
found = 0
S14
S17
((resolverAns.responseAdv==sprr))
S5
resolverQuery.responseAdv = ans
S6
resolverQuery.value = resolverAns.value
S7
resolverQuery.attr = resolverAns.attr
S8
resolverQuery.dataDest = resolverAns.dataDest
S9
resolverQuery.peerAdv = resolverAns.peerAdv
S10
resolverQuery.sendMsgReq = found
S11
protocol = PBP2
S12
bindResponseIn!resolverQuery
S13
binder?dataComm
(1)
(1)
S24
-end-
S0

Figure 11: The Input Resolver of the pipe binding protocol

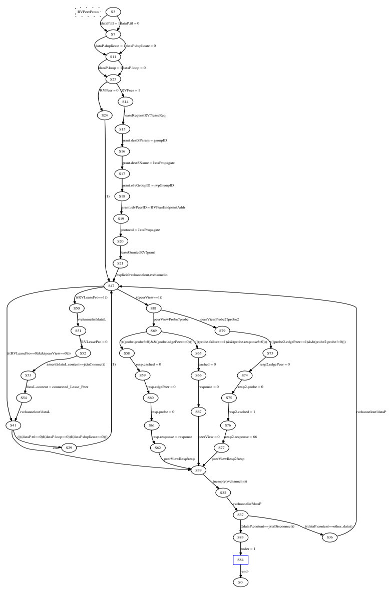

Figure 12: State diagram representing the behaviour of a rendezvous peer. All RVP protocols are included.

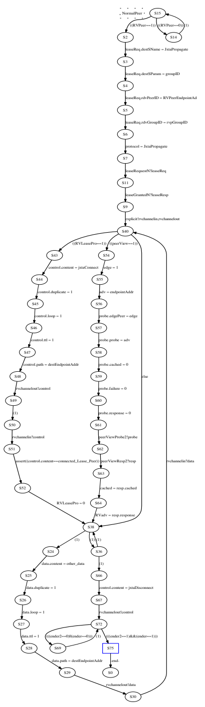

Figure 13: Finite State automata of a Regular Peer supporting the RVP's Propagation and Lease Protocols.

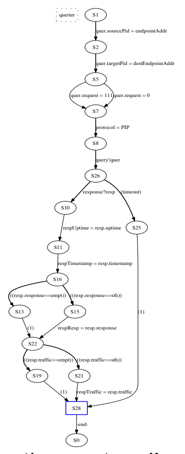

Figure 14: Information protocol's querier automata

6. MODEL-CHECKING VERIFICATION RESULTS IN SPIN

We perform automated formal verification of the integrated model of JXTA protocols. We expect the model to be very large for the reasons explained in sections 3, 4 and 5. We check for invalid end states and assertion violations in the model. These conditions are checked against the Promela description we provide to the SPIN model-checker.

The verification is performed on a personal computer with a 2.20 GHz Intel Pentium Dual Core Processor and 1.99 GB of RAM. This amount of memory caters for a state space of about 2.136 x 10^9 states when the state vector is of 1 byte. We first attempt to verify the model with an exhaustive search. However, due the size of the models we presented in section 6 and available computational resources, we expect to be faced with the state explosion problem. We thus apply state compression as well as the supertrace algorithm. Table 1 summarises the verification results we obtained.

The first line in table 1 shows the results obtained by performing an exhaustive verification of our model using the depth first search and partial reduction. Because of the size of our model, we increase the maximum size of the state-vector from 1024 (default) to 4096 bytes (-DVECTORSZ directive) and memory to be used from 1024 (default) to 2048 MB (-DMEMLIM directive). However, as anticipated, the search could not completed due to lack of memory and was interrupted at a depth of 5, 357. After applying the breadth-first search algorithm, the search could also not be completed. Expectedly the search depth was shallower but the search significantly faster than in depth-first search.

Subsequently we resorted to SPIN's built in reduction and compression algorithms. We enabled state descriptors compression through an indexing method with the –DCOLLAPSE directive. In order to minimise automata storage required to encode state descriptors, we use the –DMA=2464 directive to set an upper bound for the size of the descriptors. Although the reduction of memory requirement achieved by the application of these algorithms is very significant, this reduction comes at very steep penalty in run time. This is evidenced by the third line of table 1. In spite of the already large amount time, we cannot ascertain how much time the exhaustive verification will take to complete. Although 127,035,390 states have been examined, the results of line 3 show a non completed verification. We thus perform the approximative supertrace.

The supertrace verification is based on bitstate hashing (Holzmann, 1998) and is an approximation intended to provide a degree confidence of the correcteness of the model. Although full state search is not performed, the algorithm usually provides a very good coverage of the state space (De Renesse and Aghvami, 2004). A very important parameter that determines coverage and consequently the degree of confidence on the approximation is the **hash factor *Hf***. Its value depends on the number of reachabled states and the amout of memory available to store all states (Holzmann, 1998). The results obtained from the supertrace are shown in the last line of table 1. The run is significantly faster than the exhaustive searches performed. However, the hashing factor was **Hf = 1.78533**. This hashing factor corresponds to a state space coverage of less than 93%. SPIN recommends trusting verification results with a hashing factor greater than 100 (Holzmann, 1997) (Holzmann, 1998). Hence we cannot uphold the verification results using bitstate hashing.

The results of table 1 determine that our integrated model is too large for verification with the given computational resources and tools used. Consequently, we apply compositional verification to our model whereby we verify the JXTA architecture's components (protocols) separately. In this approach, we are able to verify internal consistency of the protocols but not the behaviour of the the JXTA network when protocols are interacting. Making abstraction of other protocols and resolving non-deterministically the possible data from other protocols, we arrive at the results shown in table 2.

Table 1: *Verification results for integrated JXTA protocols. Deadlocks and assertion violations check.*

Verification Mode	No. of States	State-vector size (bytes)	Memory usage (MB)	Depth Reached	Time (seconds)	Compile-time Parameters
Exhaustive state space search (depth-first search)	*832,199*	*2,464*	*1,848.484*	*5,357*	*278*	*-DVECTORSZ = 4096, -DMEMLIM=2048*
Exhaustive state space search (breadth-first search)	*563,309*	*2,356*	*1,846.141*	*47*	*33.3*	*-DVECTORSZ = 4096, -DMEMLIM=2048*
Exhaustive state space search with compression and reduction	127,035,390	2,464	1,849.032	267,720	250,000	*-DVECTORSZ = 4096, -DMEMLIM=2048 -DMA=2464 -DCOLLAPSE*
Supertrace/bitstate	*587,328*	*2464*	*0.781*	*1,847*	*6.74*	*-DVECTORSZ = 4096, -DMEMLIM=2048*

Table 2: *Verification of individual JXTA protocols applying compositional verification techniques. Checking for non-progress cycles*

Protocol	No. of States	State-vector size (bytes)	Memory usage (MB)	Depth Reached	Time (seconds)	Additional Parameters
Endpoint Routing	*32*[*]	*144*	*2.501*	*33*	*0.01*	*-DMEMLIM=1024*
Peer Resolver	*153,885*	*296*	*47.715*	*1400*	*0.808*	*-DMEMLIM=1024*
Peer Discovery	*3,825*	*276*	*3.477*	*57*	*0.011*	*-DMEMLIM=1024*
Pipe Binding	*238*	*636*	*2.598*	*43*	*0.004*	*-DMEMLIM=1024*
Rendezvous	*93,216*	*512*	*49.083*	*156*	*0.441*	*-DMEMLIM=1024*
Peer Information	*67*	*264*	*2.501*	*30*	*0.004*	*-DMEMLIM=1024*

For example, the verification of the endpoint routing revealed an invalid endstate at depth 32. Using the generated error trail in SPIN's simulation mode, we found that the error was due to the fact that when the route response message is dropped, the protocol did not specify how to handle message drops. However, this is explained by the fact the resolver protocol, which handles responses, does not expect to receive a response because of network conditions. This effectively shields protocols using the resolver protocol from this error. In our integrated model, this would have not been an error because the model includes protocols interactions.

In addition to the invalid endstate in the ERP protocol, non-progress cycles were detected in all protocols. In all of these cases, the non-progress cycles were related to the way these protocols attempt to protect themselves against adverse network conditions. These loops found not to progress do not have a guard conditions and indefinitely remain in those cycles. However, this behaviour is consistent with the description in (Sun Microsystems Inc. 2007) and cause invalid end states when loops are removed. The solution to this problem is to specify expiry timers, which will cause the protocols to exit the non-progress situation. This would however not be possible for the two core protocols for which the JXTA specification explicity prohibit the inclusion of time notions. Expiry timers could thus be used in higher-level services running the core protocols and in other JXTA protocols.

7.CONCLUSION

We discussed the formal model of JXTA protocols using Promela and the automated verification of the developed model with the SPIN model checker. The integrated version of JXTA protocols modelling protocols behaviour and interaction proved too large for automated verification; successively using full state search, state compression and supertrace verification. Reducing the formal specification into separate models, one for each JXTA protocol, full state space verification was possible. A number of design issues were detected. These included an invalid end state for the ERP protocol as well as non-progress cycles at for all protocols. Although the results obtained are not conclusive, they provide the basis for further work involving a higher abstraction level or full symbolic verification. With an exhaustive verification, more issues and additional improvements areas will be highlighted. These key lessons revealed by the verification will enable further development of the JXTA platform provided as contributions to the JXTA specification project. Ultimately, the platform will be improved and will become provably robust to allow adoption in critical application scenarios.

8. REFERENCES

Arnedo-Moreno, J. & Herrera-Joancomarti, J. (2009), ' Survey on security in JXTA applications', *Journal of Systems and Software*, vol 82, no. 9, pp. 1513-1525.

Clarke, EM & Wing, J.M. (1996), 'Formal methods: state of the art and future directions', *ACM Computing Surveys (CSUR)*, vol 28, no. 4, pp. 626-643.

De Renesse, F. & Aghvami, A.H. (2004), 'Formal verification of ad-hoc routing protocols using SPIN model checker', *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, 2004. MELECON 2004.*, Dubrovnik, Croatia, 9-12 May 2004, pp. 1177- 1182.

Domingo-Prieto, M., Arnedo-Moreno, J. & Herrera-Joancomartí, J. (2010), 'JXTA Security in Mobile Constrained Devices', *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, Advanced Information Networking and Applications Workshops (WAINA), Perth, Australia, 20-23 April 2010, pp. 139-144.

Gong, L 2001, 'JXTA: A Network Programming Environment', *Internet Computing, IEEE*, vol 5, no. 3, pp. 88-95.

Holzmann, G.J. (2003), *Spin Model Checker, The: Primer and Reference Manual*, Addison Wesley Professional, Boston, USA.

Holzmann, G.J. (1991), *Design and validation of computer protocols*, 1st edn, Prentice-Hall, Englewood Cliffs, USA.

Holzmann, G.J. (1997), 'The model checker SPIN', *IEEE Transactions on Software Engineering*, vol 23, no. 5, pp. 279-295.

Hossain, S., Kabir, U., Rahman, S. & Saha, A.K. (2011), 'JXTA & Web Services Using Secret Key Based Encryption', *International Journal of Computing & Information Technology*, vol 01, no. 02, pp. 71-75.

Jing, L. & Jinhua, L. (2009), 'Model Checking the SET Purchasing Process Protocol with SPIN', *5th Wireless Communications, Networking and Mobile Computing Conference*, Beijing, China, pp. 1-4.

Lamsweerde, A.V. (2000), 'Formal Specification: a Roadmap', *Proceedings of the Conference on The Future of Software Engineering*, ACM, New York, USA, pp. 147-159.

Merz, S. (2001), 'Model Checking: A Tutorial Overview', *Proceedings of the 4th Summer School on Modeling and Verification of Parallel Processes*, London, UK, pp. 3-38.

Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S. & Xu, Z. (2003), 'Peer-to-Peer Computing', Technical, HP Laboratories, Palo Alto.

Spaho, E., Matsuo, K., Ogata, Y., Xhafa, F. & Barolli, L. (2010), 'Application of a JXTA-Overlay P2P Control System for a Biped Walking Robot', *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, Fukuoka, Japan, pp 719-724.

Sun Microsystems Inc. (2007), 'JXTA v2.0 Protocols Specification', Technical Report, Sun Microsystems Inc., Santa Clara.

Velipasalar, S., Lin, C.-H., Schlessman, J. & Wolf, W. (2006), 'Design and Verification of Communication Protocols for Peer-to-Peer Multimedia Systems', *2006 IEEE International Conference on Multimedia and Expo*, Toronto, Canada, pp. 1421-1424.