Abstract— JXTA protocols describe a platform for peer-to-peer networking, independent of underlying transport protocols, for peer collaboration and message exchange. The JXTA project is an open source initiative and its protocols are informally specified as an exchange of XML-based messages between entities. This paper presents the formal specification and automated verification of its core protocols: the Endpoint Routing Protocol and the Peer Resolver Protocol. These protocols run at the core of the JXTA architecture and are respectively responsible for routing messages between peers and for resolving queries/responses for services and applications. The PROMELA based formal specification serves as input to the SPIN model checker to prove the presence of a logical flaw due to non-progress cycles.

Index Terms— formal specification, formal verification, model checking, networking, peer-to-peer.

I. INTRODUCTION

The peer-to-peer paradigm represents a credible alternative to centralised and client-server approaches in networking and computing. Its benefits include improved scalability, resources aggregation, fault resilience, dynamism, absence of a single point of failure and lower cost of ownership. Based on the premise that the world will be increasingly connected and widely distributed, the centralised and client-server models will become progressively inefficient [1]. In addition to inefficiency, centralised systems are exposed to bottlenecks and present a single point of failure vulnerability. However, despite the benefits of the peer-to-peer model, its deployment has not been as pervasive as it was envisioned to be.

Peer-to-peer approaches are still mainly limited to casual instant messaging and file sharing applications [1][2]. The examples of Napster, Bit Torrent and KaZaA are illustrative. Critical, serious and dependable applications and infrastructures do not make use of peer-to-peer networks. As mentioned in [1], among issues faced by the peer-to-peer model, are acceptance and use, security and interoperability. However, steady progress in distributed computing has allowed peer-to-peer networking to be seriously considered beyond the casual file sharing systems. For example, P-Grid [3] and JXTA (for “juxtapose”) [4] provide a platform for building application ranging from collaborative systems to resources sharing.

Specifically, the JXTA specification denote a set of six protocols designed to be a framework for the development of more specialised, robust and reliable peer-to-peer applications. These protocols are designed to be generic enough to enable implementation in any programming language, operating system or network transport. They intend to standardise peer-to-peer networking functions such as routing, peer advertisement and discovery and self-organisation. JXTA protocols can thus allow ad hoc, pervasive and multi-hop peer-to-peer networking in almost any application scenario. Initiated by Sun Microsystems, the JXTA project is a promising open source effort with wide applications potential [5][6].

JXTA protocols are informally specified as exchanges of a set of XML-based messages between entities of the protocols. Various implementations of JXTA protocols are available. These can be used to demonstrate the soundness and reliability of JXTA under specific circumstances. However, these protocols have not been subjected to formal methods such as formal specification, verification and validation. With the use of formal methods, the correctness, completeness and dependability of the protocols can be established even for unforeseen application scenarios. Indeed, formal methods have allowed uncovering severe flaws and inconsistencies in the design of software and protocols; inconsistencies and flaws previously undetected by simulations runs and implementations [7][8].

The purpose of this paper is to present the formal specification and verification of JXTA’s core protocols: the Endpoint Routing Protocol (ERP) and the Peer Resolver Protocol (PRP). They respectively provide routing information and a mechanism for peers to send and receive queries. At the core of the JXTA specification, these two protocols are critical to the operation of a JXTA network because a peer needs to implement them to be addressable. The rationale behind the formal specification and verification of these protocols is to find possible design errors, logical flaws and improvement areas. With respect to peer-to-peer networking, the novelty of this approach revolves around applying an exhaustive state space search to the core protocols specification and laying out all possible system states.

A transition-based specification language, PROMELA (Protocol Meta Language), is used since transition-based specifications are based functions of transitions between system states to specify properties and behaviour of the protocol. Furthermore, the Simple PROMELA Interpreter (SPIN) provides the infrastructure for formal and exhaustive verification. The work presented in this paper contributes towards establishing the peer-to-peer paradigm as a robust and dependable alternative to traditional models.

The rest of the paper is structured as follows. Section II
provides an overview of the JXTA platform, its protocol suite and its core protocols. In section III, we describe formal specification, formal verification, PROMELA and SPIN in order to understand their relevance and suitability to the peer-to-peer model. Section IV presents the formal specification of JXTA’s Endpoint Routing and Peer Resolver protocols. Subsequently, the discussion on the results of the formal verification and possible improvement to the protocols is provided in section V. Finally, section VI presents concluding remarks.

II. THE JXTA PROTOCOL SUITE

A. 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 [9]. JXTA intends to enable the development and deployment of interoperable services and application, thus “spring-boarding the peer-to-peer revolution on the Internet” [9]. All design considerations in the JXTA framework are governed by 3 essential objectives [4]:

• **Interoperability**: JXTA-based services should be able to seamlessly interact with each other in order to create a larger peer-to-peer community. This approach breaks away with initial peer-to-peer services such 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, network transport or operating system it is implemented in.

• **Ubiquity**: JXTA technology should be able to run on any digital device ranging from a simple sensor or network switch to a supercomputer.

Based on these objectives, the JXTA platform makes use of seven conceptual elements central to the behaviour of the platform [9]: identifiers, peers, peer groups, pipes, advertisements, credentials and messages.

A JXTA **Identifier (ID)** uniquely identifies entities and resources in JXTA in such a way that they can be referred to unambiguously and canonically. It is intended to be opaque: the context in which it is used in a protocol allows inferring its type.

A **peer** is any networked devices running at least JXTA’s core protocols. A peer can be a mobile phone, smartphone, laptop, sensor, desktop computer, etcetera.

A **peer group** is a set of peers with common goals and interests. Peer grouping is the most powerful characteristic of JXTA and is embedded in almost every aspect of its architecture. In fact, all peers join the netPeerGroup and WorldGroup peer groups at start up. Peers can self-organise in peer groups, which can be seen as logical groupings used to restrict access to resources.

A **pipe** is the fundamental communications means in a JXTA network. Pipes are virtual communication channels used by peers to exchange data. Pipes can implement a number of qualities of service: unidirectional asynchronous, synchronous request response, bulk transfer, streaming and secure. Though these are possible, JXTA protocols require and are based on unidirectional asynchronous pipes with no guarantee of delivery to function. In fact, this mode of communication is often the only mode available in many forms of wireless networking. This choice thus allows JXTA protocols to be as pervasive as possible and easy to implement in any network transport.

Advertisements are neutral metadata structures used by JXTA protocols to describe all resources, including peers. They contain the ID and all required information of the resource they describe.

Messages are the basic data exchange unit in JXTA. Protocols and peers interactions make use of messages to exchange data and communicate.

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.

B. The JXTA protocol suite

JXTA defines a set of six protocols designed for pervasive, ad hoc and multi-hop peer-to-peer network computing [9]. The protocols are specified to be independent and optional in the operation of the network with the exception of core protocols that are compulsory.

A succinct description of each protocol follows:

• **The Endpoint Routing Protocol (ERP)** is responsible for routing between peers.

• The **Peer Resolver Protocol (PRP)** provides a generic query/response interface for applications, services and protocols. The PRP and ERP protocols constitute the core layer of JXTA. A more elaborated description follows in subsection C below.

• **The Peer Discovery Protocol (PDP)** enables resource publication with the dissemination of advertisements.

• **The Peer Information Protocol (PIP)** enables each peer with status information regarding other peers.

• **The Peer Binding Protocol (PBP)** facilitates the establishment of virtual communication between peers by linking or binding the two ends of a pipe to specific endpoint Addresses.

• **The Rendezvous Protocol (RVP)** aids with the controlled propagation of messages.

C. The core protocols: ERP and PRP

The Endpoint Routing Protocol is responsible for the routing of messages between source and destination peers. It details routing procedures as a set of request/query messages exchange and determines routing information processed by a routing service. To achieve this, the protocol relies on some special peers called peer routers.

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 thus provide low-level infrastructure for the basic routing in a JXTA network. They are particularly important because of the ad hoc, multi-hop and adaptive nature of the network. Further, connections in the network may be unstable and routing of message is nondeterministic.

The operation of routing is as follow: when a peer receives a request to send a message to a given endpoint
address, it checks for the route in its cache. If it does not have the route, it requests the route information from its peer routers with a query resolved by the PRP protocol. The route information is sent back to the requesting peer if a route is available or when it is discovered. When the route is received from the peer router or if it is available in the peer’s cache, the peer sends the message. Besides sending a message, a peer can be requested to forward a message as a relay. The operation is the same but there is an additional message trace that is appended to the message to detect loops, discard recurrent messages or to add new route at peer routers’ cache.

On the other hand, 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 processed or handled.

A query Resolver message is sent from a query-originating peer and is resolved by the named handler specified in the query. A response resolver message can later be sent in response to the query. In order to match responses with the relevant query, the resolver protocol makes use of query identity numbers. For example, it is the PRP protocol that matches the response to the specific new route query that a peer made earlier. Further, at reception of a resolver message, the protocol performs access rights and authorisation based on the credentials contained in the message. In fact, security in JXTA is based on membership to a group. 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 or resource (represented by a named handler in PRP) and confirmation of the identity beyond the simple peer ID.

III. FORMAL SPECIFICATION AND VERIFICATION

Protocols specifications are often written in natural language, which often leads to ambiguity, incompleteness, imprecision and lack of clarity [10]. Furthermore, these specifications are often difficult to completely test, debug and improve. JXTA is not an exception to this fact since it is specified in English [9]. However, with the application of formal specification and other formal methods, design errors and system flaws can be detected in a timely and improvements more easily made [7][8][10].

Formal methods deal with the mathematical modelling of software and hardware systems. With their mathematical basis [11][12], formal methods allow to characterise software and hardware systems in a very precise, unambiguous and consistent way. Formal methods further facilitate the validation of the developed system models with mathematical precision, thereby proving or disproving the presence of particular properties and behaviour. Particularly, formal specification involves characterising software and hardware systems in a formal language while formal verification comprises formal manipulation of the formal specification to ascertain the existence of specific properties in the system. Consequently, while an implementation of a protocol is done at the lowest level of abstraction with the smallest details in order to provide a specific service, a formal specification is performed at a higher level of abstraction to model protocol properties at system level.

Peer-to-peer systems, like all other communication systems and protocols are classified as reactive systems as opposed to transactional systems for instance. Reactive systems are systems whose behaviour changes in response to external and internal stimuli, with concurrency and distribution as important factors. Peer-to-peer systems present both distribution and concurrency features due to the spatial separation of its subcomponents (peers and protocols execution) and of the sharing of resources, respectively [7]. The consequence of this characteristic in relation to formalisms is that concurrent and distributed systems are commonly described by transition systems [8]. Furthermore, temporal logic constitutes an excellent logic for expressing the properties of transition systems [13].

Reactive systems are typically formalised by modelling languages such as Temporal Logic of Action (TLA+) or PROMELA as well as process algebra or Petri Nets [14]. This is due to the fact that operational semantics of these formal methods are stipulated in terms of transition systems [8]. In addition to this, these modelling languages and their corresponding model checkers are optimised for specific types of systems such as synchronous shared-variables programs or asynchronous communication protocols [15]. Model checking has therefore been extensively applied in the formal analysis of software and communication systems [8][15][16].

Lu, Merz and Weidenbach [17] specify the core routing algorithm of Pastry in TLA+ and verify the model for consistency using the TLA+ model checker (TLC). In [18] the authors proposed the verification of the correctness of the stabilising algorithm of the Chord peer-to-peer overlay network using process algebra. In [19], the researcher presents the formal verification of an autonomous and wireless peer-to-peer auction handling system. The correctness and consistency of the system were established using SPIN after modelling the specification in PROMELA. Our approach also uses PROMELA, a transition-based specification language. PROMELA, based on automata theory [20] and the input language of the SPIN model checker, is typically used for specifying or modelling communication protocols.

PROMELA has a simple structure for data objects with high expressive power in relation with the level of coding required. In terms of usability and communicability, the similarity between PROMELA and C as well as the level of abstraction achievable makes PROMELA a good choice for formal specification in software engineering. Furthermore, non-determinism and the use of communication channel between processes are strong features of PROMELA. With the non-deterministic and unpredictability nature of communications in peer-to-peer networks and PROMELA’s innate ability to model communication channels, the language is the most sensible choice for modelling JXTA protocols.

The formal verification process depends on the check implemented on the model; the check is done through an
exhaustive state space search. It is therefore sufficient to find a counterexample to disprove a property and thereby uncovering design errors. SPIN optimises its search algorithms (depth-first search, breadth-search) by using partial order reduction to reduce the number of system states, or state compression to decrease the amount of memory required to store each system state [21]. In each case, the full state search is completed without affecting the results. Besides, SPIN provides the possibility to trade-off memory requirement for needed runtime.

IV. PROMELA-BASED FORMAL SPECIFICATION OF ERP AND PRP PROTOCOLS

As mentioned in section III, PROMELA is based in automata theory. Consequently, PROMELA-based specifications can be represented as finite-state machines. Finite-state machines provide an excellent mathematical abstraction in the design and analysis of computer systems such as protocols. In addition to that, when represented as a state diagram, analysis of the behaviour of the system under study becomes simpler. Hence instead of PROMELA code, we present the behaviour of protocols entities as finite-state machine. The automata in the machines are non-deterministic as dictated by the nature of the communications protocols we are specifying.

The informal specification of the endpoint routing and peer resolver protocols in [9], described in section II, constitutes the basis of the PROMELA-based formal specification presented here. In fact, JXTA protocols are informally specified in the English language as an exchange of XML-based messages. However, before we proceed with the formal specification of the core protocols, a number of key assumptions and initial conditions need to be made. These assumptions are necessary to make the formal specification feasible while retaining all of the protocols behavioural or systemic attributes.

For the endpoint routing protocol, we assume that a peer does not have any route and knows of at least one peer router (can be pre-configured). In fact, it is often the case that a new peer does not have any knowledge of the current state of the network. In these conditions, the ERP protocol would be inoperable without the existence of peer routers. We further assume, as recommended in [9], that connections are transient and only unidirectional pipes are available, with no guarantee of delivery. Thus the PROMELA specification for the ERP protocol contains 2 processes modelling each protocol participant: a served peer and a peer router. In addition to that, the lower layer, for communication between the two entities is also added.

The automaton describing the behaviour of the served router is left out due to its large size, which stems from the complexity of the model. However, the behaviour of the peer router is shown in Figure 1. The state diagram of the model of the communication layer is shown in figure 2. The specifications of the peer router and the served peer start with a sequence that sets all initial conditions and addresses. The exchanges between the protocol entities (processes or models) are possible through the exchange of query and response message types. The lower layer, providing the communication infrastructure between the entities, is characterised by the possibility of message drop without any recovery mechanism. Correct message delivery or message drop for each type (query or response) of receive operation is decided non-deterministically. The message drop is formalised as the always-executable skip statement because the informal JXTA specification does not determine how the event should be handled.

At reception of a route request the peer checks its cache for route and returns an ‘ok’ status message to the requesting protocol, informing it that a route is available if the route is not obsolete. Otherwise, the peer sends (via the unreliable communication channel) 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.

A number of scenarios are then possible:
- The peer router could not find a route, an ‘err’ status message is returned to the requesting protocol to indicate that no route was found.
- The peer router found a route but was either obsolete or its time to live has elapsed. The route request is then repeated. Message obsoleteness and the time-to-live factor are non-deterministically resolved for each route request.
- The peer route found a viable route, the peer adds the route to its cache and returns an ‘ok’ status message.

In case of a message forward request, the message is discarded (status message discard), received (status message recv), 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. Here the message route is known and contained in the message forward request.

The behaviour of the peer router is less complex than that of the requesting peer as evidenced by the relatively simple structure of the state machine in figure 1. On reception of a route request, the peer router searches its cache for the route information. A response is sent to the requesting peer in form of either the route information or message corresponding to the absence of route information.
The specific data type or structure can be determined by an from state S3 to state S4 above represented by its state diagram is depicted in the lower communication layer model (Figure 2).

Therefore the need to be sent by a peer, assigns to it a lower communication layer suffices to model the behaviour of communication channels.

The finite-state machine of the sender resolver, represented by its state diagram is depicted in figure 3. This entity of the protocol resolves all queries and responses to be sent by a peer, assigns to it a queryID for queries or retrieves the queryID of the query being responded to and sends it to the corresponding receiver resolver at another peer. For queries, a new queryID is generated after the initial queryID is used. This is shown in the transition from state S3 to state S4 above. The queryID and gen variables are represented as bytes for modelling purposes. The specific data type or structure can be determined by an implementation. The receive resolver in figure 4 models the behaviour of named handlers.

At reception of a resolver message, the receive resolver entity performs a check to ensure that the named handlers specified in the message is registered. Also, the entity checks the compulsory credential contained in the resolver message to establish the identity of the sender and to ensure that the sender is authorised to send such a message. Subsequently, the protocol passes the message to the intended registered named handler, which processes it accordingly. In the case of the ERP protocol for instance, route requests are passed to the peer routers while route responses are delivered to the requesting served peer. Besides, it is important to note that the PRP protocol does not expect a response to a query nor does it expect the query to arrive at destination error-free. Only a best effort policy is maintained. This is an important fact that influences the results of the formal verification we discuss next.

V. RESULTS OF FORMAL VERIFICATION

As core protocols, the ERP and PRP protocols are critical to the operation of a JXTA network. They are to provide a fundamental routing and query/response resolution primitives to the network. It follows that their consistency or lack thereof greatly impacts the behaviour of the JXTA platform; if either one of them fail, JXTA platform’s performance will be compromised. We thus input the formal specification presented in the previous section to the SPIN model checker.

State-vector 348 byte, depth reached 1400, errors: 0
153899 states, stored
330575 states, matched
484474 transitions (= stored+matched)
5 atomic steps
hash conflicts: 23630 (resolved)
pan: elapsed time 1.71 seconds
pan: rate 89999.415 states/second
Listing 1: SPIN Verification results for deadlocks and assertions violations

To be more specific, we formal verified the protocols (or the models characterising their behaviours), checking for deadlocks, livelocks, and assertion violations. As seen in listing 1, when checking for deadlocks (invalid end states) and assertion violations, the verification run reached a depth of 1400 with a state-vector of 348 bytes. The exhaustive state space search, using the depth-first search algorithm with partial order reduction, examined 153,899 different states and found no invalid end states and no assertion violations. This verification run took only 1.71 seconds to complete. Next, we checked for livelocks or non-progress cycles.

State-vector 307 byte, depth reached 34, errors: 1
15 states, stored
0 states, matched
15 transitions (= stored+matched)
5 atomic steps
hash conflicts: 0 (resolved)
pan: elapsed time 0.043 seconds
pan: rate 348.837 states/second
Listing 2: SPIN Verification results for non-progress cycles

With a state-vector of 304 bytes and again with exhaustive state space depth-first search with partial order reduction, the verification only reached a depth of 34 and examined only 15 states. This was due to a non-progress-cycle found at depth 31. Indeed, because the PRP protocol is specified to continue running in a loop even when a response to a query is never received and does nothing about it, the run is found not to progress. This behaviour is specified to protect the protocol from an invalid state in case of message drop. In other words, the receive resolver should not wait for specific messages as they may be dropped; but
rather continue listening from other queries or responses. When this guard is removed and the verification run again, there is a deadlock due to receiver waiting for a message that has been dropped by the communication channel.

At first, a solution might be to include an expiration timer so that the protocol may progress but the JXTA specification explicitly prohibits the inclusion of any timing facility in the core protocols. It rather recommended including the timing in the higher-level services or applications running core protocols. Hence to solve the problem flagged by the verification, it is imperative that higher-level protocols, services and applications specify an expiry time after which they can consider a query or response lost. They could thereafter specify recovery mechanism for this instance.

VI. CONCLUSION

We have presented the formal specification and verification of core protocols of a JXTA peer-to-peer network. Using PROMELA to specify the protocols, we presented the specifications as state diagrams of finite-state machines characterising the behaviour of each entity. In the ensuing formal verification in SPIN, we discovered a non-progress cycle the receive entity of the PRP protocol. However, the problem can be solved with the inclusion by higher-level services and protocols of a timer in absence of a response or in case of routing divergence. Further work includes the formal specification and verification of the remaining JXTA protocols and inclusion of recovery mechanism when the core protocols fail due to the non-progress cycle.

REFERENCES


Yannick Kala-Konga received his bachelor's degree in electrical engineering cum laude in 2010 from the Tshwane University of Technology and is presently studying towards his Master of Technology degree at the same institution. His research interests include peer-to-peer networks, formal methods, wireless communications and radio planning and optimisation.