JXTA anonymity through a replicated message-based approach

Àlex García-Domínguez, Marc Domingo-Prieto, Joan Arnedo-Moreno
Internet Interdisciplinary Institute (IN3)
Universitat Oberta de Catalunya
Barcelona, Spain
lex, mdomingopr, jarnedo@uoc.edu

***Abstract*—JXTA is a mature set of open protocols, with more than 10 years of history, that enable the creation and deployment of peer-to-peer (P2P) networks, allowing the execution of services in a distributed manner. Throughout its lifecycle, it has slowly evolved in order to appeal a broad set of different applications. Part of this evolution includes providing basic security capabilities in its protocols in order to achieve some degree of message privacy and authentication. However, under some contexts, more advanced security requirements should be met, such as anonymity. There are several methods to attain anonymity in generic P2P networks. In this paper, we propose how to adapt a replicated message-based approach to JXTA, by taking advantage of its idiosyncracies and capabilities.**[1]

Keywords: peer-to-peer, security, anonymity, JXTA, Java, replicated message, multicast.

I. Introduction

JXTA [1] is a technology that enables the deployment of peer-to-peer (P2P) applications, allowing a set of heterogeneous devices, ranging from cell phones and wireless PDAs to PCs and servers, to form groups and collaborate in a self-organized manner, regardless of the underlying network. Peers may freely interact even when some of the peers and resources are behind firewalls and network address translations (NATs) or rely different network transport methods. Created by SUN in 2001, JXTA has iterated through successive revisions during its 10 years of history, slowly gaining popularity, with over 2,700,000 downloads and more than 120 active projects. Its latest Java version, JXTA 2.7 [2], became available in March 2011. The main highlights include some long awaited basic security improvements, such as secure peer groups and advertisement signatures.

Even though JXTA can be considered a mature and thoroughly tested technology, being used in projects which range from robot control applications to systems management or real-time collaboration platforms [3], [4], there is still a long way to go as far as security is concerned. As P2P systems evolve and are used in new scenarios, more advanced security capabilities become important. An example of them is message anonymity [5]. Even though the concept of anonymity in P2P networks is often associated with situations where exposure has very strong implications, such as maintaining the right to free speech or circumventing legal responsibilities [6], there are also everyday situations in which nobody is trying to avoid ending up in jail. Anonymity also provides a way to increase the users' trust in the system and encourage participation in more mundane activities such as a corporate suggestion box, a small ballot system or a peer evaluation form.

Anonymity in P2P networks may be achieved through different methods [7]. With the objective of providing JXTA with a full suite of anonymity protocols which may cater to a broad set of scenarios, in our previous work we proposed and studied different approaches, such as unimessage and split message ones [8], [9]. The former can be considered the most popular ones, since they provide the highest degree of anonymity, whereas the latter are quite effective in very dynamic environments, where peers often go offline and then back online. In this paper, we assess how a third different approach, a replicated message-based one, may be adapted to the idiosyncrasies of a JXTA-based service.

Even though replicated message-based approaches are not very recent, we deem it interesting taking them into consideration since they rely on very different methods to transmit data, mainly broadcast and multicast, decreasing the delay of communications. Generally, anonymity is achieved by sending the message through some peers, adding delay. Replicated message-based approaches add a huge overhead, since messages are sent to a group of peers not just one. But in networks with a low message rate, using this technique may really decrease the amount of time required to receive a the response from a query. Also, adapting this protocol to JXTA is interesting, since JXTA integrates peer groups as well as broadcast and multicast mechanisms into its core architecture.

The paper is structured as follows. In Section II, we provide a thorough review of replicated message-based approaches to anonymity within P2P systems. Section III describes our proposed adaptation of a replicated message-based protocol to the specifics of JXTA. We also discuss some protocol tweaks and improvements on the original proposal. In Section IV we provide some results about the processing time requirements of the protocol in a JXTA network, based on experimentation. Finally, Section V concludes this paper and outlines further work.

[1]This work was partly funded by the Spanish Government through projects TSI2007-65406-C03-03 E-AEGIS, TIN2011-27076-C03-02 CO-PRIVACY and CONSOLIDER INGENIO 2010 CSD2007-0004 ARES.

II. Related Work

Several surveys on anonymity on unstructured P2P networks exist [10], [7], each one proposing a different taxonomy to categorize current approaches. Our work is based on the latter, being the most recent one. According to the author, approaches can be divided into three categories: unimessage, split message and replicated message.

On one hand, in a unimessage approach, the anonymous message travels across the network, usually encrypted, as a single entity. An important subset of this category, path-based approaches and its variations [11], are the most popular in peer-to-peer anonymity. On the other hand, a split message-based approache divide the message into several pieces which travel independently of each other towards the destination. Reconstruction usually relies on threshold systems [12]. Finally, in a replicated message one, the message is not divided, but multiple encrypted full replicas are spread across the network, taking advantage of multicast or broadcast capabilities, so it is not easy to pinpoint the actual destination.

In this section, we focus on the latter approach, since it is the chosen mechanism for our proposed anonymity layer, so it is possible, given our previous work, to provide a full set of anonymity protocols to JXTA services. The main strength of the kind of protocols under this category, in contrast with the other two categories, is its fault tolerance, requiring less collaboration between peers to successfully transmit messages. In addition, they are more resistant to collusion between rogue peers or traffic analysis by global attackers.

A generic approach which relies on the basics of this category can be traced back as far as [13], where multicast-based communications are used as a way to anonymously transmit data. In this proposal, all participants are structured using a spanning tree, where each entity assumes the role of a single node. Data is iteratively forwarded across the tree, after being processed using a xor operation in such a manner that only the actual receiver will get the original data.

Nevertheless, the basic ideas have been later adapted to P2P networks. One of the main contributions can be found in P^5 [14]. Peers are divided into a structure of hierarchical channels, according to the result of applying a hash algorithm on each peer's public key. Such hierarchy is implemented as a unicast tree where channels in the upper tiers include all peers in lower tier channels, the root node encompassing all peers in the networks. Therefore, the lower in the structure a channel is, the more efficient it is at broadcast transmission, but at the cost of lowering the anonymity set. Messages are propagated across channels in a manner that peers may chose their own trade-off between efficiency and anonymity. As an additional feature, peers keep a fixed transmission rate of noise packets, so it is not possible to assess when an actual valid transmission has just started.

Another proposal, that specially focuses on receiver anonymity, can be found in [15]. The authors argue that sender anonymity is the most favored scenario in current proposals, and more efforts should be made on receiver anonymity. Using the quite straightforward method of relying on multicast groups to transmit encrypted messages, it is guaranteed that the actual receiver is the only one able to decrypt and process the message. Nevertheless, its main feature is the use of what the authors call an *Incomparable Public Key* cryptosystem. In this scheme, a secret key can be linked to a set of public keys, instead of to a single public key, but, given two random public keys, it is impossible to decide whether both belong to the same set or not. Thus, colluding senders cannot share data to guess whether they are sending messages to the same destination, thus providing additional receiver anonymity.

Taking some ideas from the former proposals, in [16], a multicast tree is constructed to transmit messages. However, in this case, the hierarchy is generated in a completely decentralized and self-organized manner. In addition, the resulting structure is heavily based on the actual network topology and connections of peers, instead of on the result of a hash algorithm, thus increasing efficiency.

Hybrid approaches also exist, such as Hordes [17], an evolution of Crowds [18], which combines the features of path-based and replicated message protocols. Message requests are sent using probabilistic forwarding, whereas responses entirely rely on multicast transmission, as summarized in Figure 1. Access to the anonymous network, or *horde*, is managed through a set of servers which provide the base multicast address and the addresses of other horde members. Differently to other approaches, peers do not join to the same multicast group for their whole connection lifecycle, but whenever a sender transmits a message, it subscribes to a random multicast group, where he will wait for the response. In this scenario, given that the reply path will be the shortest one to the original sender, response time is improved.

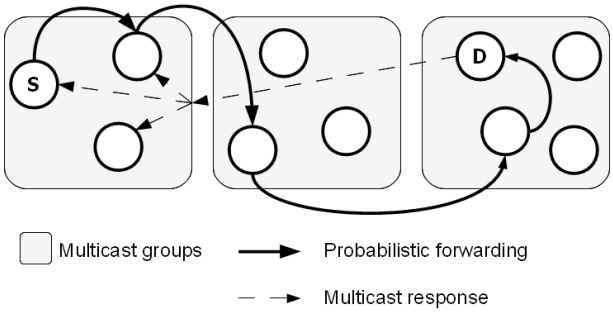

Fig. 1. Hordes hybrid approach (S: Sender, D: Destination)

We have chosen the Hordes protocol as the basis for our work, since it provides a nice trade-off between simplicity and efficiency, taking full advantage of multicast communications. Furthermore it is one the few proposals in this category where the authors explicitly quantify the anonymity of their protocol. In fact, from this assessment, it is concluded that the protocol is quite complementary to unimessage protocols, being able to withstand attacks which can be successful on them and vice-versa. Therefore, we consider it is an interesting alternative to consider in a JXTA anonymity suite layer, given our previous

work up to date. Depending on the kind of attacks a developer is more concerned about, he will be able choose the most convenient anonymity approach.

III. A REPLICATED-MESSAGE PROTOCOL FOR JXTA

JXTA core primitives provide application developers with the means to easily deploy services and exchange messages between them, independently of the services' final purpose. In order to propose an anonymizing mechanism for JXTA which takes advantage of these core capabilities, it is important to review the most important characteristics of its architecture. From this study, it is possible to propose an adaptation of the Hordes protocol as an anonymity layer which meshes with the mechanisms JXTA provides to deploy services.

A. JXTA architecture overview

The main idiosyncrasy in JXTA's design, which sets it apart from other P2P frameworks, is introducing the concept of *peer group*, a segmentation of the global JXTA network. All peers publish and consume services within the context of a group, interacting with each other by using JXTA's core services, the most important ones being the *Membership*, *Discovery* and *Pipe* services. Only peers within the same peer group may interact.

The Membership Service allows joining a peer group and claiming an unique identity within the group's context. Through this service, each group member is provided with a credential, which may be used at any time to authenticate to other group members. Different implementations exist depending how such identity is claimed and the credential format.

The Discovery Service manages group resources publication and discovery. Every resource in a JXTA group is described by an *Advertisement*, an XML metadata document. A resource cannot be accessed unless its corresponding Advertisement is previously retrieved. It is the Discovery Service's responsibility to manage and distribute Advertisements, since a resource is not considered available unless its Advertisement is periodically published.

There are several Advertisement types, but the most important ones are:

- *Peer Advertisement*: Describes a peer and the resources and services it provides, under a special service parameter entry. Each peer is responsible for the publication of its own Peer Advertisement, and a peer is only considered online while he continues to do so.
- *Pipe Advertisement*: Describes a *JXTA Pipe*, an abstract communication channel, usually deployed by a service provider to receive queries. It is the main mechanism to exchange data between applications.

Finally, the Pipe Service is responsible for managing message exchanges using JXTA Pipes. The simplest pipe in JXTA, the *JxtaUnicast* type, provides an asynchronous, unidirectional message transfer mechanism which can be easily established and managed. Nevertheless, there is a higher-level communication abstraction, the *JxtaBiDiPipe*, which provides a bidirectional communication channel. The latter is usually preferred by services, since it allows a straightforward query-response message exchange. The description of JXTA's standard service model based on this procedure follows:

1) Each service provider starts a *JXTAServerPipe* using the Pipe Service, which makes available and listens to an *input pipe* in order to process inbound communication requests. This input pipe is defined using a Pipe Advertisement.
2) The service provider publishes the Pipe Advertisement to other group members using the Discovery Service.
3) The Advertisement is propagated within the group by *Rendezvous Peers*, special super-peers who efficiently distribute Advertisements.
4) To consume a service, a peer also uses the Discovery Service to retrieve the Pipe Advertisement. Then, a connection is established via the Pipe Service and the consumer may begin sending messages.
5) Once a message is received at the server side, the results depend on the pipe type, JxtaUnicast or JxtaBiDiPipe. In the former case, messages may be processed, but no response is possible. On the latter case, a linked outbound communication channel is created and two-way exchanges are made possible.

JXTA messages sent through pipe connections follow a predefined structure comprised of a set of name/value pairs labelled under a namespace and organized as an ordered sequence. As a message passes down each JXTA layer, one or more named elements may be added to the message (for example, control data). Their order within the message structure always follows the same order they were added. As a message is processed back up the stack, each layer will remove these elements, until only application data remains.

Message exchanges can be secured in JXTA by using a group based on the *PSE (Personal Security Environment)* Membership Service implementation. Under this kind of peer group, each peer is provided with a credential based on public key cryptography. This guarantees that each peer has initialized a valid pair of public-private keys and that the public key of each peer is automatically distributed inside its Peer Advertisement, in a special service parameter entry.

B. Anonymizing procedure

From JXTA's architecture overview, we propose an anonymity layer that causes the minimum interferences on the JXTA messaging layers. The chosen approach to provide anonymity capabilities to JXTA services through the Hordes' protocol is based on the deployment of an anonymizer service at the JXTA Community Service layer. The anonymizing service works within the context of a peer group, meaning that only peers from the same peer group may exchange anonymous messages (as is the case for any other JXTA service). The group becomes equivalent to the *horde* concept of the Hordes protocol.

Each peer is free to deploy the service, or not, and it is not assumed that every group member always does all the time. The service is tailored to JXTA's core services features.

Therefore, the deployment procedure follows the same steps as for any other peer service, making use of JXTA's service model without the need of modifying JXTA's initial design. Applications which execute end services or/and clients may communicate through the anonymity service, which acts as an invisible layer. However, as a requirement, the peer group must operate under the PSE Membership Service, guaranteeing that all group members have a properly initialized keystore.

An overview of the proposed architecture is summarized in Figure 2.

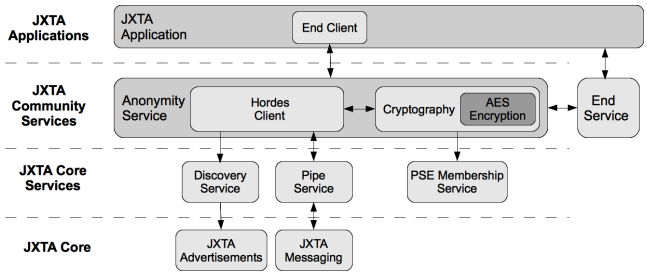

Fig. 2. Anonymity service operation in the context of JXTA's architectural design.

The execution of the anonymizer service in any peer can be divided in the following procedures: Service Initialization, Message Transmission and Message Processing and Forwarding.

Anonymity Service Initialization:

Before the anonymizer service can be used, it is necessary to locate the horde peer group. This group is identified by a well-known JXTA's group identifier. At this point, two things can happen depending on whether the group is found or not. If the group is found, the normal procedure continues. If that is not the case, the peer assumes that no other peer is using the anonymizer service and creates a new horde group. When the Hordes peer group is deployed, a *probability of forwarding (pf)* value, that must have a value in the range $0.5 < pf < 1$, is also established by the group creator. This value's utility will be explained at the Message Processing and Forwarding step. Then, the advertisement of this new group, which includes the pf parameter, is periodically published using the Discovery Service.

Once the Hordes peer group has been established, any peer that desires to join, including the creator itself, must initialize its cryptographic keys and certificates and use the PSE Membership Service. When the group has been joined, the peer's Peer Advertisement is periodically published in the horde group and can be located by any other horde group member from then on. Then, the peer actually deploys the anonymizer service at the JXTA Community Services layer.

Taking into account the characteristics of the anonymizer service and JXTA, a measure to optimize the performance of the network has been taken into account. Both the peer's cryptographic information and the anonymizer service's pipe are published inside the Peer Advertisement. This minimizes the number of advertisements spread along the JXTA network, and speeds up all the process, since all the information needed to exchange anonymous messages between horde group members is directly inside each Peer Advertisement.

Before every anonymous message can be sent, it is necessary to perform an additional set up: initialize a *Forward subset* and subscribe to a multicast group.

The Forward subset is a random list of peers used to forward messages to the destination. Each peer has its own Forward subset which is renewed periodically (by default, 24 hours). The Forward subset has to contain only a part of the total horde group members in order to prevent path analysis attacks. The size of this list and the renew time are parameters that can be modified to fine tune the service for different scenarios.

As far as the multicast group subscription is concerned, the main characteristic of Hordes is the use of multicast group for response transmission. Our anonymizer service relies on a premade set of multicast groups with well-known identifiers and all of these groups have an associated pipe. Peers must subscribe to any of these groups in order to receive responses. Is is worth noting that the multicast pipe can be a propagated one, bound to the multicast group, or a JXTA JxtaMulticastSocket one, based on multicast socket emulation. In both cases, all the messages sent to this pipe are propagated to all peers that subscribed the pipe group.

The subscribe process to a multicast group is made by choosing it randomly at query transmission time. The peer is unsubscribed from the multicast group as soon as the response is received, or a time out expires. This implies that a peer cannot send more than one anonymous message at once. Also, Hordes recommends to perform the unsubscribe process only after receiving the next message after the one responding the initial anonymous query. This measure avoids false reply attacks, even though it may cause deadlocks if infinite or long timeouts are set. Nevertheless, these limitations can be overcome by implementing a sending/receiving message queue mechanism.

Message Transmission:

The message transmission procedure is executed whenever a peer wants to send an anonymous message to consume a JXTA end-service. We have chosen to deploy the message exchange between peers using JXTA's secure pipes, which relies on the cryptographic data from the PSE Membership service, such as cipher keys and algorithm type. JXTA can be configured to automatically access this information, or it can be manually accessed in the services code itself. In any case, the exchange of information between two peers is encrypted, and only these two peers can access it.

The steps to send an anonymous message, Msg using the Hordes approach under the JXTA architecture follows:

1) The sender locates the destination peer's Peer Advertisement, $DstPeerAdv$. This advertisement also contains the destination's JXTA identifier $DstId$. The sender and the destination peer should not be the same.

2) From within $DstPeerAdv$, the chosen end-service Pipe Advertisement, $SvcPipeAdv$, is extracted.
3) A random message identifier, $MsgId$, is generated, uniquely identify the message for its life cycle period. Hordes recommends a length of at least 128 bits in order to avoid collisions.
4) One of the available multicast channels in the Hordes group, with identifier $MchId$, is randomly chosen. The peer joins to the chosen multicast group.
5) A message, $HordesMsg$, is constructed, following the Hordes protocol parameters, under the JXTA message name-value pair syntax. It contains the following fields:
   - `tag_desti` : $DstId$
   - `tag_mcast` : $MchId$
   - `tag_id` : $MsgId$
   - `tag_msg` : Msg, structured using a syntax as expected by the end-service.
   - `tag_pipe` : $SvcPipeAdv$.
6) The sender randomly chooses a peer from its *Forward Subset* list of peers, $fwdPeer$.
7) $fwdPeer$'s Peer Advertisement is retrieved. The anonymizer protocol Pipe Advertisement is extracted from it.
8) Using the Pipe Advertisement, a connection to $fwdPeer$'s anonymizer service instance is established and $HordesMsg$ is forwarded.
9) A timeout counter is started and the sender blocks until a message is sent to the multicast group or the timeout ends. The message exchange is only successful in the former case. The received multicast message has the following JXTA message structure:
   - `tag_msg_id` : $MsgId$.
   - `tag_msg_data` : The reply message generated by the end-service.

   Whenever a message is received through a subscribed multicast group, the value of its `tag_msg_id` field of any message received via the multicast group is compared with any identifier generated. If a match is produced, it means that the reply has arrived and the operation is finished. If the timeout expires and the reply has not arrive, it implies that the message has been a lost. At this point a new query can be sent relying on this multicast group.

***Message Processing and Forwarding**:*

This procedure is activated automatically when the anonymizer service receives a forwarded message from any other peer of the horde group (be it the original sender or a peer from the same Forwarding subset). Any peer that has joined to the horde group and belongs to some other peer's Forwarding subset may receive messages and has to process them.

The processing mechanism follows:

1) The probability of forwarding (pf) parameter is retrieved.
2) Generate a random number (n) in the range: $0 \leq n \leq 1$.
3) The forward condition is checked: $n \geq pf$.
4) If the forward condition is true the message is forwarded to another peer in the Forward Subset. The peer randomly chooses another peer from its *Forward Subset* list of peers. and acts like the original sender in the Message Transmission procedure, Steps 6-8.
5) If the forward condition is false the message is not forwarded, but finally sent to the end-service. Extracting the information from inside the Hordes message, the peer:
   a) Tries to contact with the destination peer (tag_desti) and opens the a channel to the end-service's pipe (tag_pipe).
   b) Consumes the service by sending the actual message (tag_msg) to the pipe and waiting for the reply from this pipe.
   c) Subscribes to the multicast group (tag_mcast) and create the multicast message, as expected in the Message Transmission procedure, Step 9.
   d) The multicast message is sent to the multicast group, awaking all subscribed peers that are waiting for a response using this the multicast group.

One of the main differences between the returning multicast mechanism and the forwarding mechanism is that, according to the Hordes protocol, in the former, the message is sent in plain text, without encrypting it. This means that any peer that is subscribed to the multicast group will get the message identifier and the reply to the message and can understand it. However, knowing this information does not reveal the identity of the source or destination peer, guaranteeing anonymity. Nevertheless it is not difficult to add privacy to the response.

IV. Experimental results

Some small preliminary tests have been done in order to assess the performance of this anonymity protocol in JXTA, at least in comparison with other approaches. We compared the JXTA implementation of Hordes with Onion Routing, given some of its similarities in its forwarding mechanism. Out test tried to evaluate the computational effort required at each hop. By measuring it, we could assess the differences in both protocols' forwarding algorithm given an actual implementation. The testbed chosen has been a MacBook Pro computer, where 5 peers running each protocol have been deployed. Since the tests done are just a comparison of the computational resources required and not a general analysis, it is enough if the same environment is used for the testing of both protocols.

The results of showing the amount of time required to process a single hop in a JXTA peer are in Figure 3. This shows that in average Onion Routing is a bit faster than Hordes, which a bit unexpected, since the forwarding algorithm is a more complex. On average, Hordes needs 13,1 ms to compute a hop, while Onion Routing only needs 11,3. Carefully investigating, the reason seems to be because of JXTA secure connection's overhead. In Hordes, a Peer has to decrypt de message, process it, and encrypt it again, whereas

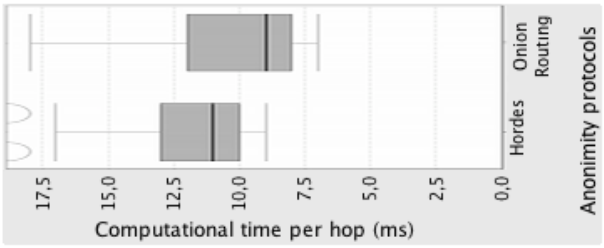

Fig. 3. Time to process a single hop

pf	0.5	0.6	0.7	0.8	0.9
Probability	96.88	92.22	83.19	67.23	40.95

TABLE I
PROBABILITY OF USING LESS AVERAGE COMPUTATION TIME IN HORDES THAN IN ONION ROUTING BASED ON THE pf PARAMETER

in Onion Routing only has to peel out one layer of encryption. Also, the figure shows that these measures are less dispersed in Hordes than in Onion Routing.

Nevertheless, given the pf parameter, the total aggregated processing time used between all nodes may vary, and become higher or lower than the one in Onion Routing for some cases. Given the typical value of total 6 hops in Onion Routing [19], 3 for the query and 3 for response, and the average computation time per hop found previously, Table I has been calculated. This table shows the probability that, for a given value of pf, the average aggregated processing time will be actually lower for Hordes.

V. CONCLUSIONS

This paper has presented the design of a replicated message-based anonymity layer for JXTA, as an alternative to the more popular unimessage-based or split message ones. This layer was created by adapting the Hordes protocol to the specifics of JXTA. The protocol adaptation is tightly integrated to its architecture, working only within the context of a standard service's operation method. Thus, it has not been necessary to define new protocols or primitives aside from the ones already available in JXTA. A further advantage of this is that pipe and cryptographic data publication is seamlessly integrated within its standard presence mechanism.

Currently, the implementation is finished and we have been able to preliminary test it, assessing how fares against other approaches in very basic metrics. Based on the experiments done up to now, it looks like that, surprisingly, even though the algorithm looks much more simple that the one used in Onion Routing, in its actual implementation using the JXTA middleware, the computation time is a bit worse under some conditions (but, on the other hand, is less variable). The use of JXTA secure sockets seems to impact its performance, showing that some designs decisions that look good on paper, or when simulated, do not behave as expected when implemented, interacting with other systems. Thus, experimental evaluation on real systems becomes important.

Further research is twofold. On one hand, further evaluation using other useful metrics is necessary, such as round-trip-time or reliability, in order to meaningfully compare this implementation to others in JXTA. This study will also allow us to fine tune some its overall performance. Once this fine-tuning is finished, it will be possible to create a full anonymity suite that may be integrated into existing JXTA applications, so developers may choose which approach to use depending on their specific needs. On the other hand, in retrospective, we'd also like to assess other replicated-message protocols, such as P5, which also relis on an core design based on peer grouping, such as JXTA.

REFERENCES

[1] Sun Microsystems Inc., "JXTA v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.
[2] Project Kenai, "JXSE: The Java Implementation of the JXTA Protocols", 2011, http://jxse.kenai.com.
[3] Liberatio Systems Management, "Liberatio", 2006, http://www.ohloh.net/p/liberatio.
[4] ConneX, "Connex Project", 2007, http://connex.sourceforge.net.
[5] A. Pfitzmann and M. Hansen, "Anonymity, unlinkability, undetectability, unobserv- ability, pseudonymity, and identity management a consolidated proposal for terminology", 2008, http://dud.inf.tu-dresden.de/Anon_Terminology.shtm.
[6] J. D. Wallace, "Nameless in cyberspace: Anonymity on the internet", 1999, http://www.cato.org/pubs/briefs/bp-054es.html.
[7] X. Ren-Yi, "Survey on anonymity in unstructured peer-to-peer systems", *Journal of Computer Science and Technology*, vol. 23, no. 4, pp. 660–671, July 2008.
[8] M. Domingo-Prieto and J. Arnedo-Moreno, "JXTAnonym: An anonymity layer for JXTA services messaging", *IEICE Transactions on Information and Systems*, vol. E95-D, no. 1, January 2012.
[9] J. Arnedo-Moreno and N. Pérez-Gilabert, "Split message-based anonymity for jxta applications", in *The Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2012)*, 2012, pp. Accepted, to be published.
[10] M. Rogers and S. Bhatti, "How to disappera completely: A survey of private peer-to-peer networks", in *In Proceedings of International Workshop on Sustaining Privacy in Autonomous Collaborative Environments (SPACE)*, 2007.
[11] P. Syverson, D. Goldsclag, and M. Reed, "Anonymous connections and onion routing", *Proceeding of the IEEE 18th Annual Symposium on Security and Privacy*, pp. 44–54, 1997.
[12] Yvo G. Desmedt and Y. Frankel, "Threshold cryptosystems", in *CRYPTO '89: Proceedings on Advances in cryptology*, New York, NY, USA, 1989, pp. 307–315, Springer-Verlag New York, Inc.
[13] S. Dolev and R. Ostrobsky, "Xor-trees for efficient anonymous multicast and reception", *ACM Trans. Inf. Syst. Secur.*, vol. 3, pp. 63–84, 2000.
[14] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5: A protocol for scalable anonymous communication", in *PROC. IEEE SYMP. SECURITY AND PRIVACY*, 2002, pp. 58–70.
[15] R. B. Waters, E. W. Felten, and A. Sahai, "Receiver anonymity via incomparable public keys", in *The 2003 ACM Conference on Computer and Communications Security*, 2003, pp. 112–121.
[16] Y. Wang and P. Dasgupta, "Anonymous communications on the internet", in *Proc. of the 10th ACM Conf. on Computer and Communication Security*, 2005.
[17] B.N. Levine and C. Shields, "Hordes: A multicast based protocol for anonymity", *Journal of Computer Security*, vol. 10, no. 3, pp. 58–70, 2002.
[18] M.K. Meiter and A.D. Rubin, "Crowds: Anonymity for web transactions", *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–93, 2004.
[19] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second generation onion router", *Proceeding of the 13th USENIX Security Symposium*, pp. 303–320, 1998.