Experimental evaluation of anonymous protocols under the JXTA middleware

Joan Arnedo-Moreno, Noemí Pérez-Gilabert, Marc Domingo-Prieto
Internet Interdisciplinary Institute (IN3)
Universitat Oberta de Catalunya
Carrer Roc Boronat, 117 , 7a planta 08018 Barcelona, Spain
jarnedo,nperezg,mdomingopr@uoc.edu

***Abstract*—JXTA is a peer-to-peer (P2P) middleware which has undergone successive iterations through its 10 years of history, slowly incorporating a security baseline that may cater to different applications and services. However, in order to appeal to a broader set of secure scenarios, it would be interesting to take into consideration more advanced capabilities, such as anonymity. There are several proposals on anonymous protocols that can be applied in the context of a P2P network, but it is necessary to be able to choose the right one given each application's needs. In this paper, we provide an experimental evaluation of two relevant protocols, each one belonging to a different category of approaches to anonymity: unimessage and split message. We base our analysis on two scenarios, with stable and non-stable peers, and three metrics: round trip-time (RTT), node processing time and reliability.**[1]

Keywords: peer-to-peer, JXTA, Java, anonymity, experimental evaluation, onion routing, rumor riding.

I. Introduction

JXTA [1] is a technology that enables the deployment of peer-to-peer (P2P) applications, allowing a set of heterogeneous devices to form groups and publish services in a self-organized manner. JXTA has matured through its 10 years of history, undergoing successive iterations that have improved its capabilities. In the latest version at this date, JXTA 2.7 [2], available since March 2011, special emphasis was made to a long forgotten topic: security. Special care has been taken to ensure that now its is possible to effectively use secure endpoint connections, by means of secure sockets, providing an adequate degree of privacy and authentication.

Once a satisfactory security baseline has been established with these additions, it would be interesting to move forward towards providing support for more complex scenarios, which require advanced security services. An example of them is message anonymity, which may consider privacy a necessary but not sufficient requisite, and goes in the opposite direction regarding authentication. Alas, in order for messages to remain truly anonymous, it is not enough that the system just hides source message information to the users at the upper protocol layers, or such data is managed only by a trusted entity. It must not be actually possible for anybody within the P2P network to discover the message source and/or destination, regardless of their role. Providing complete anonymity at all levels may greatly benefit several kind of applications and services, such as auctioning or evaluation systems, since participation may be encouraged by hiding the user's identity.

Currently, there is a broad selection of anonymity protocols which may cater to a P2P middleware such as JXTA, each one using a quite different approach to this problem. However, proposals tend to be evaluated from a qualitative standpoint, their exclusive goal being hiding the identity of participants in a communication exchange, at whatever the cost. In fact, anonymous protocols have a high cost, and comparing them to non-anonymous communications would not be fair, but, nevertheless, it is important to assess whether such cost may be too high under some scenarios or application constraints. In the cases where a quantitative analysis exists, the differing methods (simulation based on logs or theoretical models, experiments using filesharing middlewares, instead of service oriented ones) make it difficult to make a clear comparison between them. Therefore, the only way to be sure which protocol would better suit JXTA is by actual experimentation under this particular middleware.

In this paper we present a quantitative analysis of two significative protocols, each using a different approach. The contributions of this paper are twofold. First and foremost, it provides an empirical study of the protocols' behavior under the JXTA middleware and its ability to integrate differing anonymous protocols when they operate within the context of a service oriented peer group. This provides an insight about its capability to efficiently incorporate anonymity services into its structure and assess which protocol may be more adequate to JXTA's idiosyncracies. Finally, this study can also be considered a general analysis of the chosen anonymous protocols on an even playing field: using exactly the same P2P substrate and lower level protocols and under the same set of conditions, something which, up to our knowledge, is seldom found in the current literature on this topic.

This paper is structured as follows. In Section II, a brief overview of the chosen anonymous protocols is provided. Section III presents the experimental testbed and the parameters chosen for the analysis. The results of such analysis are shown and then discussed in Section IV. Finally, Section V draws some conclusions and provides insights on future 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".

978-0-7695-4779-4/12 $26.00 © 2012 IEEE
DOI 10.1109/NBiS.2012.28

II. Protocols overview

A survey of the most popular anonymous protocols can be found in [3]. The protocols are organized using a taxonomy based on three main categories, based on their approach to achieve anonymity: unimessage, split message and replicated message. The analysis gives special emphasis to the two former categories, and quickly dismisses the third one as too costly, since, directly citing the study, "the cryptographic overhead is accordingly tremendous". Therefore, for our study, a significative protocol from only these two categories has been chosen. This section provides a general overview of each protocol, so it is easier to understand their idiosyncracies and the main differences between each approach. Nevertheless, a much more in depth description of each protocol can be obtained from the original sources.

A. Unimessage protocol

In unimessage protocols, the original message is sent as a single entity through a random sequence of proxies before it reaches the destination. The original data is handled in such a manner that none of the proxies are able to discern whether a message has been received from the the original source or another proxy. The same happens regarding the final destination whenever the message is forwarded.

For our study, we have chosen the onion routing protocol [4], using a fundamental path-based approach, in a similar vein as the one used by the well known Tor infrastructure [5]. The main reason being that it is the most popular kind of anonymous protocol, not only from this category, but between all of them.

Under this approach, the proxies are labelled *Onion Routers* and a path of Onion Routers is pre-constructed by the sender before the message is sent. The message is repeatedly encrypted in a manner that, during transit towards the destination, a single encryption layer can be taken out, one at a time, at each Onion Router. Each time a layer is peeled off, only the identity of the next hop is disclosed, but whether this next hop is the final destination or not is unknown to the Onion Router. Thus, at each hop, the message gets nearer to the destination, but, given a message at any stage of the transmission, it is not possible to know the identities of both the sender and the destination. This behavior is summarized in Figure 1.

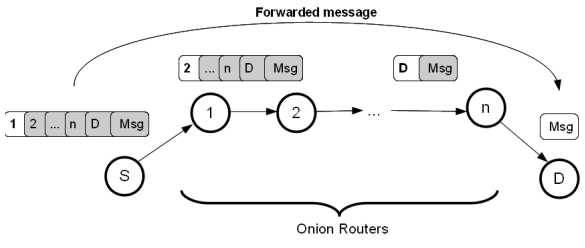


Fig. 1. Onion Routing message forwarding mechanism (S: Sender, D: Destination).

B. Split message protocol

This category groups protocols which don't send the original message as a single entity, but split into different parts which are sent across the network and must be somehow recollected. This is usually achieved using threshold systems [6], where a secret is split into n parts, which are distributed through the network. It is enough that t parts ($t < n$) are gathered to be able to recover the original secret. This idea is directly applied to messages, which take the role of the shared secret, distributed among the peers. The peer which is able to recover the secret is the one that finally transmits the data, acting on behalf of the actual sender, and thus hiding its identity.

From the different available protocols using this approach, Rumor Riding [7] has been chosen, being the most recent one at this date. Its behavior is summarized in Figure 2. When a peer, the *initiator* I, wishes to issue an anonymous message, an AES [8] symmetric key is generated and used to encrypt the query. Both the key and the ciphertext are arranged into different packets, called *rumors*, which are separately forwarded across the network following a random walk algorithm. The rumor's Time-To-live (TTL) is adaptatively chosen so there is a high probability that, at some time, they will have crossed the same peer. That peer, the *sower* S_I, is able to decrypt the ciphertext and retrieve the original query. Then, S_I acts on behalf of the initiator, probabilistically flooding the query across the network until it reaches any peer able to process it, the *responder* R. Since queries are related to document retrieval, more than one responder may exist.

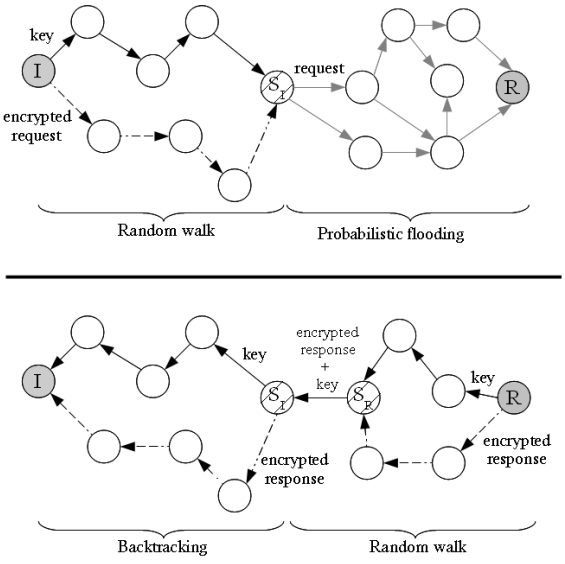


Fig. 2. Rumor Riding protocol messaging (I: Initiator, R: Responder).

Once the query has been processed, the response is sent back from R to I using a similar strategy. However, this time the response is encrypted using I's public key prior to rumor generation. Then, a new pair of rumors is forwarded until they have crossed a new sower, S_R. However, the

response is not recovered at S_R, but the response rumors are directly submitted to S_I. From S_I, both rumors independently backtrack the path the query rumors took at the start of the protocol, until they converge at I, which is the only peer able to recover the plaintext response.

III. Experiment description

The chosen testbed to deploy our experiments was PlanetLab [9], an open, globally distributed network service platform surpassing the mark of 800 nodes spanning over several countries. Because of its attractive features, PlanetLab has practically gained the status of a standard for conducting large scale experiments between nodes spread across the Internet. Nevertheless, to get the best out of it, some of its limitations must be previously taken into account and some good practices have to be followed [10]. The details on how the chosen anonymous protocols were adapted to the specifics of the JXTA platform at its service layer can be found in our previous work [11], [12]. Our experiments were directed towards assessing three metrics: RTT, processing time required by the protocol algorithm at each node whenever a message is received, and reliability, measured as amount of messages lost given the total number of messages sent.

Each experiment was divided into a set of 10 individual tests. For each test, a JXTA peer group, composed by 32 peers, was created, in such a manner that individual peers were deployed in separate nodes at random locations. Such number of peers was chosen according to the typical maximum value for peer group size in a JXTA network [13]. Using this approach, we could test the behavior of the peer group under different network configurations, using different sets of nodes. Anonymous requests between peer group members were randomly generated according to a parameter p, with a value between 0 and 1. Each individual test in an experiment was divided in 250 time slots, each slot 5 seconds long. During a time slot, peers had a p probability to actually send a request to a random node. By increasing and decreasing the value of p, we could increase and decrease the expected amount of requests in the network at a given time, but still maintaining a random aspect, instead of generating them at a fixed predictable rate. Each test is considered concluded when the last response is actually received (which may happen some time after the 250th slot). We ran experiments for values of $p = 0.2$, 0.4, 0.6, 0.8 and 1.0.

The respective values for number of time slots and slot length were chosen by executing some preliminary tests using different values and taking initial measurements on expected RTT. From these tests, we could estimate a value that guaranteed that, in most cases, messages were sent at a rate that would not clutter requesting nodes with too many pending unanswered requests. In addition, we observed that PlanetLab nodes tended to become unresponsive after some time, and therefore, it was difficult to guarantee that most peer group members would remain active for long test lengths. From these observations, we decided to limit each individual test to about 20 minutes long, and wait about 5 seconds between requests. It must be admitted that the chosen values cannot be used as a generalization for any application environment, since PlanetLab nodes are shared with other applications, executed by other researchers at the same time, which may affect its behavior. However, since the goal of our study is comparing the protocols, and not actually measuring its overall performance, we considered it is good enough that they are just always executed under equal and fair, but minimally sensible and justifiable, conditions.

Once the basic experiment layout was established, we deemed interesting to execute them, given a protocol and a value of p, using two scenarios that took into account some of the PlanetLab nodes' idiosyncrasies. Depending on the scenario, the way nodes were chosen for each individual set in an experiment slightly differed, allowing us to focus on a particular metric and emulate a different environment.

The first scenario assumed that peers are usually stable. They are not always online and can, in fact, disconnect from the network, but they tend to do it in an orderly manner and such occurrence is very rare nevertheless. This scenario is built by carefully looking up and choosing a selected set of responsive nodes before the set is executed, emulating a much more controlled peer group. From our previous experience using Planetlab, we have detected that, due to the heavy sharing of nodes, a non negligible number of them tend not to be sufficiently stable, stopping being responsive at random intervals or disconnecting for an indefinite amount of time. That's the main reason nodes must be carefully harvested before they are deemed useful for this scenario. Our main goal by building this ideal context was focusing on the impact of network load on the protocols' behavior, since the amount of dropped messages should be low, and thus, many packets actually coexist in the network at any given interval of time as the value of p increases. In addition, this scenario would mimic a controlled environment, for instance, an enterprise service accessed by a very specific set of peers, owned by a single organization or a coordinated sysadmin team.

As far as the second scenario is concerned, instead of considering PlanetLab node instability a hinderance, we seized the opportunity to use this behavior to compare the protocols under a "free for all" approach, in an unpredictable environment, where peers may become unstable at any moment and drop from the network without previous notice. This scenario is much more focused towards analyzing protocol reliability under a dynamic environment, imitating a much more ad hoc environment, were peer ownership uncertain. It must be remarked, however, that in this scenario, peers are not purposely unstable, or an actively harmful behavior is forced on them. Simply, their stability is not guaranteed at all.

Given this experiment configuration, a total of 200 tests were executed (2 protocols * 2 scenarios * 5 transmit probabilities * 10 network layouts). After its execution, the data of about 200000 messages was recollected for each scenario and used to analyze the RTT and reliability metrics. As far as the analysis of processing time is concerned, the amount of data for each case was in the range of about 2-6 million individual

measurements, depending on the scenario and protocol.

IV. Experimental results

In this section, we show the results of the experiments and discuss some of our conclusions extracted from the data analysis, using a separate section for each metric analysis. In each one, we also highlight the assumed, given the claimed protocol strengths and flaws, and actual protocol behavior for each scenario. In those cases where results did not match the initial assumptions, we further investigated the protocol behavior and we hypothesize about the probable reasons.

All results are shown using box-and-whiskers diagrams, since, given that tests encompass different network topologies, we deem it is more interesting to assess its variability. Nevertheless, the specific average values are listed in an adjoining table, given each transmit probability, converted to a percentage format from the value of p for each experiment. The horizontal line in each box corresponds to the median value.

A. Round Trip Time measurements

The resulting RTT values extracted from the experiments, for both protocols, are shown in Figure 3. The left hand chart shows the results for the stable scenarios whereas the right hand one shows the non-stable one. In this case, the given transmit probability makes it possible to assess how different degrees of traffic congestion affect the RTT values. It must be noted that, even though both figures look similar at first glance, their actual range of values in the y axis differ. The non-stable scenario has a much higher range, as could be expected.

From the figure, it can be cleraly seen that Onion Routing has a much lower RTT value for all cases, as well as a quite low variability, whereas Rumor Riding needs more time (about 5 times in some cases) to fully finish a query-response exchange. In fact, it is interesting that Onion Routing keeps a quite stable behavior regardless of network traffic, or at least, the RTT increase is negligible. However, the behavior of Rumor Riding is quite the opposite. The main reason we found for this difference is the fact that Rumor Riding tend to depend on a high number of hops, and, in fact, to some aspects linked its required processing time, that will be shortly explained.

B. Processing Time measurements

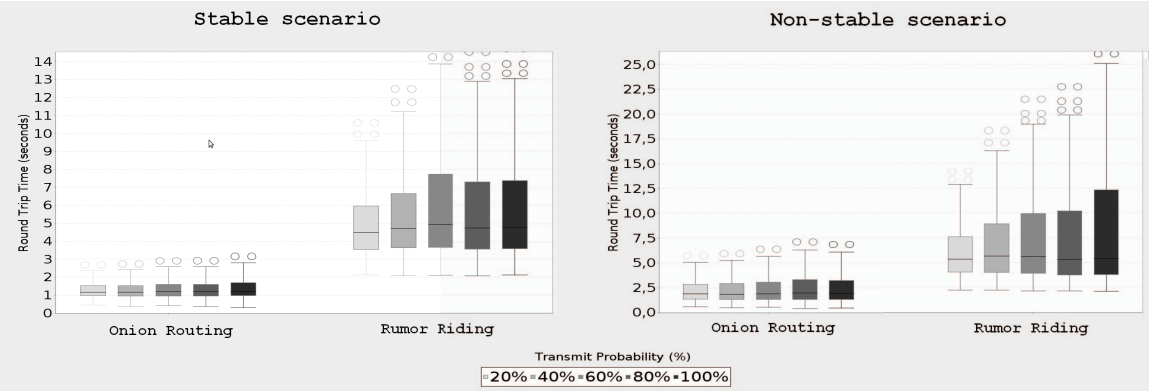
As each message travelled across the network, being routed through different peers, we measured the amount of time it took to process the message since the moment it is received to the moment it is forwarded to a new peer. The results are shown in Figure 4. The charts have also been divided according to different transmit probability values, since as the amount of messages travelling across the network at the same time increases, so does the size of some data structures where look up operations are executed by the message processing algorithms. Nevertheless, it is worth noting that this circumstance is much more prevalent in the Rumor Riding algorithm than in the Onion Routing one.

Analyzing this benchmark is specially interesting since one of the main presumed advantages of the Rumor Riding protocol is relying on symmetric key cryptography, more specifically the AES algorithm. The authors explicitly state that this approach is 100 times faster than using asymmetric algorithms, as is the case in Onion Routing. Even though the expected amount of hops is higher in this protocol, having a big advantage in the required processing time at each hop is their main bet to outperform Onion Routing. Therefore, it should be one of the key features in a fair comparison between both protocols. However, from the previous RTT results, it seems such theoretical advantage does not pay off in an actual implementation. In fact, as far as processing time is concerned, the Rumor Riding algorithm is a bit slower overall. This is even more surprising given the fact that in the JXTA design and implementation of its algorithm some performance tweaks were considered [12] in the look up method on data structures. The original proposed algorithm relied on an even more time consuming approach.

Carefully analyzing the peer behaviour, we have found two possible explanations for these results. First and foremost, the statement that using symmetric cryptography will soundly outperform any asymmetric cryptography approach is, at least, misleading under the context of message protocols, since a pure asymmetric approach is seldom used. An hybrid approach, based on wrapped keys, is almost always implemented instead [14]. This is the case for Onion Routing protocol. Even though using the AES algorithm may be still faster, the advantage is not by two orders of magnitude. In addition, the Rumor Riding data processing model is a quite more complex than in Onion Routing (messages are stored in a cache in order to match previously processed rumors). At the end, cryptographic operations only play a small, though important, part in the algorithm. This consideration becomes of greater importance if some of the operations in the JXTA messaging model (waiting for connections, data encoding/decoding between layers, etc.) are taken into account, apart from the processing algorithm steps. Since they are even more costly than the overall time required for cryptographic transformations, we have found that any edge gained by using the AES algorithm becomes negligible.

C. Reliability measurements

The results regarding protocol reliability, measured as the number of replies received given the amount of requests sent, are presented in Figure 5. In the stable scenario it is expected that the amount of lost messages is much less than the stable one. Even in the stable scenario, we found that connections could be dropped because nodes, being shared, also transmitted traffic related to other applications apart from our tests. Nevertheless, that would be just the case in a real environment. Peers would not be dedicated to a single application. Again, the results show that Onion Routing's performance is better, specially in those cases with high traffic (higher transmit probability values). It is remarkable that Onion Routing is able to maintain a good reliability level even when operating under

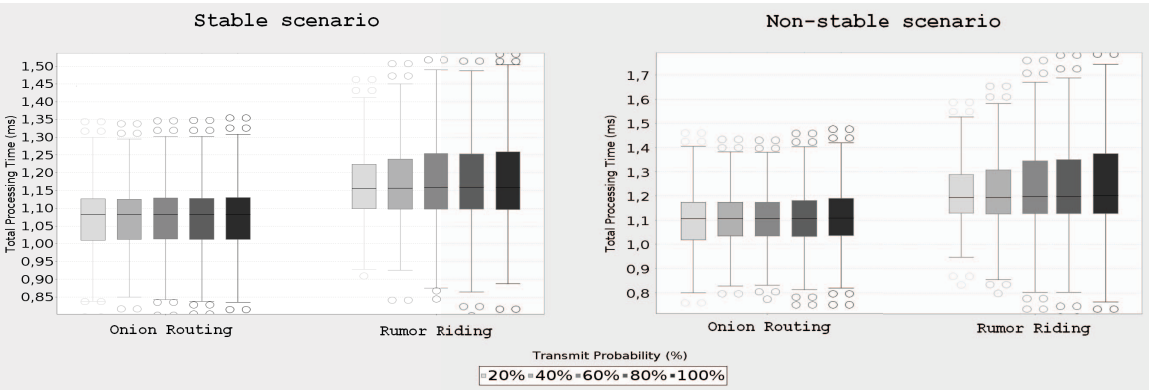


AVERAGE RTT(SECONDS):

Protocol	20%	40%	60%	80%	100%
Onion Routing	1.45	1.48	1.64	1.62	1.75
Rumor Riding	5.27	6.06	7.20	7.39	8.10

Protocol	20%	40%	60%	80%	100%
Onion Routing	2.80	3.10	3.06	3.33	3.22
Rumor Riding	6.58	7.72	8.74	9.09	10.34

Fig. 3. Round Trip Time experimental results



AVERAGE PROCESS TIME(SECONDS):

Protocol	20%	40%	60%	80%	100%
Onion Routing	0.88	0.88	0.92	0.90	0.92
Rumor Riding	1.13	1.16	1.24	1.28	1.33

Protocol	20%	40%	60%	80%	100%
Onion Routing	1.15	1.28	1.27	1.23	1.38
Rumor Riding	1.38	1.49	1.68	1.63	1.81

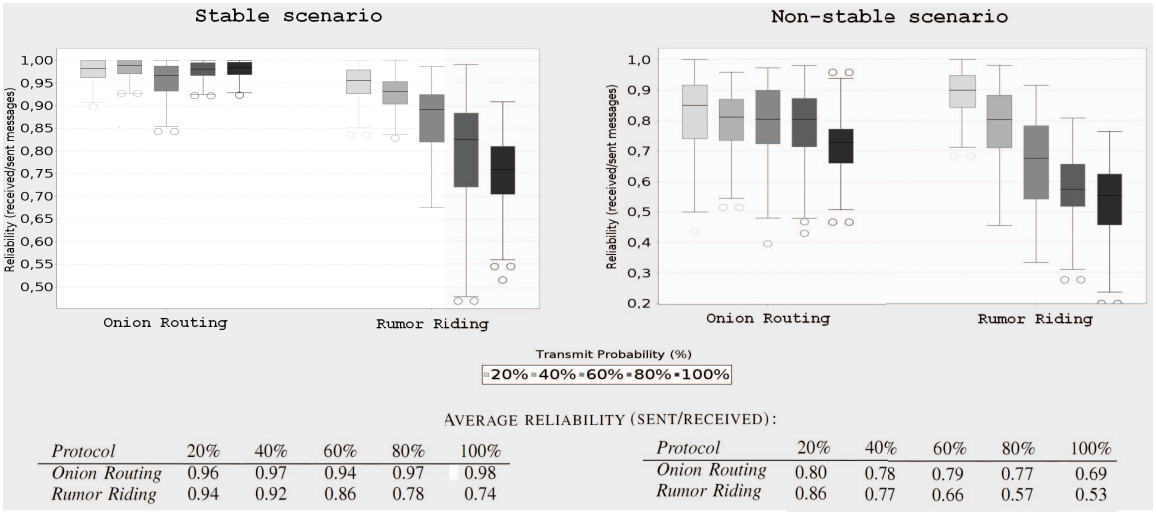
Fig. 4. Processing time experimental results

heavy load scenarios. Rumor Riding, even though it is assumed to cater to dynamic environments, did not fare very well in the unstable scenario.

V. CONCLUSIONS

In this paper, we used Planetlab to design and deploy a testbed that allows to experimentally evaluate a set of efficiency metrics, round trip-time, node processing time and reliability, in two anonymous protocol implementations adapted to JXTA service access. Two scenarios were taken into account to evaluate the protocols, one where nodes are very stable and another where no preassumptions can be made on that regard.

After studying the results provided by the experiments, had the final conclusions to be summarized in two brief sentences, they would be ”newer does not mean better” and ”simplicity beats complexity”. Even though Rumor Riding is a much more recent and sophisticated protocol than Onion Routing, a quantitative analysis regarding their efficiency in a broad set of network configurations shows that the latter is more appropriate as far as its deployment in a JXTA-based network is concerned. Truth be told, before initiating our study, we expected a higher contention on some of the chosen benchmarks and scenarios, and thus the study would enable us to identify in which particular cases one protocol was better than the other. The main reasons for this expectation were some of the claims given in the original sources, which presented Rumor Riding as a direct alternative to path based approaches. Mainly, that message processing was presumed to be faster, being based on symmetric cryptography, and more adaptable to dynamic conditions during message forwarding from end-client to end-



AVERAGE RELIABILITY (SENT/RECEIVED):

Protocol	20%	40%	60%	80%	100%
Onion Routing	0.96	0.97	0.94	0.97	0.98
Rumor Riding	0.94	0.92	0.86	0.78	0.74

Protocol	20%	40%	60%	80%	100%
Onion Routing	0.80	0.78	0.79	0.77	0.69
Rumor Riding	0.86	0.77	0.66	0.57	0.53

Fig. 5. Lost messages experimental results

server. This protocol relied on these two presumed advantages to provide an overall better performance, compensating the need for a higher hop count before reaching the destination. However, after the dust settled, Onion Routing just resulted to be better in all scenarios. Given our results, it can be concluded that it is not certainly by chance that it is one of the most popular approaches to anonymous communications, and this fact also translates to JXTA service access under peer groups. Right now, the only matter open to discussion would be a qualitative comparison from a pure anonymity degree standpoint, assessing their respective vulnerabilities to different attacks, but from an efficiency standpoint, the victor is clear.

Truth be told, in the current analysis, peer groups were quite static. Once a set of peers was chosen, it could be possible that members dropped from the network (specially in the non-stable scenario), but there was never a cycle of never members. As further work, we deem interesting to evaluate the behavior of anonymous protocols in groups with a much greater degree of dynamism, were peers routinely join and disconnect from the group, but following a typical peer lifecycle, and not necessarily because they just entirely drop from the network. We also found that, in some average values, some statistical oddities appeared (some average values did not linearly increase as the transmit probability did). This fact deserves also further investigation.

Finally we also plan to give a chance to the third category of anonymous protocols, replicated message-based ones, so we can confirm that their overhead is actually too high or not. Hordes [15] is a promising candidate for experimentation, an hybrid approach that tries to alleviate the main shortcomings in protocols from this category by using some of the perceived strengths in path based approaches. It is one the few proposals where the authors explicitly quantify its anonymity degree and is quite complementary to unimessage protocols, being able to withstand attacks which can be successful on them.

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] Ren-Yi X., "Survey on anonymity in unstructured peer-to-peer systems", *Journal of Computer Science and Technology*, vol. 23, no. 4, pp. 660–671, July 2008.

[4] Goldsclag D. Syverson P. and Reed M., "Anonymous connections and onion routing", *Proceeding of the IEEE 18th Annual Symposium on Security and Privacy*, pp. 44–54, 1997.

[5] Mathewson N. Dingledine R. and Syverson P., "Tor: The second generation onion router", *Proceeding of the 13th USENIX Security Symposium*, pp. 303–320, 1998.

[6] Yvo G. Desmedt and Yair Frankel, "Threshold cryptosystems", in *CRYPTO '89: Proceedings on Advances in cryptology*, New York, NY, USA, 1989, pp. 307–315, Springer-Verlag New York, Inc.

[7] Liu Y., Han J., and Wang J., "Rumor riding: Anonymizing unstructured peer-to-peer systems", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 3, pp. 464–475, 2011.

[8] FIPS Federal Information Processing Standard, "Advanced Encryption Standard (AES)", 2001.

[9] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services", *ACM SIGCOMM Computer Communication Review*, pp. 3–12, 2003.

[10] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using PlanetLab for network research: myths, realities, and best practices", *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 17–24, 2006.

[11] 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.

[12] 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.

[13] Deters R. Halepovic E., "The JXTA performance model and evaluation", 2005, vol. 21, pp. 8377–390.

[14] J. Staddon B. Kaliski, "PKCS1: RSA Cryptography Specifications. Version 2.0", 1998.

[15] Shields C. Levine B.N., "Hordes: A multicast based protocol for anonymity", *Journal of Computer Security*, vol. 10, no. 3, pp. 58–70, 2002.