SPP: A Secure Protocol for Peer-to-Peer Systems

Quang Hieu Vu [1,2]
[1] *Cryptography and Security Department, Institute for Infocomm Research, Singapore*
[2]*ETISALAT BT Innovation Center, Khalifa University, UAE*
qhvu@i2r.a-star.edu.sg

***Abstract*—The main challenge of reputation-based methods that are used to evaluate trust of peers in Peer-to-Peer (P2P) systems is how to collect and distribute reputation scores of peers efficiently. While several protocols have been proposed to address this challenge, most of them rely on a gossiping algorithm, which is expensive and communication-intensive. In this paper, we propose SPP, a Secure Protocol for P2P trust management using trees in which we present a trust model between nodes in a tree, and explain how trust is established and maintained between pairs of nodes. We show that, compared to existing methods, our design allows for scalability and efficient algorithms with low overhead. We present a possible implementation of our basic tree design, and explain how it could be made stable and robust to network dynamism, thus addressing the greatest weakness of a tree structure. We also analyze the implementation for its security against various adversarial scenarios, and suggest further improvements that are possible for general tree-based systems.**

Keywords - Trust management; Security; Protocol; P2P.

I. INTRODUCTION

While Peer-to-Peer (P2P) systems have become very popular, security is still a problem of greatest concern among people using these systems. It is because in P2P systems, peers are usually anonymous. A popular method for evaluating trust in P2P systems is to use reputation, where the reputation of peer is determined based on its prior transactions with other peers. The main challenge of this method is how collect opinions of all peers in the system about a particular peer, and to provide access to the reputation score to all who request it. In existing reputation-based systems like eBay [1] and Amazon Auctions [2], the solution to both challenges is to use servers. However, this solution suffers from problems of server-based systems such as network bottlenecks, and having a single point of failure.

An alternative solution is to employ a gossiping algorithm [3], [4], [5], [6] for exchanging knowledge among peers in the system. In this way, after a sufficient number of knowledge exchange steps, every peer should have a global knowledge about reputations of peers in the system. The gossiping algorithm can be implemented in two ways. In the first way, each peer itself has to maintain global state and knowledge of the whole system. After each transaction or after some interval time, peers report the score of their partners in new transactions to all other peers in the system. Based on this report, peers update their global state. This method requires that peers keep and maintain reputation scores for all peers, which is inefficient. The second way avoids this problem by letting each peer keep track of the reputation of peers that it has been in transactions with previously. Whenever a peer wants to retrieve the reputation of another peer, it can apply the gossiping algorithm to ask for that peer's reputation from its neighbors, the neighbors of its neighbors, and so on. Combining the feedback with its local knowledge, it can determine a trust value of that peer. Even though these two ways are different, they share the same drawback of the gossiping algorithm: both are expensive in terms of computation and communication costs.

Instead of using gossiping, in this paper, we present SPP, a Secure Protocol for trust management in P2P systems based on a tree structure. Our method organizes nodes at different positions in a tree based on their reputation, with peers of higher reputation at higher levels. In this tree structure, reputation of a peer is maintained at its parent. A peer always trusts its ancestors while it is answerable for its descendants. When two peers execute a transaction, a trust route is formed between them. If the transaction succeeds, a reward is given to all nodes in the route. On the other hand, if the transaction fails, all nodes in the route are penalized. The main advantage of SPP is that it does not incur a high cost in reputation management compared to methods that use the gossiping algorithm for reputation distribution. Furthermore, the flexible design of SPP allows us to develop a complete system for trust management for use in any existing decentralized P2P system. To sum up, our paper makes the following contributions in the area of P2P security:

- We formulate a general-purpose solution to trust management in P2P systems based on a tree structure and show how to augment a tree with extra links to create robustness and to allow nodes to exchange queries without overwhelming the root. This eliminates the problems of bottlenecks and single points of failures.
- We extend BATON [7], an existing tree structure, to support our proposed protocol, implement the protocol, and conduct an experimental study to evaluate the effectiveness and efficiency of our protocol.

The rest of the report is organized as follows. In Section II, we introduce related work in the area of trust management in

 ISBN: 978-1-61208-102-1

P2P systems. In Section III, we explain our proposed basic design in terms of the trust and security models. In Section IV, we discuss some issues of our basic design, and suggest possible solutions to improve it. In Section V, we present a way to use our design to extend an existing tree structure (BATON [7]) to support reputation management. Section VI describes our experimental study and its results. Finally, in Section VII, we summarize the important contributions of our design and its potential.

II. Related Work

Trust management in P2P systems can be classified into two main categories: credential-based and reputation-based management. Credential-based management systems employ the classical method where a peer trusts another peer after examining the other peer's credentials. If the credentials satisfy the peer's policy, that peer can be trusted in a transaction. Otherwise, the peer would refuse to be in a transaction with the other peer. The weakness of this method is that it has to rely on servers for keeping every peer's credentials, which is not entirely a scalable method. Moreover, since credentials are usually generated once and stored, past transactions of peers, both good and bad, are not considered. As a result, this method is only suitable for specific kinds of systems with fixed credentials, like access control systems. Examples of systems that apply this trust model include X.509 [8], PGP [9], PolicyMaker [10] and its successors, REFEREE [11] and KeyNote [12].

On the other hand, reputation-based management systems rely on reputation to evaluate the trustworthiness of a node. In general, the reputation of a node is computed based on its previous transactions with other nodes in the system and how they rated these transactions. Reputation-based management systems can be further classified into two sub-categories. One type of system considers only the reputation of an individual, like those in [3], [4], [5], [6], [13], [14], [15], while the other takes into account social relationships between nodes in addition to individual reputation, such as [16], [17]. Since no nodes know of all nodes in the system, reputation of nodes have to either be collected and stored on servers for reference or distributed to all nodes in the network by the gossiping algorithm. Both of these methods are not viable for large networks because the first method is not scalable while the second method is expensive.

In the field of data structures, the structure of a tree has a very important role. NICE[1] can be used to do scalable application layer multicast [18] by using the idea of overlay trees for efficient content distribution. However, very few networks proposed so far uses the topology of a tree. In this kind of structure, if the standard query processing algorithm is used, nodes near the root will be accessed many times more compared to nodes near the leaves, and hence congestion at the root or nodes near the root may happen. This is not acceptable in P2P systems. To avoid this problem, P-Tree [19] suggests a use of partial tree structure. In this method, each leaf node in the tree is represented by a P2P node while internal nodes are all virtual. Each P2P node maintains a path from the index root to the leaf node. As a result, queries can be processed at any node without pushing all queries to a special node. Note that, however, if a node has to maintain the whole tree structure, the maintenance cost is very expensive and not suitable for P2P systems. Alternatively, BATON [7] creates links between nodes at the same level in the form of routing tables. Consequently, queries can be processed at any node in the tree without going through the root. Nevertheless, these systems focus only on range query processing, and not trust management.

[1]NICE is a recursive acronym for "NICE is the Internet Cooperative Environment". See http://www.cs.umd.edu/projects/nice/.

III. Basic Design

A. Trust Model

Our tree consists of peers arranged by their reputation. Peers of higher reputation occupy positions at higher levels in the tree, with each parent having a higher reputation score than their children, and so the root node is the peer with the highest reputation. Peers of higher reputation are accorded higher privileges of some kind, to provide incentive for nodes to increase their own reputation. We develop the following terminology and use it to present the model of trust relationships between nodes in the tree.

- *Trust link*. A link exists between a peer and its child, and this denotes a link of trust. We say that (1) the child peer in this link trusts its parent because the parent has a higher reputation than itself, and (2) the parent is answerable for the child. The latter point means that any misbehaved action on the part of the child reflects poorly on the parent as well, and the parent is also held accountable for any misbehavior of the child. This is desirable because it is every peer's responsibility to minimize the presence of malicious peers entering the network as children. Trust links are inherently transitive, because a child that trusts its parent would also trust its parent's parent of higher reputation, while a parent is accountable for its children and thus its children's children as well.
- *Trust chain*. A chain of trust is formed by consecutive trust links. In such a chain, we say that the lowest peer trusts the highest peer, based on transitivity of trust in our model.
- *Trust route*. A trust route is the path between any two peers in the tree. It is composed of one or two trust chains that meet at a common ancestor of the two nodes. We call that ancestor the *connector* of the route. The trust route also includes the connector's parent, which we label as the *arbiter* of the route. A trust

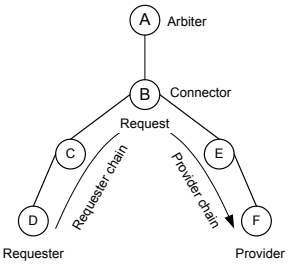

Figure 1. Trust relationships in a trust route

route is formed when a peer requests content from another peer in the system. The former peer is known as the *requester*, and the latter is the *provider*. The trust chain from the requester to the connector is called the requester chain, and that from the provider is called the provider chain. Figure 1 illustrates the relationships mentioned here.

- *Transaction*. A transaction is initiated by a requester, by sending a request through the tree to a chosen provider. The provider responds with the appropriate content to the requester. Transactions occur over a trust route in our tree, and they have a *transaction outcome* in the form of a report sent out by the requester. A positive outcome indicates a successful transaction when the requester is satisfied with the received information. Conversely, a negative outcome indicates a failed transaction when the requester is not satisfied with some part of the received information.
- *Rewards and punishments*. To give a reward means a peer increases the reputation score of a child peer, and a punishment is the converse, a decrease in the reputation score of the child peer. Rewards and punishments are managed based on the transaction outcomes reported by requesters.

B. Trust Management

This subsection describes how trust in our model can be managed. There are two possible outcomes of transactions each of which is dealt with in a separate way.

Successful transactions: if a successful transaction occurs between two nodes via a trust route, parent nodes would reward the child nodes. The rationale is that rewarding a child would allow it to be trusted by more nodes, and hence to increase its potential for bringing in more transactions for itself. This would lead to more opportunities for the parent node to earn its own rewards. In general, after a successful transaction, the arbiter rewards the connector, the connector rewards both the children in the requester and the provider chains, and so on, downward both trust chains. The only exception is the requester, which does not get any reward for initiating a request, since it adds no value to the network.

Failed transactions: for a failed transaction, the converse happens. The arbiter punishes the connector, which in turn pushes the blame downward the tree from parents to children in both chains. The requester again is unaffected by the punishments because it has nothing to gain or lose for accurately reporting the outcome of the transaction. A truthful report would, however, increase the effectiveness of the whole network. To prevent the malicious scenario of a node deliberately reporting multiple failed transactions, a parent might keep track of node failure reports, and identify any nodes that are misbehaving in this way. The parent could then terminate trust links with any evil node, deeming it to be deliberately causing trouble by either requesting content from reputably bad nodes, or inaccurately reporting many failed transactions.

This protocol leads to several implications: Nodes will try to maximize the number of successful transactions and minimize the number of failed ones, in order to optimally increase their reputation. This selfish and self-centered behavior, however, allows for optimal gains for the system as a whole, because each node selfishly seeks to maximize its own rewards and to do so, it has to shrewdly monitor its children and their behavior in transactions. A node would quickly break off links with children that result in many failed transactions and refuse to forward transactions from such nodes, because it is being held accountable for the behavior of its children. At the same time, a node would be willing to forward requests and content from reputable nodes or new nodes because doing so would give it the potential to increase its reputation.

C. Node Ranking Management

If we want to know reputation score of a node, we have to ask its parent, since the parent in our tree is of higher reputation and is thus more trustworthy. If an internal node cannot accomplish its task or turns malicious, we should replace it with a better node. Since a node may never want to step down from its position, we have to exert control over that node through its parent.

Additionally, reputation scores of a node is not only stored at the parent but also at the grandparent. Consider the situation where a node now has a reputation lower than that of its child, implying that the tree is currently not well-formed. The solution to this situation is to swap the positions of these two nodes through a swap operation, and that can only be done from the position of the parent of the ill-placed node. By changing positions, these nodes also exchange knowledge information of their children and reputation of these children they are keeping. An example of node swapping is shown in Figure 2 in which node B has to swap its position with its child E because E has a higher reputation. Actually, since A knows reputation of all B, C, D, E, F, G, it can also swap positions between B and G if G has a better reputation than both B and E. This sort of swapping can be done if A wants to assign a node that is known to be trustworthy from another subtree to be

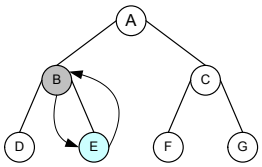

Figure 2. Swapping positions between nodes

the parent of a subtree that could possibly contain colluding malicious nodes.

IV. An Improved Model

The above basic model works well under an assumption that the information given by a node to another node about its children is always correct. In other words, all internal nodes can be trusted in giving information. This is because if an internal node is bad, it can return wrong reputation results about its children to other nodes. For example, a malicious node could return a good reputation score about a bad node or a bad reputation score for a good node. To avoid the problem of the basic model, we introduce a new type of score called a *reference score* for internal nodes. The reference score is used to reflect exactness of information a node gives to others. Now, the trust value of a node is based on not only its reputation score but also the reference score of its parent. In other words, if a node always gives correct information about its children to others, we should trust its information. However, if a node often makes mistakes or gives incorrect information deliberately, our trust in information provided by that node is reduced. Similar to reputation score, a reference score of a node is stored at its parent. So now, as illustrated in Figure 3, before each transaction, a node y should find not only x's reputation score, which is stored at z, the parent of x, but also z's reference score, which is stored at t, the parent of z and after each transaction, y updates scores for both x and z.

The problem now is how to evaluate correctness of information received from z to give feedback of a score after a transaction. Here, we propose a simple solution as follows. When a node is asked about reputation of its children, in addition to giving the total reputation score, it also gives the standard variation of the scores calculated from previous transactions. As a result, the correctness of received information is evaluated by both the reputation score and the standard variation. For example, if the result of the transaction falls far away outside the standard variation, the node giving information should be rated with a bad reference score.

That is not all. Assume that in the worst case, x, z, and t are all malicious peers and they cooperate with each other. If t gives a wrong reference score for z while z gives a wrong reputation score for x, y would still be cheated. To further enhance security, y can also ask reference score of t from its parent. In general y asks for reference scores of a chain of k ancestors of x in which k is a configurable parameter

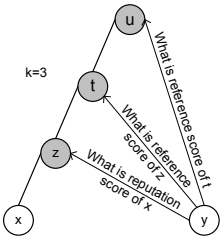

Figure 3. A k=3 reference chain

of the system. Note that since these k nodes form a chain, the cost of lookup algorithm and update algorithm is just $logN + k$. By setting k with a large number, the system becomes strong against collaborative malicious peers. A worry is that k may have to be large, and hence it may be costly. However, since nodes in the system cannot determine the location of them in the tree structure, they have to follow the join algorithm, which scatters nodes along the system to make the tree balanced. As a result, forming a long chain of malicious peers connected by parent-child links is not easy. An example of a $k = 3$ reference chain is shown in Figure 3 in which y asks z for reputation score of x, t for reference score of z and u for reference score of t.

Another technique which can be used by a group of malicious nodes to trick other nodes is to create fake transactions and report good results to their parent to increase their reputation score. To avoid this problem, we just use a simple technique in score calculation as follows. First, we do not simply consider the number of successful transactions as the score. Instead, we limit the score at a maximum value, and the score of a node can only reach that maximum score. Second, we calculate not only the number of successful transactions but also the number of *different* successful transactions of nodes. By "*different*", we mean that transactions of the node that are done with different nodes. As a result, even though a node may have many good transactions with a specific node, it still has a low score if it has many other bad transactions with other nodes.

V. System Design

At this point, we are able to describe in greater detail how to extend SPP to use in BATON, an existing tree based framework. In essence, we try to place our proposed trust management layer on top of an existing networking framework that provides the topology of a tree. The challenge here is to ensure that we can effectively and efficiently implement SPP. In this section, we will first describe the structure of BATON. After that, we introduce the way to deploy SPP on it.

A. BATON

In BATON, each peer participating in the network is responsible for a node in the tree structure. The position of a node in the tree is determined by a pair of a *level* and a *number*. The level specifies the distance from the node

 ISBN: 978-1-61208-102-1

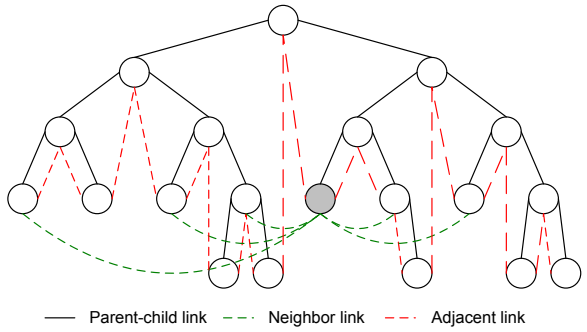

Figure 4. BATON structure

to the root while the number specifies the position of the node within the level. BATON uses three kinds of links to make connections between nodes: parent-child links are used to connect children and parents; adjacent links are used to connect adjacent nodes; and neighbor links are used to connect neighbor nodes at the same level having a distance 2^i from each other. Neighbor links are kept in two special sideways routing tables: left routing table and right routing table. The main purpose of neighbor links is to allow a flexible way to forward queries between nodes in the tree structure without going to the root, and hence BATON can avoid the bottleneck problem as well as single point of failure at the root node. An example of a BATON tree is shown in Figure 4. Note that in this figure, only neighbor links of the grey node are shown.

B. SPP Deployment

Since the most important issues in deploying SPP are how reputation of a node is looked up and how transaction results are reported to responsible nodes, we focus our discussion of these issues.

- Reputation lookup: before each transaction, nodes exchange information about their location in the tree to each other. Knowing the location of a node x, its partner y can infer the location of x's parent, which is z as below:

$$z_{level} = x_{level} - 1$$

$$z_{num} = \begin{cases} x_{num}/2 & \text{if } x_{num} \text{ is even} \\ (x_{num}+1)/2 & \text{if } x_{num} \text{ is odd} \end{cases}$$

  Note that in the tree structure, the level is setup increasingly from the root to the leaf starting at 0 while the number is assigned from the left to the right of each level starting at 1. Now, knowing the location of z, y can issue a query to lookup x's reputation towards z. The algorithm of sending a query towards a node knowing its location is represented as in Algorithm 1. Since at each step, this algorithm makes the search space reduce by half, it is guaranteed that after maximum $O(logN)$ steps, the query should reach the destination node z. When z receives the query, it returns the reputation score of x to y. Note that if x does not tell a truth about its location, and hence when y issues the query either z can not be found or z is not the parent of x. As a result, x can be considered as a bad node.

Algorithm 1 :Query (level l, number n, node z)

```
l_node = level of the current node
n_node = number of the current node
if l_node = l then
   t = the nearest node to z
   t.Query(l, n, z)
else
   if l_node > l then
      t = a child of the current node
      t.Query(l, n, z)
   else {l_node < l}
      t = parent of the current node
      t.Query(l, n, z)
```

- Transaction result report: after each transaction, a similar process is done to report the result of the transaction between partners to their parent. In particular, each peer rates the transaction by giving its partner a score in a range of [-1.0, 1.0]. Depending on the level of satisfaction or dissatisfaction, a value is given in which a positive score is used to indicate a good transaction while a negative score indicates a bad transaction.

VI. Experimental Study

To evaluate the performance of our proposal, we have implemented an extension of BATON [7] to support our security protocol. We tested our system in a network of 1,000 nodes, where exists two kinds of nodes: good nodes and malicious nodes. We just make a simple assumption that that good nodes always do good transactions and give correct answers if they are asked for reputation of their children. On the other hand, malicious nodes always do bad transactions and give incorrect answers about reputation of their children.

A. Effect of Varying Number of Malicious Nodes

We first evaluate the effect of varying number of malicious nodes on the strength of the system. The result is displayed in Figure 5 in which the x-axis presents the percentage of bad nodes in the system while the y-axis presents the percentage of correct answers about reputation of nodes. The length of reference chain in this experiment is fixed at 3. The result shows that our system can suffer up to 20% of malicious nodes while still provide good answers for a reputation of nodes: more than 80% of answers is correct. It is because in order to fully cheat other nodes, malicious nodes have to form a subtree height greater than 3. However, it is difficult to do that since nodes are distributed equally in the leaf level to keep the tree balance.

 ISBN: 978-1-61208-102-1

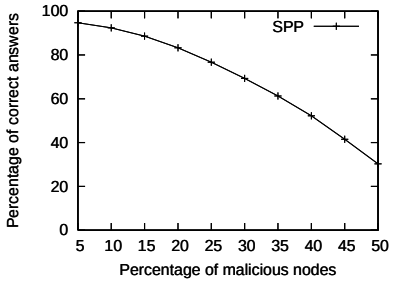

Figure 5. Effect of varying number of malicious nodes

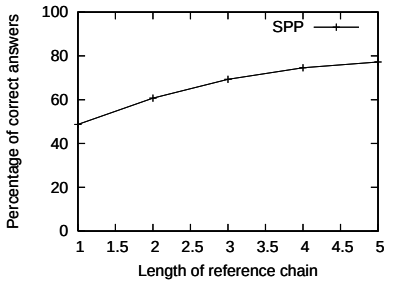

Figure 6. Effect of varying length of reference chain

B. Effect of Varying Length of Reference Chain

In this section, we vary length of reference chain from 1 to 5 while keeping the percentage of malicious nodes at 30%. The result is displayed in Figure 6. The result confirms that the system increases its strength with the increasing length of reference chain.

VII. Conclusion

In conclusion, in this paper, we proposed SPP, a general secure protocol for reputation management in peer-to-peer systems based on a tree structure. By using a tree structure, SPP can avoid the high cost of broadcasting messages that is seen in gossiping-based solutions. At the same time, SPP does not suffer the problem of bottlenecks and single points of failure as seen in server-based solutions through the employment of extra links in the tree structure. We came up with a specific tree structure extended from BATON [7] to implement SPP. Finally, we conducted experiments to evaluate the effectiveness and efficiency of SPP, and presented the above positive results.

References

[1] eBay, "http://www.ebay.com."

[2] Amazon Auctions, "http://auctions.amazon.com."

[3] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "Eigenrep: Reputation management in p2p networks," in *Proceedings of the 12th WWW Conference*, 2003.

[4] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative peer groups in nice," in *Proceedings of the 2003 Infocom Conference*, 2003.

[5] B. Dragovic, B. Kotsovinos, S. Hand, and P. R. Pietzuch, "Xenotrust: Event-based distributed trust management," in *Proceedings of the 2nd International Workshop on Trust and Privacy in Digital Business*, 2003.

[6] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, no. 7, pp. 843–857, 2004.

[7] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, "Baton: A balanced tree structure for peer-to-peer networks," in *Proceedings of the 31st VLDB Conference*, 2005, pp. 661–672.

[8] International Telegraph and Telephone Consultative Committee (CCITT), *The Directory - Authentication Framework, Recommendation X. 509*, 1993 update.

[9] P. Zimmermann, *PGP Users Guide*. MIT Press, 1994.

[10] M. Blaze and J.Feigenbaum, "Decentralized trust management," in *IEEE Symposium on Security and Privacy*, 1996.

[11] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, "REFEREE: Trust management for Web applications," *Computer Networks and ISDN Systems*, vol. 29, no. 8–13, pp. 953–964, 1997.

[12] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, *The KeyNote Trust Management System, Version 2*. RFC-2704. IETF, 1999.

[13] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proceedings of the 9th International Conference on Information and Knowledge Management*, 2001.

[14] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a p2p network," in *Proceedings of the 11th WWW Conference*, 2002.

[15] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proceedings of the 2002 ACM Conference on Computer and Communication Security*, 2002.

[16] J. Pujol and R. Sanguesa, "Extracting reputation in multi agent systems by means of social network topology," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.

[17] J. Sabater and C. Sierra, "Regret: A reputation model for gregarious societies," in *Proceedings of the 4th Workshop on Deception, Fraud and Trust in Agetn Societies*, 2001.

[18] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 205–217, 2002.

[19] A. Crainiceanu, P. Linga, J. Gehrke, and J. Shanmugasundaram, "Querying peer-to-peer networks using P-Trees," in *Proceedings of the 7th WebDB*, 2004, pp. 25–30.

 ISBN: 978-1-61208-102-1