Query Forwarding Algorithm Supporting Initiator Anonymity in GNUnet

Kohei TATARA
Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581
tatara@itslab.csce.kyushu-u.ac.jp

Yoshiaki HORI, Kouichi SAKURAI
Faculty of Information Science and Electrical Engineering, Kyushu University
{hori, sakurai}@csce.kyushu-u.ac.jp

Abstract

Anonymity in Peer-to-Peer network means that it is difficult to associate a particular communication with a sender or a recipient. Recently, anonymous Peer-to-Peer framework, called GNUnet, was developed. A primary feature of GNUnet is resistance to traffic-analysis. However, Kügler analyzed a routing protocol in GNUnet, and pointed out traceability of initiator. In this paper, we propose an alternative routing protocol applicable in GNUnet, which is resistant to Kügler's shortcut Attacks.

1. Introduction

In this paper, we refer to anonymity in Peer-to-Peer framework in the sense that a particular communication cannot be associated with a sender or a recipient. Recently, GNUnet, an anonymous Peer-to-Peer framework, has been proposed to preserve resistance to traffic-analysis and censorship. In GNUnet, all of queries and responses are distributed as keeping a way of bucket brigade flowing. Because when nodes transfer queries, they can rewrite the original source IDs with their own IDs, traffic analysis or censorship do not work very much [3, 2, 4, 1].

However, Kügler proposed two techniques, so called shortcut attacks, for finding out the initiator of session by analyzing routing protocols applied in GNUnet [5]. In GNUnet, in order to restrain performance decrement, nodes are able to forward queries without rewriting the original source IDs. In the basic shortcut attack, an attacker exploits its feature and ascertains the initiator of queries. In the improved shortcut attack, the attacker assumes that nodes closer to the initiator certainly receive much more queries from it, and confirms it with one of the statistical tests. As background of effectiveness in shortcut attacks, there is an assumption that the attacker has a capability to discriminate sessions. Therefore, she can observe statistical deviation of the number of received queries, when the routing protocols such as selecting destination of queries based on probability are used.

In this paper, we propose a routing protocol, which is tolerative to shortcut attacks by adjecting a certain degree of directionality in query propagation.

This paper is organized as follows. In section 2, we briefly introduce the GNUnet. In section 3, Kügler's technique and its efficiency are described. We touch on the distributed hash table in section 4, which is important to illustrate our idea. Then, we will describe a concrete protocol in section 5. We analyze its security and efficiency in section 6. At last, we conclude in section 7.

2. GNUnet

Currently, a mainly application of GNUnet is framework for file sharing [3, 2, 4, 1]. GNUnet supposes some APIs, by which confidentiality and integrity of contents are preserved. We can also emit queries for boolean search to retrieve some contents. But, any initiator and responder of queries cannot be aware of each other. Then, intermediaries are utterly ignorant of queries and contents, and have a plausible deniability with regard to them.

2.1. Encoding

Firstly, content C is divided into some 1K length of blocks, so called $DBlock$. B_i is encrypted with $K_i = H(B_i)$, and identified by name $Q_i = H(E_{K_i}(B_i))$. A pair (K_i, Q_i) is called Content Hash Key (CHK), it is described as $CHK_i := (K_i, Q_i)$. When these blocks are transferred, intermediaries can use CHKs and check those integrity, without knowing those hashed values. A block of unified 25 CHK_i, to which CRC32 checksum is appended, are called $IBlock$. $IBlock$ is also encrypted as well as $DBlock$ and contained. This processing is repeated recursively, and accordingly produce only one $root\ IBlock$. Then, $root\ IBlock$ is included in $RBlock\ R$. $RBlock\ R$

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: T U MUENCHEN. Downloaded on September 9, 2009 at 10:58 from IEEE Xplore. Restrictions apply.

is encrypted under hashed value of keyword K, and identified by name $H(H(H(K)))$. These $DBlock$, $IBlock$ and $RBlock$ are collectively called $GBlock$. For example, if an initiator searches a content related to keyword K, it will publishes some queries which contain $H(H(H(K)))$ values. The only node which retain the corresponding content, can sends the $GBlocks$ related to it. The initiator which receives them, will verify its integrity, decrypt and retrieve the whole content.

2.2. Anonymity and Efficiency

When a node receives a query, it looks up load of CPU or bandwidth, and space in its routing table. If it finds out a cache corresponding to the keyword included in the query, it will put it in the sender queue of destination. Then, the node decides how many and to which neighbor nodes it needs to transfer the query. When it rewrites the source ID field, original source ID must be recorded in its routing table. A reply received from a neighbor node, will be transferred by looking up the entry in its local routing table and putting it in the sender queue. Queries or responses in the sender queue are flushed at various intervals based on random values.

Nodes associate the values, called credit, with neighbor nodes. These values affect priority of queries [1]. For example, a credit c indicates degree of assessment from a node A to a node B. When A sends a query with priority p to B, p will be subtracted from A's credit c by B. On the other hand, when A processes and transfers a reply to B with priority p, p is added to A's credit.

Anonymity in GNUnet is based on following methodologies. Figure 1 shows that a query transferred from node A to node B, is conveyed to C with rewriting the source ID field. If B transfers it without replacing the source ID, then C can understand it conveyed from A. But, A and C cannot confirm that the communication partner is the initiator or the responder of the query each other. Specifically, any behavior of B does not affect anonymities of A and C. Their anonymity is only dependent on the number of queries whose source IDs are rewritten by themselves. B may hurt its own anonymity without replacing it.

When the source ID of query conveyed from B is A, C can reply to A. As A considers C is closer to content holder and sends further queries to A directly, the distance between A and the content holder is shortened. Then, efficiency of content retrieval is heightened. Thus, there is trade-off between anonymity and efficiency in GNUnet.

3. Threat to anonymity in GNUnet

3.1. Tracing initiator

In Kügler's shortcut attacks, an attacker has capability to discriminate some sessions in file-sharing. That is to say, she can consider whether there is relationship among multiple $GBlock$s flowing in network. In order to achieve this condition, she needs to prepare some interested keywords, calculate hashed values of them, and receive some replies to queries involving them in advance.

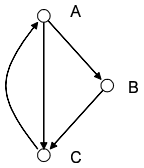

Figure 1. Conveying a reply

Basic Shortcut Attack Here, it is assumed that B transfers a query received from A to an attacker. The attacker sends many queries with high priority to B. Because nodes under overload cannot replace source IDs of them, the attacker, which knows that those queries are conveyed from A, can send many queries to A subsequently. Thus, she can shorten the distance between the initiator and her on purpose. She can also hide the trail of the attack by controlling multiple nodes and sending many queries via them. In this attack, when B is connected to hundreds of nodes, there is little possibility that some queries she can identify are distributed. Besides, because the attacker needs to earn enough credits in advance, it is inefficient and impractical [5].

Improved Shortcut Attack In this attack, the attacker needs to have a certain degree of knowledge about network topology. At the beginning, she guesses that A is closer to the initiator of query than B. Then, she directly sends a reply for the query to A. When her guess is correct, she can return the proper content to A. Therefore, she will be likely to receive further queries from A after that time.

Because of inefficiency in the basic shortcut attack, it is assumed that the attacker applies the improved shortcut attack. She selects a certain couple of nodes in the set of her neighbors, and collects linkable queries related to same content from them. Here, she assumes that the closer to the initiator a node is, the more queries she receives from it. Then, she can conduct a statistical test on the set of queries. In this test, the null hypothesis is that two nodes are equal likely to receive queries from the initiator. If the null hypothesis is not accepted, she can find out the closer node based on her assumption. Subsequently, she selects it and another neighbor node, as a couple of nodes for test. This procedure is repeated recursively. Consequently, she can locate the initiator.

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: T U MUENCHEN. Downloaded on September 9, 2009 at 10:58 from IEEE Xplore. Restrictions apply.

3.2. Efficiency of Improved Shortcut Attack

Kügler estimated lower bound on the number of queries required to locate the initiator with random routing and hot path routing [5]. Let k be the number of neighbors. For every query, a node A randomly selects m of k neighbors and transfers the query to them. Then, the probability that a certain neighbor is randomly selected is uniformly $P_{rnd}(A) = m/k$ for all neighbors.

3.2.1. Random routing Let P_{rnd} be equal for all nodes in GNUnet. Also, let l be distance of an attacker to an initiator.

1. The attacker receives linkable queries from intermediary node i with probability P^i_{rnd}, where $1 \leq i \leq l$. The first query is therefore received after the issuer has sent $m_i \geq 1/P^i_{rnd}$ queries.
2. The attacker will receive linkable queries from the neighbors of the initiator with probability P^2_{rnd}. The first query is therefore received after the issuer has sent $m_0 = 1/P^2_{rnd}$ queries.

Then, a lower bound on the total number of queries required to determine the initiator is as follows.

$$m \geq \frac{1}{P^2_{rnd}} + \sum_{i=1}^{l} \frac{1}{P^i_{rnd}}$$

3.2.2. Hot path routing With hot path routing, a node selects some of neighbors as next receivers by the number of valid responses to recent queries. A node A sends a query to neighbor node I with probability $P_{hot}(A, I) = r/q$, where q is the number of queries that A has recently sent to I, and r is the number of valid responses to them. Let P_{rnd} and P_{hot} be equal for all nodes in GNUnet. Also let l be the length of hot path.

1. The attacker will receive linkable queries from the preceding node with probability P^l_{hot}. The first query is therefore received after the issuer has sent $m_l \geq 1/P^l_{hot}$ queries.
2. The attacker will receive linkable queries from intermediary node i on the hot path with probability $P^{i-1}_{hot} P_{rnd}$, where $1 \leq i \leq l-1$. The first query is therefore received after the issuer has sent $m_i \geq 1/P^{i-1}_{hot} P_{rnd}$ queries.
3. The attacker will receive linkable queries from the neighbors of the initiator with probability P^2_{rnd}. The first query is therefore received after the issuer has sent $m_0 = 1/P^2_{rnd}$ queries.

Then, a lower bound on the total number of queries required to determine the initiator is as follows.

$$m \geq \frac{1}{P^l_{hot}} + \frac{1}{P^2_{rnd}} + \frac{1}{P_{rnd}} \cdot \sum_{i=0}^{l-2} \frac{1}{P^i_{hot}}$$

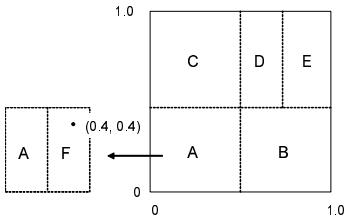

Figure 2. 2-dimentional distributed hash table in CAN, where 5 node (A, B, C, D, E) share the $[0, 1] \times [0, 1]$ coordinate space.

4. Distributed hash table

In recent years, content retrieval using distributed hash table is in the spotlight [6, 7]. By sharing the hash table with the participants, routing and search processing are very fast. In CAN [6], the distributed hash table is constructed on d dimensional torus. Each node in CAN only has knowledge about nodes which administer neighboring zones. Then, the average path, in which a query is transferred, is $O(dn^{1/d})$. In Chord [7], with routing a query according to the hash table as sort of a circle, the average path is $O(\log n)$. n is the number of nodes. In this paper, though we suppose the distributed hash table used in CAN, our routing protocol is applied in the distributed hash table used in Chord too.

Here, for the brief description, we provide 2-dimentional distributed hash table in CAN. In figure 2, 5 node (A, B, C, D, E) share the $[0, 1] \times [0, 1]$ coordinate space. The domain ascribed to each node is called zone. Each node understands the zones administered by neighboring nodes in advance. Then, node F is the new participant in the network. F randomly selects a point on the hash table, as his node ID. For example, when node ID of F is $(0.4, 0.4)$, F asks A for partitioning his zone and providing the zone which holds $(0.4, 0.4)$ for A. At that time, F informs the nodes neighboring to A and F of its own zone. When F gets out of the network, F will notify them and give back his zone.

The node which administers the zone containing the hashed value (a point on coodinate space) of content, knows the address of content holder. If D inserts a content corresponding to a point $(0.8, 0.8)$, it notifies E of its address and node ID. Next, we illustrate that F retrieves the content. When F decides that the B's zone is closer to the destination, it will ask B for the node which contains $(0.8, 0.8)$ and send a query to B. B conveys the query to E. E knows the contents holder D, then returns the ID and address to B. Subsequently, A receives the response from B, and directly communicates with D to get the content.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: T U MUENCHEN. Downloaded on September 9, 2009 at 10:58 from IEEE Xplore. Restrictions apply.

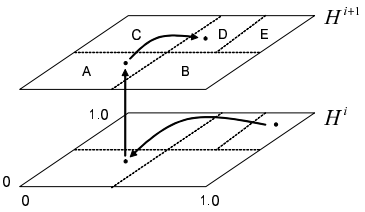

Figure 3. Layered distributed hash table formed by 5 node ($H^i(K) = (0.4, 0.4)$, $H^{i+1}(K) = (0.6, 0.7)$)

5. Our proposal

The background for efficient shortcut attack is that nodes select some intermediaries at random, or according to the previous transactions. For that reason, we need to improve the routing mechanism.

5.1. Contents insertion

At first, the content holder must decide i (≥ 3). Then, it selects the keyword K corresponding to the content, and calculates the hashed value $H^i(K)$. Here, $H^i(K)$ is calculated, such that the initial value K is recursively hashed i-th times. The $H^i(K)$ is the identity of content to search for, the destination of query. Because it is assumed that the hash function is defined on d-dimensional torus, the output value locates a certain point on the coordinate space. The content holder notifies the node which administers the zone taking in the point, of the $H^i(K)$, its own ID and address. It can also select and insert the same content with multiple keywords.

5.2. Contents retrieval

When a node retrieves the content, it must select j (≥ 3). Then, it selects the keyword K, relating to the content to take out, and calculates the j-th hashed value $H^j(K)$. The $H^j(K)$ locates the point on the coordinate space. Next, it decides the value in the Time-to-live field for query in consideration of the load of CPU and bandwidth. It subsequently transmits the query to the node which administers the zone taking in the point. When it receives the response for the query, it can decrypt the $E_{H(K)}(R)$, and get the $RBlock$. After that, it can retrieve the remaining $GBlock$s according to the $RBlock$, and restore the content.

5.3. Processing in each node

A node receives a query, and processes it as follows. The query contains the $H^j(K)$ for identifying the corresponding $GBlock$.

1. The node checks the load of CPU, bandwidth and routing table, and decides whether or not to process the query. If it cannot handle the query, it will discard it.
2. If it finds out the corresponding entry to $H^j(K)$ in local cache, it transmits the $H^j(K)$ and the recipient node ID specified in the query to the contents holder. Actually, the query is put in the local queue. Each queue is flushed at random intervals, and then all messages waiting in this queue are sent to the corresponding node. If there are two or more content holders, the query is conveyed to the node selected at random.
3. Next, it must decide to which it transmits the query, according to the load and value of Time-to-live field. Then, it also decides whether or not to rewrite the sender ID of the query, from the point of view of the anonymity and efficiency. If it replaces the original sender ID with its own ID, it has to put it in the routing table. Finally, it conveys the query to the node whose zone is closest to the zone containing $H^j(K)$. When its own zone contains $H^j(K)$, it has to calculate the $H^{j+1}(K)$, and puts it in the queue.

5.4. Layered distributed hash table

In this paper, we refer to H^i as i-th hash table. It is difficult to finds out the corresponding value on i-th hash table from a given value on $(i+1)$-th hash table. This is due to the property of one-way hash function.

With our routing protocol, a query is conveyed as shown in figure 3. On the i-th hash table, the query has the $H^i(K) = (0.4, 0.4)$. When E decides that D is closer to the zone which contains $(0.4, 0.4)$ than B, E probably transmits it to D. The query is conveyed from D to C, subsequently from C to A. When A does not have the corresponding content for query, it can calculate $H^{i+1}(K) = (0.6, 0.7)$ and decide the destination on the $i+1$-th hash table. Similarly, the query is transmitted toward $(0.6, 0.7)$. Namely, the query is conveyed from A to C, subsequently from C to D. The life time of the query depends on the value in the Time-to-live field. If the node is too busy to process the query, it can discard the query. At last, D has a knowledge of the contents holder, D necessarily transmits the query to it.

6. Security and efficiency

We describes that our routing protocol is free from the shortcut attacks, as shown in the chapter 3. Then, we touch upon the anonymity of the content holder and the efficiency of the routing protocol.

6.1. Possibility of basic shortcut attack

At first, we consider a certain scenario, such that an attacker tries to spoil our routing protocol with the basic

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: T U MUENCHEN. Downloaded on September 9, 2009 at 10:58 from IEEE Xplore. Restrictions apply.

shortcut attack. In the distributed hash table, as shown in figure 3, the query containing $H^{i+1}(K)$ is transmitted. When an attacker imposes too heavy the load on the intermediaries, he can find out the original senders of the query, and ascertain the initiator A. However, when node A is the initiator, the query certainly contains $H^j(K)$. Therefore, the attacker has to guess j. He also must go over the same attack until j becomes to be equal to i. Accordingly, the attack is as difficult as that against original routing protocol in GNUnet.

On the other hand, as an attacker applies the improved shortcut attack, it is assumed that a closer node to the initiator receives the more query than the other node. With our routing protocol, this assumption is not necessarily approved. In GNUnet, a query is conveyed to the node which manages the point specified by its $H^j(K)$. This means that P_{rnd} can be approximated to be zero. Namely, m, which is the value required to ascertain the initiator, can be increased very much. The attacker can find out the initiator by keeping his eyes on the query containing the $H^i(K)$, whose i is the smallest of all the queries. However, in order to affirm that a specified node is the initiator, the attacker needs to look up the i. After all, the attacker has diff iculty eavesdropping all the $H^i(K)$ in the network, and checking out them.

6.2. Anonymity of contents holder

A node, whose zone contains $H^i(K)$, is able to correlate it with a node ID. Then, if an attacker can guess the keyword K, and calculate $H^i(K)$, he may ascertain the content holder. When it is assumed that the number of nodes is n, and each node has the same size of zone, the probability, with which attacker's zone contains the point specified by $H^i(K)$, is $1/(ni)$. For this reason, we need another solution, as selecting the keyword K as difficult to guess as possible.

6.3. Efficiency of routing

With random routing or hot path routing, the query is conveyed to the nodes selected at random or according to the previous transactions. Therefore, the queries for retrieval are spread out like the concentric circle. This mechanism is inefficient compared with the conventional server-client model. With our routing protocol, because the queries leave for the destination specified by $H^j(K)$, we can retrieve some contents efficiently.

Then, it is important what the values the nodes set to j required for content retrieval, and i required for inserting. When i is smaller than j, the probability with which a query arrives a content holder specified in it, is very small. When i is much larger than j or j is much smaller than i, the value in the Time-to-live field may become to be zero and be discarded before the query arrives the content holder. Because there is a trade off between anonymity and efficiency, each node needs to select an appropriate value according to the network condition.

On the other hand, when inserting content with multiple keywords, we need an entry of hash table per keyword. Therefore, while retrieving the content with boolean search, the number of queries is proportional to the number of keywords.

7. Conclusion

In GNUnet, each intermediary replaces the original sender ID of query. Then, an attacker cannot ascertain the initiator. However, Kügler designated that because the query is conveyed to the nodes selected at random or according to previous transactions, the attacker can finds out the initiator by conducting the statistical test [5].

In this paper, we proposed a routing protocol, which is more efficient and free from Kügler's shortcut attacks. Specifically, by applying a notion of distributed hash table and giving certain directionality to each query, we can improve the anonymity of initiator. On the other hand, an attacker can find out the content holder by guessing the keyword. The future work is to contrive a protocol which has anonymity of content holder and is tolerative to censorship.

References

[1] K. Bennett and C. Grotho. Gap - practical anonymous networking. In *Proceedings of Workshop on Privacy Enhancing Technologies*, pages 141–160. Springer-Verlag.

[2] K. Bennett, C. Grothoff, T. Horozov, and J. Lindgren. An encoding for censorship-resistant sharing. avaiable at http://www.ovmj.org/GNUnet/download/ecrs.ps, 2003.

[3] K. Bennett, C. Grothoff, T. Horozov, and I. Patrascu. Efficient sharing of encrypted data. In *Proceedings of Australasian Conference on Information Security and Privacy 2002*, pages 107–120, Jul. 2002.

[4] R. Ferreira, C. Grothoff, and P. Ruth. A transport layer abstraction for peer-to-peer networks. In *Proceedings of IEEE/ACM Third International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems*, pages 398–405, May 2003.

[5] D. Kügler. An analysis of gnunet and the implications for anonymous, censorship-resistant networks. In *Proceedings of Workshop on Privacy Enhancing Technologies (PET 2003)*, pages 161–176. Springer-Verlag.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM 2001 Conference*, pages 161–172, Aug. 2001.

[7] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001 Conference*, pages 149–160, Aug. 2001.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: T U MUENCHEN. Downloaded on September 9, 2009 at 10:58 from IEEE Xplore. Restrictions apply.