International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **6**, June **2013**

 ISSN 1349-4198
pp. **2327–2346**

AN ANONYMOUS UNIVERSAL RELAY ARCHITECTURE OVER STRUCTURED PEER-TO-PEER NETWORKS

Jiunn-Jye Lee, Li-Yuan Lee and Chin-Laung Lei

Department of Electrical Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd., Da'an Dist., Taipei City 106, Taiwan
{ jye; conky }@fractal.ee.ntu.edu.tw; lei@cc.ee.ntu.edu.tw

Received March 2012; revised July 2012

Abstract. *In this paper, we proposed a Crowds-like architecture to achieve the goal of sender anonymity over existing structured peer-to-peer networks. Users are able to anonymously request both internal peer-to-peer services and external Internet ones via a relay path. One appealing feature of the architecture is the ease of integration, that is, the anonymous path is constructed without modifications to the underlying peer-to-peer networks. Moreover, an optional feature called dummy path composition is provided to resist local traffic analysis attack, and there are also two kinds of routing methods with different degree of anonymity to fulfill user requirements. Through analysis and simulations, we show that our system outperforms existing systems on both performance and anonymity.*
Keywords: Anonymity, Structured peer-to-peer network, Crowds-like, Anonymous network, Network security

1. **Introduction.** From file sharing, instant messages to Internet telephony, applications on peer-to-peer networks have been growing rapidly and form a trend of network usage. With the increased amount of private information being exposed to the Internet by peer-to-peer applications, the importance of data security and privacy protection cannot be omitted, especially if the military communication and important business transcations. However, since a member node in a peer-to-peer network may act as a server and a client simultaneously, existing solutions on the traditional client-server architecture may not function correctly. Thus, new algorithms and architectures are proposed to deal with these problems.

In the region of privacy protection, anonymity plays a basic but decisive role. Data encryption is an effective way to protect data from eavesdroppers; however, private information, such as timestamp of a request and IP addresses of both client and destination server, are still exposed. Therefore, anonymity mechanisms are designed to prevent the reveal of these private information. In 2000, Pfitzmann and Köhntopp [1] gave a clear definition of anonymity related terminologies:

```
Anonymity is the state of being not identifiable within a set
of subjects, the anonymity set. The anonymity set is the state
of all possible subjects who might cause an action.
```

For a message being transmitted in a network, there are typically one sender, one or more receivers (or responders) and maybe some additional cooperative nodes. Therefore, based on the target to which an anonymous scheme wants to protect, research objectives in a peer-to-peer network can be divided into three categories: sender anonymity, receiver anonymity and sender-receiver unlinkability. In this paper, we put our effort on sender

anonymity and proposed a new architecture named AURA (which stands for Anonymous Universal Relay Architecture), a Crowds [2]-like multi-proxy architecture over structured peer-to-peer networks. By wisely employing the characteristic of structured peer-to-peer networks, AURA does not require a trusted third party to achieve anonymity, and it can be deployed on any existing structured peer-to-peer scheme without affecting the original functionality of the underlying network. In other words, only when a user wants to build up an anonymous connection will AURA be invoked. The most outstanding feature of AURA is a disguise technique called dummy path composition (DPC) that can be used to hide the sender in a composite path and make him unidentifiable to a local eavesdropper. DPC provides pretty good resistance to a local eavesdropper and even to a certain amount of collaborators in the same network. By taking the trade-off between routing efficiency and size of anonymity set, AURA supplies two kinds of deployment policies: locate by routing (LBR), and locate from entry (LFE). Although the underlying structured peer-to-peer network may affect the degree of anonymity in some cases, analysis results show that AURA still provides pretty good sender anonymity.

The rest of this paper is organized as follows. Section 2 introduces some previously proposed anonymous solutions and gives the definitions of anonymity. Detailed algorithm of AURA is described in Section 3. In Section 4 and Section 5, analysis on both performance and degree of anonymity are discussed. And then we give a conclusion and corresponding future improvements in Section 6.

2. **Related Works.**

2.1. **Anonymous peer-to-peer networks.**

A well-known solution to achieve anonymity in traditional client-server architecture is the use of a trusted third party (TTP), such as anonymous proxy servers. Through the relay of an anonymous proxy server, end server or eavesdroppers cannot identify the real originator of a message. This concept is also widely used in many anonymous peer-to-peer schemes, but the server itself may become the performance bottleneck and even the single point of failure. Mix-Net [3] uses multiple proxies to send e-mail anonymously to avoid the problems of TTP servers. Although peer-to-peer networks is not mentioned in the paper, the peer concept is implicitly included in.

After the publishing of Mix-Net, some relative researches such as the anonymous e-mail system Mixmaster [4] and Mixminion [5], tried to improve Mix-Net to gain maximal degree of anonymity. However, their modifications caused huge latency and were limited in some specific purposes. Onion Routing [6] used layered encryption to keep high anonymity, but the side effect of high computational power consumption lowered down the overall system performance. On the other hand, Crowds [2] inherited the spirit of multi-proxy architecture. By constructing a multi-hop path, Crowds achieved pretty good sender anonymity from the destination server's point of view. Taking network latency into consideration, neither Onion Routing nor Crowds adopted message reordering mechanism. This made them vulnerable to traffic analysis attack.

At the initial development stage of anonymous mechanisms, sender anonymity was the most concerned issue since it was quite sufficient for Web browsing and publishing. With the maturing of peer-to-peer networks, receiver anonymity also became a valued topic. Tor [7], the second-generation Onion Routing system, embedded a "hidden service" that any node in the system could provide a TCP service without revealing its own IP address. This is done by deploying proxy servers at the opposite side and service providers receive requests through these proxy servers.

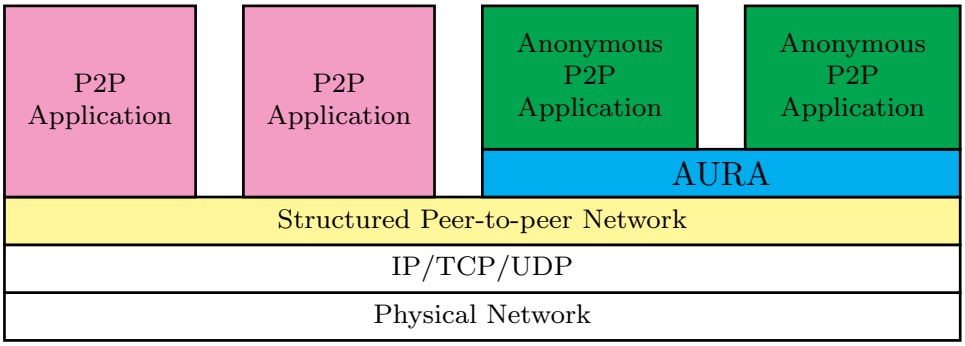


FIGURE 1. The layered architecture of AURA

However, most schemes mentioned above required a centralized directory server to maintain the whole system state, but the server may become the single point of failure and the weak point of intrusion. Unstructured peer-to-peer networks, such as Gnutella [8], flood message to discover resources. There are also some mechanisms, for example, Ants [9] and Freenet [10], designed to meet the goal of anonymity over unstructured peer-to-peer networks. Although these schemes solved the problems that came along with centralized directory servers, they encountered problems of bandwidth consumption, inefficient object locating, and scalability.

Unlike unstructured peer-to-peer networks, structured ones use a distributed hash table (DHT) [11] to allocate member nodes and resources; and therefore, eliminate the necessity of centralized directory servers. With proper design, each object with a key value can be located uniquely at a node. On the basis of some famous structured peer networks, such as Chord [12] and Pastry [13], many anonymous solutions were proposed. Achord [14], Neblo [15] and Agyaat [16] modify the routing algorithm of Chord to support anonymity, but these changes lead to poor performance. On the other hand, some schemes like Agyaat and Sason [17] introduced additional overlay networks to meet the goal, but at the same time the bandwidth overhead also grow up. Moreover, lots of anonymous schemes lack generality and flexibility, users in these systems have to use anonymous communications, no matter they want it or not.

Based on the former published architecture [18] which is a primitive model focused on sender anonymity against end servers, AURA improved the mechanism of building up a multi-hop path and handled the problem of path reconstruction when one or more relay nodes along a path failed. There are many attractive properties in AURA. First of all, the independent layered design, as shown in Figure 1, greatly reduced the cost of both integration and maintenance. In other words, no modification to the underlying structured peer-to-peer network is needed. On the other hand, AURA is triggered only when an anonymous application is invoked. It can coexist with many schemes focused on specific areas, such as load balancing schemes, performance enhancement schemes, and other security schemes. All that AURA needs from the underlying structured peer-to-peer network is the node discovering service, and any existing solution with such a service can be used as the underlying peer-to-peer network of AURA. This makes AURA more flexible and scalable than other anonymous schemes.

2.2. **Definitions of anonymity.** Different roles in a communication environment may have different anonymous requirements, and Free Haven [19] classified many anonymous requirements based on the roles of documents. Depending on the type of attacks, Cothtia and Chatzikokolakis [20] classified anonymity in a communication network into sender anonymity, responder anonymity, and sender-responder unlinkability. In addition, they

did not use anonymity set as the quantitative measure, but instead used a more classical classification. In the paper of Crowds, the degree of anonymity was divided into six levels. In practical, most existing anonymous schemes can be classified into one of the three following degrees:

Beyond suspicion: A sender's anonymity is beyond suspicion if though the attacker can see evidence of a sent message, the sender appears no more likely to be the originator of that message than any other potential sender in the system.

Probable innocence: A sender is probably innocent if, from the attacker's point of view, the sender appears no more likely to be the originator than to not be the originator.

Possible innocence: A sender is possibly innocent if, from the attacker's point of view, there is a nontrivial probability that the real sender is someone else.

There are still various types of anonymity metrics. For example, Díaz et al. [21] used entropy to measure anonymity. Different metric reflects different viewpoint of anonymity. In Section 5 anonymity set and Crowds' classification described above will be used as the anonymity metrics of AURA.

In addition to the anonymity property, the type of attackers is also an issue that system designers should pay attention to. In most previous researches, attackers are assumed to be passive observers. That is, an attacker can only break the anonymous property through the way of information collection. In this case, attackers are divided into three categories.

Local eavesdropper: An attacker who can observe all (and only) communication to and from the user's computer.

Collaborative members: Other members that can pool their information and even deviate from the prescribed protocol.

End server: The destination server to which the web transaction is directed.

A receiver can be an attacker from the sender's point of view. Contrarily, the sender may also be an attacker of the receiver. In addition, an attacker who observes all messages over network is called a global attacker. Tarzan [22] analyzed if the information is exposed to the first, last and other intermediate relays.

The main design goal of AURA is to achieve sender anonymity with at least probable innocence degree. By using a relay-based method that is similar to Crowds, AURA provides good sender anonymity to both receiver and other nodes. For those applications that require extra high degree of anonymity, dummy path composition can be activated to support resistance of the local traffic analysis attack. Restricted by the nature of structured peer-to-peer networks, that is, a node ID responsible for a specified item key is always fixed or limited in some special range, receiver anonymity cannot be integrated without considerable modifications to the underlying structured peer network. Therefore, receiver anonymity is not included as a design factor.

3. The Proposed Scheme.

The layered structure of AURA has shown in Figure 1. A well-configured structured peer-to-peer network which enables AURA to locate a node with a specified key value is required. If a user wants to send a message anonymously, he has to build up a relay path first. The path construction procedure is quite similar to that of Crowds. Figure 2 shows an example of relay paths in AURA. Here the sender S has built two paths, $Path_1 :< S, N_1, N_2, N_3 >$ and $Path_2 :< S, N_4, N_5, N_6 >$. Messages sent along the path will be relayed by the intermediate nodes, such as N_1, N_2, N_4 and N_5. On receiving a message sent by the sender, the end node of the path examines the content of the message to see whether it requests for an internal peer-to-peer service or an external Internet service. In Figure 2, N_3 receives a external request Req_1, so he sends

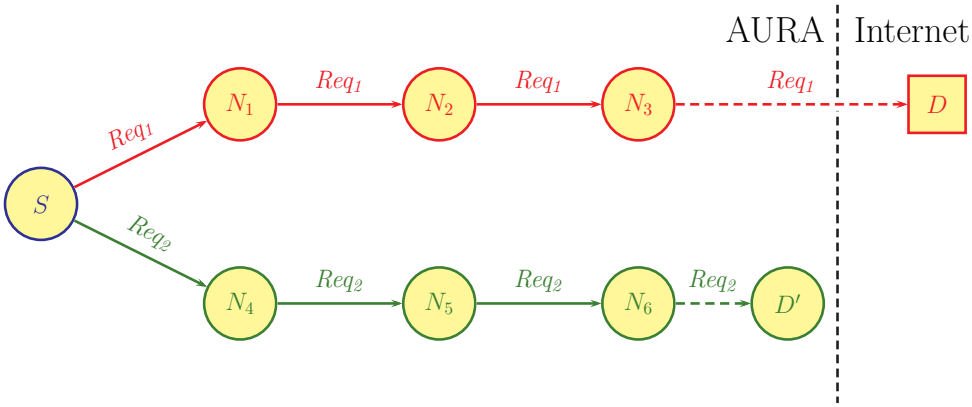


FIGURE 2. An illustration of relay paths in AURA

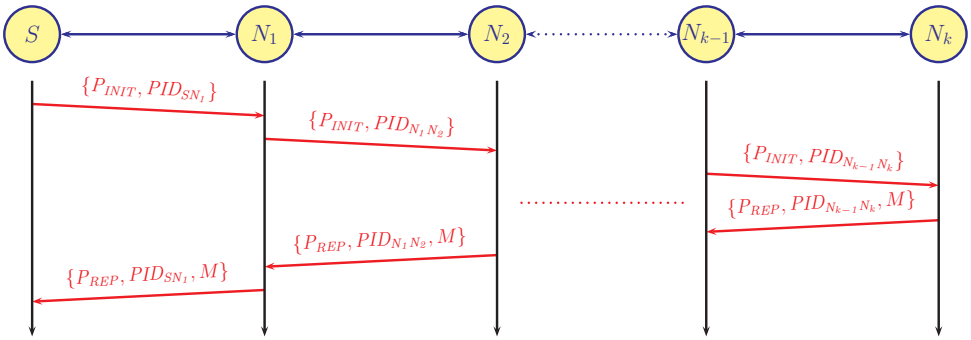


FIGURE 3. The path construction message flow in AURA

request Req_1 to the destination server D directly. On the other hand, S sends Req_2 along $Path_2$ to acquire some peer-to-peer resources, so the request Req_2 will be transmitted to the destination peer node D' by the end node N_6.

3.1. **Path construction phase.** Message flow of AURA's path construction procedure is shown in Figure 3. Initially the source node S sends message $\{P_{INIT}, PID_{SN_1}\}$ to a randomly selected peer node, say N_1, from his candidate set. Here P_{INIT} implies a path initialization message and PID_{SN_1} is a random number indicating the path ID between S and N_1. According to the deployment policy, the candidate set may be the whole peer space or a routing table or even a customized one. When N_1 receives the message, he decides either to be an intermediate relay node with probability p_f or an end node with probability $1 - p_f$. The value of the forwarding probability p_f should be set to greater or equal than 0.5. If N_1 determines to be a relay node, he will randomly select another peer node N_2, generate a random number $PID_{N_1N_2}$ and sends $\{P_{INIT}, PID_{N_1N_2}\}$ to N_2. Each subsequently chosen node N_i repeats the same procedure until an end node appears. The end node node, say N_k, returns a reply message $\{P_{REP}, PID_{N_{k-1}N_k}, M\}$ to his predecessor N_{k-1}, where $M = \{N_k, PID_{N_kS}\}$. The returned message will be sent through the path and reach the source node S. Thereafter an anonymous path is constructed.

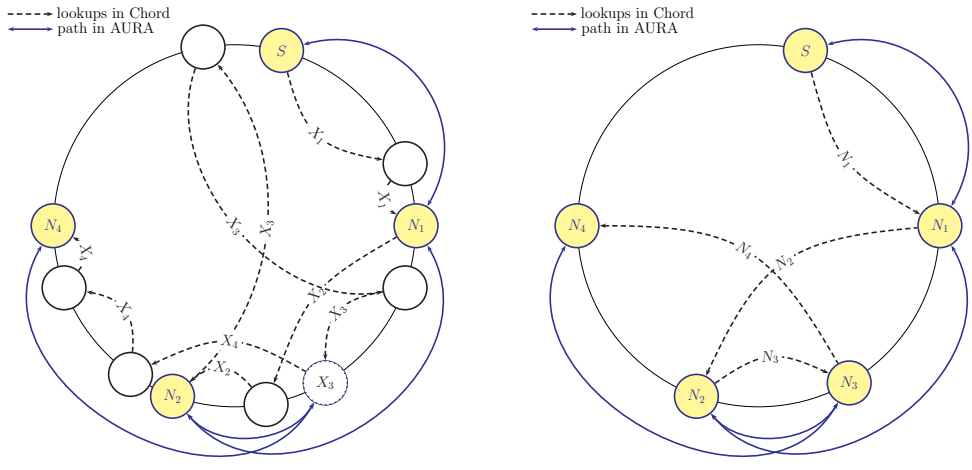
Each node in AURA maintains a relay table (RT). A record in a RT is of the form $\{ROLE, N_{prev}, PID_{N_{prev}}, N_{next}, PID_{N_{next}}\}$, where field $ROLE$ should be set as either $SOURCE$, $RELAY$ or END, and the rest four fields denote the corresponding node and path ID of the predecessor and successor nodes in the path. In Figure 3, after receiving the returned message, each intermediate node N_i records $\{RELAY, N_{i-1}, PID_{N_{i-1}N_i}, N_{i+1}, PID_{N_iN_{i+1}}\}$ in his own RT. The record in source node S's RT will be $\{SOURCE, N_k, PID_{N_kS},$

$N_1, PID_{SN_1}\}$. The returned message M is written in fields N_{prev} and $PID_{N_{prev}}$, and will be used in path reconstruction procedure if one or more nodes along current path fail. On the other hand, from the end node N_k's point of view, fields N_{next} and PID_{N_kS} should be filled with information shared between himself and source node S. However, since N_k does not have any information about S, N_{next} will be marked as $NULL$. Therefore, the N_k's RT record will become $\{END, N_{k-1}, PID_{N_{k-1}N_k}, NULL, PID_{N_kS}\}$.

Once a path is constructed, each pair of adjacent nodes share a session key, including the end node and the source node pair. Since a trusted third party (TTP) is not a constraint of AURA, a key exchange scheme such as the public key infrastructure, is required. Taking simplicity and efficiency into consideration, we adopt the famous Diffie-Hellman key exchange protocol [23] as the default key exchange scheme.

3.2. **Node-discovering methods.** A key point of the path construction procedure lies on the way to find the next relay node. AURA itself does not provide such functionality and the mission is handled by the underlying structure peer-to-peer network. Based on the common capabilities of most structured peer networks, two relay mechanisms are proposed. To illustrate the design philosophy of AURA in a more realistic way, here we choose Chord as the underlying peer network.

Locate by Routing (LBR): During the path construction phase, when a node, say N_i, decides to be an intermediate relay node of a path, he will choose a node ID X at random and pass X to Chord using the API provided by Chord. After a series of lookups, Chord will return node X's information or the closest online successor node's if node X does not exist. Methods to find the responsible node may vary between different peer-to-peer networks, but they should always return a unique node. The returned node will become the successor node N_{i+1} in the path, and subsequent communication will be transmitted directly between N_i and N_{i+1}. An example of LBR is shown in Figure 4(a). To find the first and second relay nodes, Chord each invokes two lookups to find N_1 and N_2 that are responsible for node ID X_1 and X_2, respectively. On receiving the demand of finding a node with ID X_3, Chord takes three lookups but the node with ID X_3 does not exist. Chord then returns the closest successive node to AURA , and the node will be regarded as N_3. Since the next relay node is chosen from the whole peered space, the size of candidate set is equal to number of nodes in the system.



(a) LBR method (b) LFE method

FIGURE 4. Examples of LBR and LFE methods in AURA-Chord

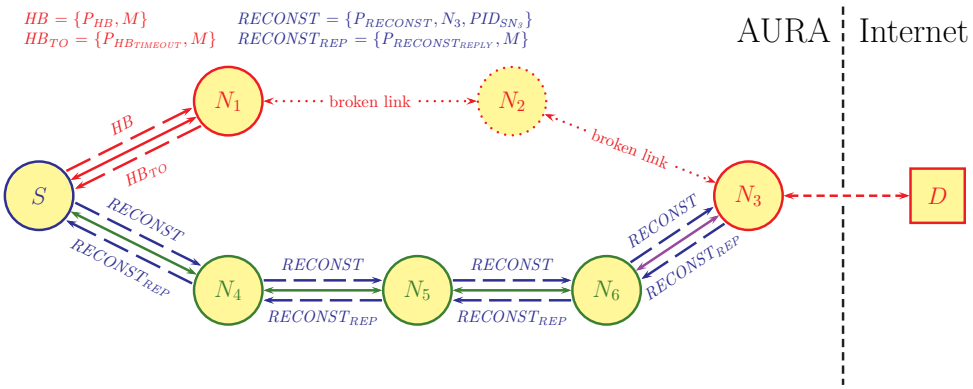


FIGURE 5. An example of path reconstruction in AURA

Locate from Entry (LFE): To increase the efficiency of node discovering, most structured peer-to-peer networks adopt routing tables to cache those members that help increase the efficiency of finding other nodes. Instead of searching the whole key space, in LFE a node N_i randomly picks one entry from his routing table as the next relay node N_{i+1}. Since information of N_{i+1} has already been cached in N_i's routing table, N_i is able to directly contact N_{i+1} without invoking search routine of the network. Figure 4(b) is an example of LFE. The routing table in Chord is called a finger table. When node S wants to build a path, he first chooses a target N_1 from his finger table randomly and forwards the path construction request directly to it. N_1 and those subsequently chosen nodes also repeat the same procedure, until finally a node decides to be the end node of the path.

3.3. **Path reconstruction mechanism.** The low-latency design makes AURA capable of supporting anonymous interactive network connections such as TCP/UDP applications. Unfortunately, an anonymous path may be broken due to one or more relay nodes' leave or failure. One way to solve the problem is to abandon the broken path and rebuild a new one, but all transactions and data connections to the destination servers will also be dropped as a matter of course. However, as long as the end node keeps alive, it is not necessary to rebuild everything.

Figure 5 illustrates an example of AURA's path reconstruction mechanism. For simplicity, some message fields such as the path ID between S and N_4 (PID_{SN_4}) will not be shown in the figure. Initially node S has two paths, $Path_1 :< S, N_1, N_2, N_3 >$ and $Path_2 :< S, N_4, N_5, N_6 >$. S is communicating with an external server D through $Path_1$ and meanwhile $Path_2$ is temporarily unoccupied. To keep track of each path's status, heartbeat messages are periodically sent along the path. If a node, say N_2, fails suddenly, a timeout status will be returned to S. To resume the connection between S and D, first S examines the paths of his own and tries to find a free one. If such a path cannot be found, S has to create a new one. In our example $Path_2$ is unoccupied; therefore, S sends a path reconstruction message $\{P_{RECONST}, N_3, PID_{N_3S}\}$ through $Path_2$, where fields N_3 and PID_{N_3S} uniquely identify the end node of $Path_1$. After the end node of $Path_2$, that is, N_6 receives the recover message, he will try to contact N_3 and forwards the path reconstruction message to him. If N_3 is still alive, he will use N_6 to replace the original predecessor node N_2 in $Path_1$ and return a reply message of the form $\{P_{RECONST_{REPLY}}, M\}$. The message M is encrypted by the session key shared between S and N_3, and contains the information kept since $Path_1$ has broken. When the returned message arrives at S, S can resume the communication between himself and D. The original end node N_6 of $Path_2$ will change his role to an intermediate relay node, and exchange necessary information

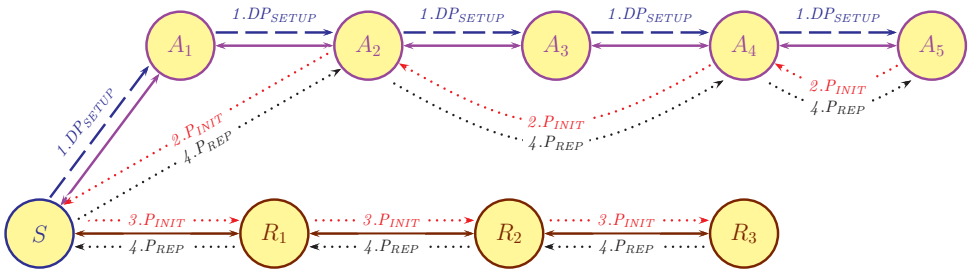


FIGURE 6. An overview of the DPC initialization

with the new end node N_3. $Path_1$-related records stored on relay nodes along the original path will be discarded.

The path reconstruction mechanism will only work when the end node of the second path successfully contacts the end node of the broken one. Otherwise, all the sender can do is to restart the whole connection through another anonymous path.

3.4. **Dummy path composition.** The anonymous path described above provides good sender anonymity against the end server and the collaborative nodes, but a sender is still exposed if his traffic has been monitored by an eavesdropper. Therefore, for those applications that require extra high degree of sender anonymity, AURA provides an appealing mechanism called dummy path composition (DPC).

A local eavesdropper identifies a sender by analyzing the incoming and outgoing traffic of a node. The design philosophy of DPC is to forge a fake sender and make the real one act just like a relay node, and as a result, the local eavesdropper will be misled. To achieve the goal, DPC combines two paths initiated by the sender to form a composite path (CP) and disguise the sender as an intermediate relay node. In the following paragraph, the path used to transfer real data is called the real path (RP) and the other one used to balance the traffic and disguise the sender is called the dummy path (DP).

3.4.1. *DPC initialization.* We use Figure 6 as an example to describe the basic flow of a DPC initialization. The initialization procedure can be divided into four steps:

1. S sends $DP_{SETUP} = \{JUMP_{INIT}, F_{JUMP} = TRUE, \{DP_{INIT}\}_{K_{A_5S}}\}$ to an unoccupied anonymous path $Path_A :< S, A_1, \ldots, A_5 >$. $JUMP_{INIT}$ is a command that notifies nodes along $Path_A$ to send or record the information of the "jump predecessor (JP)". If the jump flag F_{JUMP} is set to $TRUE$, a node A_i will turn it to $FALSE$ and forward the message to A_{i+1} with node ID A_{i-1} and path ID $PID_{A_iA_{i-1}}$ appended. On the other hand, if $F_{JUMP} = FALSE$, A_i will turn the flag to $TRUE$ and record the information of A_{i-2} sent from A_{i-1} as the jump predecessor JP_i. As a result, node A_2 records S as his JP and A_4 records A_2, respectively. The last field $\{DP_{INIT}\}_{K_{A_5S}}$ in DP_{SETUP} is encrypted by the key shared between A_5 and S and is used to notify A_5 to create a dummy path.
2. When the end node A_5 receives DP_{SETUP}, he decrypts $\{DP_{INIT}\}_{K_{A_5S}}$ and sees the dummy path initialization command. Then A_5 issues a "JUMP" P_{INIT} request. In Section 3.1, the PID field in the path construction phase should be filled with a randomly generated number. In DPC, the PID field will be replaced by the one recorded in the node's JP field or by the one shared with his predecessor if the JP field is empty. Then this P_{INIT} message will be transmitted to the jump predecessor of the node or to the immediate predecessor if the node has an empty JP field. Since in Figure 6 A_5 does not have a JP record, he fills $PID_{A_5A_4}$ in the P_{INIT} request and sends it along $Path_A$. During the transmission, node A_4 discovers that he has a

JP record pointing to A_2; therefore, he will set the PID field of the "JUMP" P_{INIT} request to $PID_{A_3A_2}$. Then A_4 skips A_3 and directly forward the "JUMP" request to A_2. Similarly, A_2 will contact S directly and skip the node A_1. This "JUMP" behavior will form a new path $Path_J :< S, A_2, A_4, A_5 >$. Each pair of nodes along the jump path should also exchange the related path information as a matter of course.

3. S starts to build an normal anonymous path after retrieving the P_{INIT} message from A_2. Since S doest not contain any corresponding successor information, a new anonymous path construction procedure will be performed, and in Figure 6 $Path_R :< S, R_1.R_2, R_3 >$ is built.
4. The returned message P_{REP} will be transmitted through $Path_R$ and $Path_J$, and a composite path (CP) is successfully constructed, where $Path_R$ is the real path RP that transfers requests of sender S, and $Path_J$ is the dummy path DP that accounts for balancing the incoming and outgoing traffic along CP. The node A_5 turns out to be the dummy sender of CP.

From the viewpoint of a local eavesdropper who monitors S, the message flow in step 1 has no difference to a single request sent from node S to A_1. Messages handled by S in steps 2 to 4 are all paired and of the same size, this makes S looked like a relay node between node A_2 and node R_1. Thus a composite path is created unsuspiciously.

The "JUMP" behavior is the key point of the DPC initialization procedure. It provides another path that looks like being initialized from a node other than S; and therefore, cheats the local eavesdropper monitoring S. However, if the original $Path_A$ has only one relay node, say A_1, this mechanism will fail. Thus if S discovers that the new successor node along DP is the same as the previous one, that is, A_1 in this case, he shall abandon this path. In addition, node S, A_1, and A_2 should not be in the same local network, otherwise they may be monitored by the same eavesdropper and the trick used in DPC will be disclosed.

3.4.2. *Data transmission over DPC.* Figure 7 shows the message flow of a typical request/reply procedure over DPC. A heartbeat message is sent periodically by the dummy sender D_j. The heartbeat message is of the form $\{P_{HB}, \{DP_{NO}, X_1\}_{K_D}\}$, where P_{HB} is the heartbeat command, and $\{DP_{NO}, X_1\}_{K_D}$ is a message encrypted by the session key K_D shared between S and D_j. The DP_{NO} field in the message means no operation assigned and X_1 is a random number. When the real sender S wants to send a request, he has to wait for a heartbeat message, re-encrypt it by the session key K_R shared between S and R_k, forward it to the real path RP, and wait again until the reply heartbeat message arrives. Once S receives the reply message $\{P_{HB_{REPLY}}, \{DP_{NO}, X_1\}_{K_R}\}$, he replaces the message by $\{P_{HB_{REPLY}}, \{DP_{SR}, |M_1|\}_{K_D}\}$ and forwards it to the dummy path DP. The field DP_{SR} asks the dummy sender D_j to send a single dummy request and the size of the request should be $|M_1|$.

According to S's instruction, D_j generates a dummy request $DREQ_1$ containing dummy message DM_1 of size $|M_1|$ encrypted by K_D, and sends it to S along DP. On receiving $DREQ_1$, S directly swap the request by the one he wants to send, that is, $\{REQ_1,\{M_1\}_{K_R}\}$ and forwards it to R_k along CP. Finally, the request will be processed by the end node R_k of RP and sent to the destination server.

To keep S behave as a relay node, after receiving the returned data, S should also forward a dummy reply message of the same size to D_j. If S has subsequent requests to be sent, he may inform D_j by embedding messages in the dummy reply message $\{DREP_1, \{DDATA_1\}_{K_D}\}$.

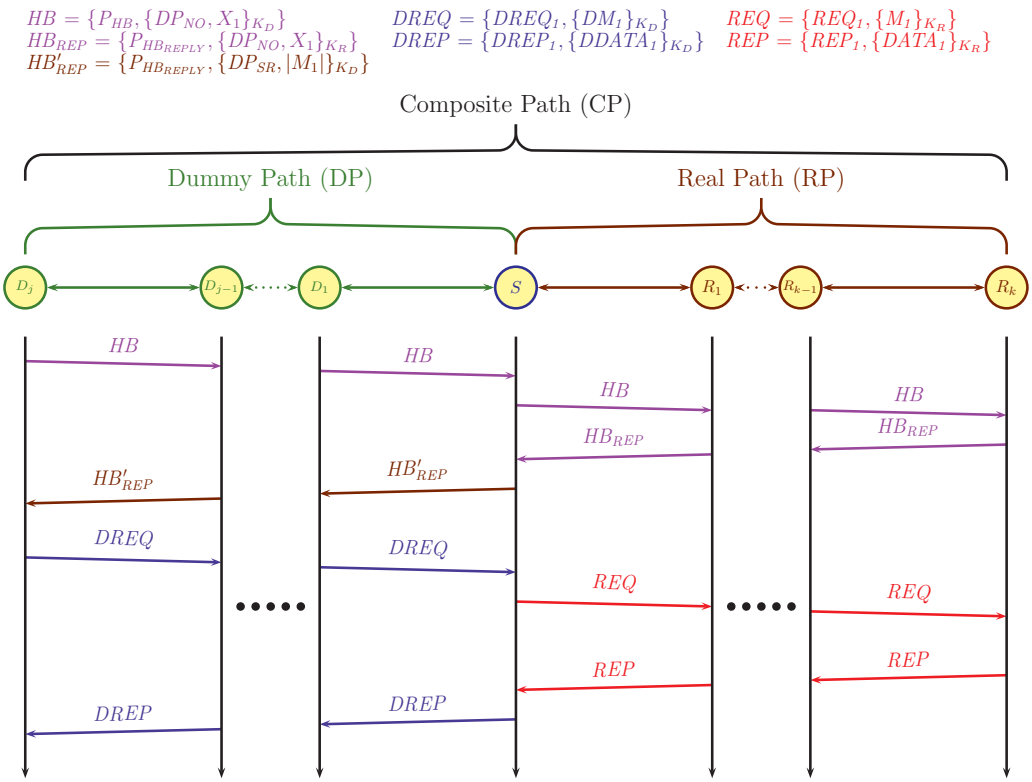


FIGURE 7. Dummy path deployment message flow

Since each pair of nodes in a path segment shared a session key, an incoming message will be decrypted by one key and then encrypted by another before being forwarded. In addition, sender merely swaps the dummy message by his own request of the same size. Therefore, a local eavesdropper cannot detect the content changes by monitoring the incoming and outgoing messages of the sender node. As a result, the sender will become unidentifiable to a local eavesdropper.

The path reconstruction scheme can also be used when a composite path is broken as long as the end node of the real path is still connectable. However, it should be processed with more care. According to the position of the failed node, there are several policies that can be considered. If the breakpoint is on the dummy path, the sender can only create another composite path and then reconnect the new end node to the old one. It is because that without the support from the dummy sender, the real sender cannot do any action. Otherwise the local eavesdropper may lock down the sender. On the other hand, if the breakpoint is on the real path, the sender can either create a new composite path or instruct the dummy sender to initialize an "append" P_{INIT} request to form a new path appended to the breakpoint, and then reconnect the end node of the old real path.

4. **Performance Analysis and Simulation Results.** In this section we analyze the performance of AURA from many aspects. To show the effectiveness of AURA, we also executed a series of simulations. In our settings, there are 100 nodes sending requests at a mean poisson rate 30 seconds. The type of a request is randomly chosen from REQ, DPC, REQ_{NEW}, and DPC_{NEW}. A request of type REQ or DPC will be sent through an existing path while request of type REQ_{NEW} and DPC_{NEW} will require the construction of a new path. If a node has no existing path, REQ will be treated as REQ_{NEW} and

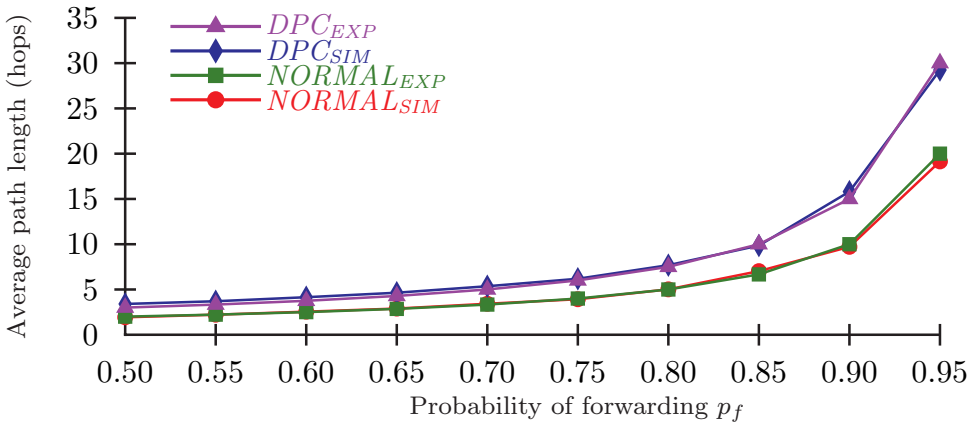


FIGURE 8. Average path length in AURA

DPC as DPC_{NEW}. The downlink/uplink bandwidth of a node is randomly set as either 100Mbps/100Mbps or 10Mbps/10Mbps or 2Mbps/512Kbps or 1Mbps/64Kbps. The default p_f is set to 0.75, I_{HB} to 500 ms, and message size to 1KByte, respectively.

4.1. **Cost of path construction.** In this paper, the path length is counted as number of hops a message will be sent through; therefore, the expected path length L will become

$$L = (1 - p_f) \sum_{i=0}^{\infty} (k+1) p_f^k = \frac{p_f}{(1 - p_f)} + 1$$

A composite path is composed of a dummy path and a real path. Since the "JUMP" step described in Section 3.4.1 half cuts the dummy path, the expected length L_{CP} of the composite path is equal to $1.5L$. Figure 8 shows the average path length of both normal path and composite path cases. $NORMAL_{EXP}$, DPC_{EXP} represent the expected value L and L_{CP}, and $NORMAL_{SIM}$ and DPC_{SIM} are averaged value of the simulation.

AURA's path construction procedure is almost the same as that in Crowds except the way to find the next relay node. Rather than one query to the centralized directory server in Crowds, the cost of finding a next relay node in AURA depends on the underlying structured peer-to-peer network. In the following paragraph, we take Chord as the underlying peer-to-peer network to show the path construction cost (number of lookup hops) of the two routing methods in AURA.

Locate by Routing: Let N be the total number of nodes in the system, the average number of hops to find a node in Chord is equal to $\frac{1}{2}\log_2 N$. In LBR, each relay node is discovered by Chord's lookup routine. Therefore, for an anonymous path with expected path length L, the average path construction cost of LBR will be $\frac{1}{2}L\log_2 N$ hops.

Locate from Entry: In LFE a node chooses the next relay candidate from entries of the routing table. Since the IP address of each entry node is already recorded, a direct connection can be built without extra lookups. Thus, the path construction cost of LFE is exactly L hops.

Figure 9 shows the comparison of LBR and LFE. The cost expected path length will be affected by the system size, on the other hand LFE does not require the node-discovering procedure and thus the number of hops required to construct a path is exactly the same as the path length.

Most structured peer-to-peer networks implemented based on the concept of binary search cost $\log_2 N$ on average to discover a node; and therefore, the cost of LBR shown

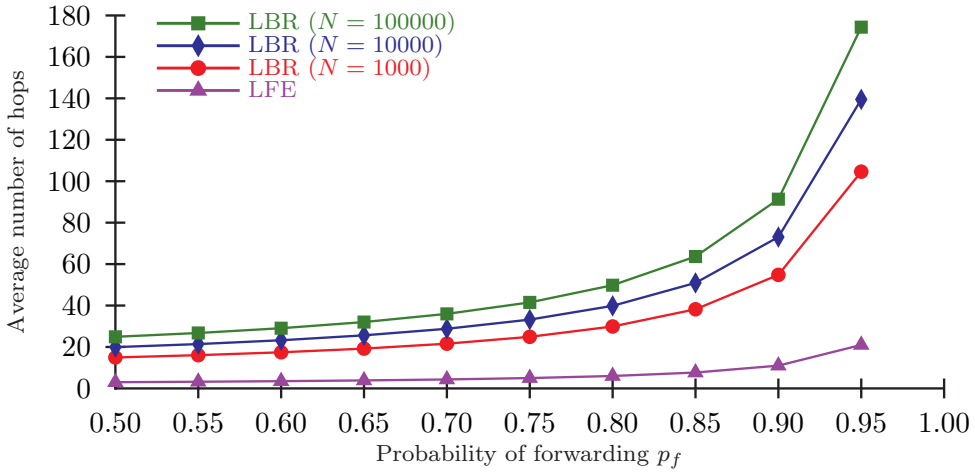


FIGURE 9. Average lookup path length of LBR and LFE in AURA-Chord

above can be viewed as a representative measure. With specific underlying structured peer-to-peer network, a system designer may develop his own node-discovering method to gain more benefit. Again taking Chord as an example, a hybrid method may be designed as follows.

Hybrid Method over Chord: To find the next relay node, a node N_i first chooses a random number $0 \leq q \leq Q-1$, where $0 \leq Q \leq \log_2 N$ is a predefined constant. Then N_i selects q non-repeated random numbers $R_0, R_1, \ldots, R_{q-1}$ where $0 \leq R_j \leq \log_2 N$. The next relay node ID $N_{i+1} = N_i + 2^{R_0} + 2^{R_1} + \ldots + 2^{R_{q-1}}$. By the binary search nature of Chord, N_{i+1} can be found with exactly q lookups and the candidate set will become $\sum_{j=0}^{Q-1} C_j^{\log_2 N}$. When Q is set to $\log_2 N$, this is exactly the LBR method. On the other hand, when $Q = 1$ it will reduce to the LFE method.

4.2. **Cost of path reconstruction.** The way to reconstruct a path in AURA is to build a new one and then connect it to the end node of the old path; therefore, the cost of path reconstruction can be divided into two parts: the cost of constructing a new path and the cost of connecting to the old end node. Since the end node of the old path is unlikely to appear in the routing table of the new path's end node in most cases, a lookup query to the underlying structured peer-to-peer network is necessary. Therefore, the cost of finding the old end node is equal to $\log_2 N$. Combined with the analytical results shown in the previous section, the average path reconstruction cost in LBR method is $(\frac{1}{2}L + 1)\log_2 N$, and $L + \log_2 N$ hops in LFE. However, if the IP address of the old end node is also kept by the sender, this $\log_2 N$ cost can be reduced to one hop.

4.3. **Interval of heartbeat messages.** If DPC is used, interval of the heartbeat message I_{HB} becomes another factor that affects the latency perceived by the user. When a user wants to issue a request through the composite path, he has to wait $\frac{I_{HB}}{2}$ seconds on average to capture a HB_{REP} message and embed instructions in it. A small I_{HB} lowers the average waiting time but at the same time increases bandwidth consumption of the network. However, in a densely request-reply communication, every reply message can be used to embed subsequent instructions and forwarded to the dummy sender, and the necessity of waiting for a heartbeat message can be diminished.

In practice, for a request that does not specify a dummy path, the waiting time can be further lowered to $\frac{I_{HB}}{p+1}$ if sender holds p dummy paths. In Figure 10, each DPC request in "Chosen DP" case waits for a HB_{REP} message of a specific path, and "Random DP" means

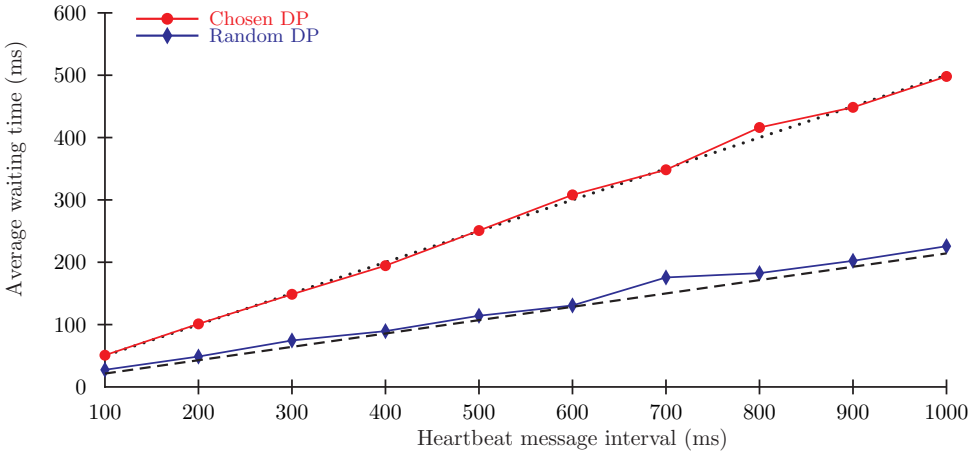


FIGURE 10. Average heartbeat message waiting time in DPC

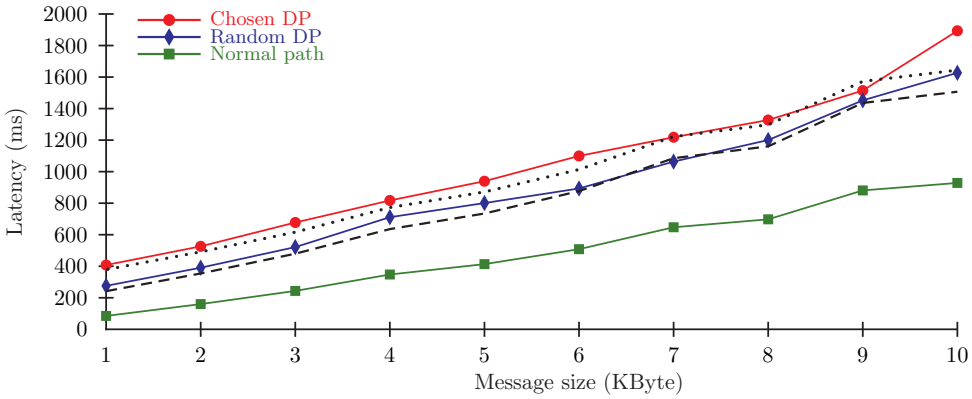


FIGURE 11. Transmission latency in AURA

that a DPC request use the first arrived HB_{REP} message to embed necessary information. Since each path is created at run time, the expected waiting time of "Random DP" case will become $(\frac{1}{k}\sum_{i=1}^{k}\frac{1}{i+1})I_{HB}$, where k is the number of paths of a node. In our simulation each node constructs about 9 composite paths on average ($k = 9$), and the dashed line in Figure 10 shows this expected value. The expected waiting time in "Chosen DP" case is drawn in dotted line. Simulation results show that the time wasted on waiting for a HB_{REP} message can be effectively reduced if a request uses the first arrived HB_{REP} message when a node holds more than one composite path.

4.4. **Transmission latency.** Figure 11 shows the message transmission latency of both normal relay paths and composite paths. The transmission latency of a DPC request $T_{DPC} = t_w + t_h + t_d + t_m$, where t_w, t_h, t_d, and t_m represent the time waiting for a HB_{REP} message, the time of a HB_{REP} message sent from sender to the dummy sender, the dummy request transmission time, and the request transmission time, respectively. Since the size of a dummy request is the same as the DPC request and $L_{CP} = 1.5L$, $t_d + t_m = 1.5\,T_{NORMAL}$, where T_{NORMAL} is the transmission latency of a normal request. Because the size of a HB_{REP} message is generally much smaller than that of a request, for the convenience of estimation, we assume that t_h is also much smaller than t_d. Thus we have $T_{DPC} = t_w + 1.5\,T_{NORMAL}$. Based on the simulated values of T_{NORMAL}, estimated latency T_{DPC} are plotted as a dotted line ("Chosen DP") and a dashed line ("Random DP") in Figure 11. Simulation results are close to the estimated values. This shows

TABLE 1. Comparisons between Crowds, Tor, and AURA

	Crowds	Tor	AURA_{LBR}	AURA_{LFE}	$\text{AURA}^{\text{DPC}}_{\text{LBR}}$	$\text{AURA}^{\text{DPC}}_{\text{LFE}}$
Path construction	$L+1$	4	$\frac{1}{2}L\log_2 N+1$	$L+1$	$\frac{3}{4}L\log_2 N+1$	$L+1$
Data path length	$L+1$	4	$L+1$	$L+1$	$\frac{3}{2}L+1$	$\frac{3}{2}L+1$

TABLE 2. Comparisons between Achord, Agyaat, Sason, AP3, and AURA

	Achord	Agyaat	Sason	AP3	AURA_{LBR}	AURA_{LFE}
Path construction	$\frac{1}{2}\log_2 N$	$\frac{1}{2}\log_2 N+2r_{diam}$	$\frac{1}{2}\log_2 N+2L$	$(L+1)\log_2 N$	$\frac{1}{2}(L+1)\log_2 N$	$\frac{1}{2}\log_2 N+L$
Data path length	$\frac{1}{2}\log_2 N$	$2r_{diam}+1$	$2L+1$	$L+1$	$L+1$	$L+1$

that the latency of a DPC request can be estimated, and system designers may adjust parameters such as p_f and I_{HB} to keep the transmission latency in an acceptable range.

4.5. **Comparisons with previous researches.** In this section we compare AURA with some famous former schemes. Since AURA supports both internal peer-to-peer services and external internet services, each type of service will be discussed and compared separately.

4.5.1. *Schemes for external services.* For anonymous peer networks that support external internet services such as HTTP services, the address of the destination server is usually a known information to the sender. Since the end node of an anonymous path is able to contact the destination server directly, the path length between the end node and the destination server can be treated as one hop. System performance of AURA is very similar to Crowds. With the same probability of forwarding p_f, the expected length from a sender to the destination server is equal to $L+1$ in both Crowds and AURA. Since Crowds maintains a centralized directory server, node discovering can be done in exactly one query. Therefore, the cost of path construction including the last segment from the end node to the remote server is still $L+1$ hops, which is better than AURA_{LBR}. On the other hand, Tor uses layered encryption to control the path length and the path length is currently fixed to 3, which is roughly the same as the expected length of Crowds and AURA when the probability of forwarding is set to 0.5. The cost of path construction in AURA_{LBR} and $\text{AURA}^{\text{DPC}}_{\text{LBR}}$ may be higher than others, but it can be built beforehand and the actual data transmission is not affected. Path length in $\text{AURA}^{\text{DPC}}_{\text{LBR}}$ and $\text{AURA}^{\text{DPC}}_{\text{LFE}}$ may look a little bit longer, but they provide good sender anonymity against a local eavesdropper, which is not supported in both Crowds and Tor. Comparison results are listed in Table 1.

4.5.2. *Schemes for internal services.* In the layered design of AURA, a structured peer-to-peer network is required. Thus, we choose some famous structured schemes for comparisons. Achord relays traffic along Chord's lookup path, so the path construction cost and data transfer path length are both $\frac{1}{2}\log_2 N$. The data path length in Achord is determined by the number of nodes in the system; therefore, with the growth of system the data path will become longer and longer. However, data path length in AURA is unrelated to the system size and only determined by the probability of forwarding p_f. Thus the latency and bandwidth consumption perceived in Achord will be more serious than that in AURA.

The path length in Agyaat is affected by the cloud size. Let r_{diam} denotes the cloud diameter, the cost of path construction will be $\frac{1}{2}\log_2 N + 2r_{diam} + 1$, and the data transfer path length will be $2r_{diam} + 1$, respectively. In the paper of Agyaat, the value of r_{diam} is suggested from 3 to 9, which makes the expected path length close to the L_P in AURA. Agyaat sends request messages to the target cloud in a multicast way, this may increase the bandwidth consumption, but at the same time provides the feature of responder anonymity.

Sason requires an anonymous sender routing network. To compare Sason with AURA, here we combine Sason with Crowds. The cost of path construction in Sason is $\frac{1}{2}\log_2 N + 2L$ and the data transfer path is $2L + 1$. The path length for data transfer is twice of AURA's. This is because Sason use L hops for server's listener connection to ensure responder anonymity.

The mechanism of AP3 is similar to AURA, it use the same way as AURA to deliver messages from source node to the destination server. However, the reply message in AURA will go through the same path while in AP3 it is transferred through another anonymous channel. The way to construct an anonymous path is not shown clearly in the paper of AP3. We believe, however, it should be the same as that in AURA_{LBR} due to the DHT nature of structured peer networks. Since AP3 requires another anonymous channel to receive the reply message, the total cost of path construction should be $(L + 1)\log_2 N$. The data path length is the same as AURA.

The comparison results are shown in Table 2. In addition to path length, the extra overlay network may also cause overhead on both maintenance and data transfer latency. AP3 and Achord do not have any additional overlay network. Agyaat requires special Agyaat clouds and R-ring overlay networks, and Sason embeds an anonymous sender routing network. Compared with these schemes, AURA does not require any addition overlay network, even the underlying structured peer-to-peer network can be kept unchanged. From the user's point of view, AURA is only an add-on service that efficiently provides various degree of anonymous communication.

5. **Anonymity Analysis.**

In this section we use two kinds of metrics to analyze the anonymity properties that AURA achieves: the degree of anonymity and the size of sender anonymity set. Assume that the message content does not reveal any information about the message originator.

5.1. **Degree of anonymity.**

Based on the definitions in Section 2, three kinds of attacker are discussed: end servers, collaborative nodes, and a local eavesdropper.

End Server: Since in AURA the role to contact the end server is played by the end node of an anonymous path, from the end server's perspective, each node in AURA has equal chance to be the message originator. Therefore, the sender anonymity for AURA_{LBR}, AURA_{LFE}, and AURA_{DPC} are all beyond suspicion. As shown in Crowds' analysis, the probability of forwarding p_f does not affect the anonymity against end server.

Local Eavesdropper: A local eavesdropper is an passive attacker that monitors the incoming and outgoing traffic of a certain node. In most cases the sender sends outgoing requests without corresponding incoming messages, and subsequently will be exposed to an eavesdropper monitoring the sender. Like Crowds and many other schemes that supports TCP/IP applications, AURA_{LBR} and AURA_{LFE} have no extra mechanisms to balance the message flow of the sender, thus they cannot resist the traffic analysis attack.

On the other hand, by using a dummy path and a dummy sender to balance the traffic, AURA_{DPC} is able to fight against a local eavesdropper. The encryption/decryption processes between each segment of the path shuffle the message content and disable the ability of content analysis from the local eavesdropper. Timing attacks may reveal some clue but still can be diminished with simple cache and replace method. With careful design and implementation, the sender behaves the same as other relay nodes. Therefore, from the local eavesdropper's perspective, the sender looks more likely a relay node than a message originator, in other words, AURA_{DPC} is probable innocence to a local eavesdropper.

In opposition to sender anonymity, all the three mechanisms in AURA provide beyond suspicion receiver anonymity to the local eavesdropper. This is because the message sent to an end server is encrypted by the key shared between the sender and the end node. As long as the local eavesdropper cannot decrypt it, he cannot obtain any knowledge about the end server.

Collaborative Nodes: Collaborative nodes are attackers that passively collect and exchange information to expose the message originator. Analytical results of Crowds have shown that if $N > \frac{p_f}{p_f - 1/2}(C+1)$, where N is the number of nodes in the system, then the path initiator has probable innocence against C collaborators. This result can be applied directly to AURA_{LBR} because the relay method is almost the same as that in Crowds. If the collaborators are normally distributed in the system, AURA_{LFE} has the same property, too.

The case of AURA_{DPC} is more complicated. Ideally if each node on the path does not realize that the path is a composite one, then we have the same result as that in $AURA_{LBR}$, that is, the path initiator has probable innocence against C collaborators when $N > \frac{p_f}{p_f - 1/2}(C+1)$.

A situation that may expose the sender in AURA_{DPC} is that the immediate predecessor and successor nodes are both collaborative nodes. The collaborators cannot decrypt the message, but they can detect content modifications of the message. If a collaborator has a successor node that is the same as the predecessor node of his immediate successive collaborator and they detected a content modification, then with a non-trivial probability the node between two collaborators is the real sender. Let C_i, C_{i+1}, be the two collaborative nodes that are closest to the sender S and S' be the repeated occurrence of sender S. S' only relay the message, the actual message replacement is done by S. From the sender's perspective, there are two possible cases that will make him exposed.

Case 1: $< \ldots, C_i, S, C_{i+1}, \ldots >$

When S is the only node between two collaborators, he will be exposed. For a path with length $i \geq 3$, the probability that the immediate predecessor and successor nodes of S are all collaborator is equal to

$$p_1(i) = (i - 3 + 1)\left(\frac{C}{N}\right)^2$$

Therefore, the expected occurrence for all possible path length i is

$$p_1 = (1 - p_f)\sum_{i=3}^{\infty} p_f^{i-2} p_1(i) = (1 - p_f)\frac{C^2}{N^2}\sum_{i=3}^{\infty}((i-2)(p_f)^{i-2}) \tag{1}$$

Case 2: $< \ldots, C_i, S, \ldots, S', C_{i+1}, \ldots >$ or $< \ldots, C_i, S', \ldots, S, C_{i+1}, \ldots >$

The second situation is that S becomes the immediate successor of C_i and another appearance S' is the immediate predecessor of C_{i+1}, or vice versa. The

intermediate nodes between S and S' are chosen from $N-C$ nodes since C_i and C_{i+1} are adjacent collaborators. Let the path length be $i \geq 4$, the probability of this case can be expressed as

$$p_2(i) = \left(\frac{C}{N}\right)^2 \frac{2}{N} \sum_{j=0}^{i-4} \left(\left(\frac{N-C}{N}\right)^j (i-(4+j)+1) \right)$$
$$= \frac{2C}{N^3} \sum_{j=0}^{i-4} \left((i-j-3) \left(1-\frac{C}{N}\right)^j \right)$$

And the expected occurrence is

$$p_2 = (1-p_f) \sum_{i=4}^{\infty} p_f^{i-2} p_2(i)$$
$$= (1-p_f) \frac{2C}{N^3} \sum_{i=4}^{\infty} \sum_{j=0}^{i-4} \left(p_f^{i-2} (i-j-3) \left(1-\frac{C}{N}\right)^j \right) \quad (2)$$

The total probability that S will be exposed is $p_1 + p_2$.

From the collaborator's perspective, even if they detect a modification and a common node between two adjacent collaborators, the common node may still have chance not being the sender. Such a case is shown as follows:

Case 3: $< \ldots, C_i, O, \ldots, S, \ldots, O, C_{i+1}, \ldots >$

O is one of the $N-C-1$ relay nodes excluding the sender and collaborators. However, in this case O appears to be the sender, which misleads the collaborators. Again, on a path with length $i \geq 5$, the occurrence probability will be

$$p_3(i) = \frac{C^2}{N^2} \frac{N-C-1}{N} \sum_{j=0}^{i-5} \left(\left(1-\frac{C}{N}\right)^j (j+1)(i-(j+5)+1) \right)$$
$$= \frac{C^2(N-C-1)}{N^3} \sum_{j=0}^{i-5} \left((i-j-4)(j+1) \left(1-\frac{C}{N}\right)^j \right)$$

And the expected occurrence is

$$p_3 = (1-p_f) \sum_{i=5}^{\infty} p_f^{i-2} p_3(i)$$
$$= (1-p_f) \frac{C^2(N-C-1)}{N^3} \sum_{i=5}^{\infty} \sum_{j=0}^{i-5} \left(p_f^{i-2} (i-j-4)(j+1) \left(1-\frac{C}{N}\right)^j \right) \quad (3)$$

As a result, when two adjacent collaborators find a common node X between them and detects a message modification, X is the sender with probability $p_S = \frac{p_1+p_2}{p_1+p_2+p_3}$.

Figure 12 shows the impact on p_S with different values of p_f and various percentage of collaborative nodes in a system. Here we fix the system size N to 10000 nodes. When the number of collaborative nodes grows, the probability p_S that collaborators expose S becomes higher. On the other hand, when the probability of forwarding is increased, p_S decreases. Take $C = 3000$ as an example, once p_f is set larger than 0.65, S has probable innocence ($p_S \leq 0.5$) even if two collaborators find a common node and a message modification between them. Therefore, with properly selected p_f, AURA$_{\text{DPC}}$ achieves probable innocence degree of anonymity against collaborative nodes.

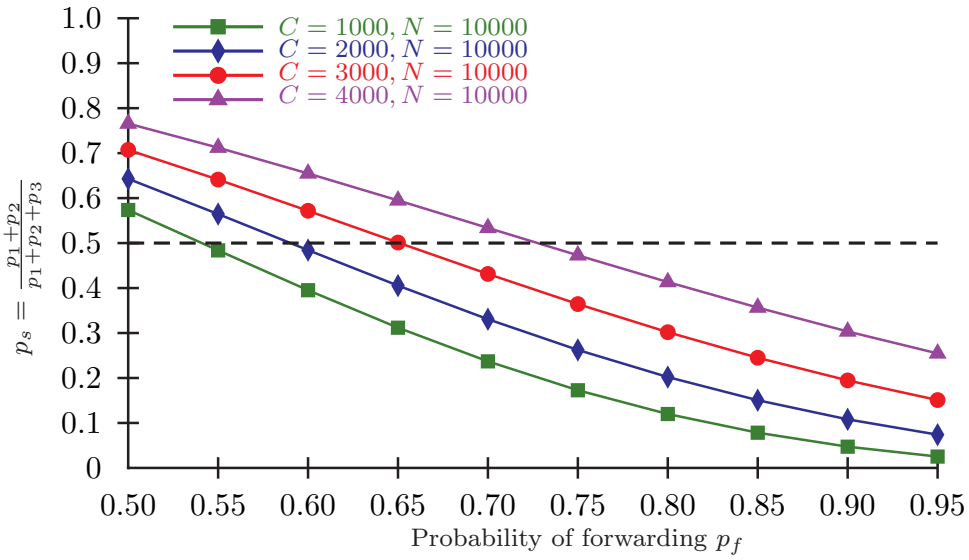


FIGURE 12. The relationship between p_S and p_f

It may seem that the lookup operations of the underlying structured peer network also expose sender's identity to other nodes in the system, but in our design the lookup message has no difference to other applications'. Only when the next relay node is found will the path construction message be sent. Even if the first node that receives the lookup message is a collaborator, he has no evidence to regard the message as a path construction one. Thus, the analysis results shown above are not affected.

5.2. **Sender anonymity set.** Sender anonymity set is another way to evaluate anonymity of a scheme. The anonymity set of AURA_{DPC} depends on the type of node-discovering method; therefore, in the following we only discuss AURA_{LBR} and AURA_{LFE}. For a node N_i along a path, here the sender anonymity set is defined as the set of all possible predecessor nodes who may originate a message. Since AURA_{LBR} differs Crowds only from the node-discovering method, the size of anonymity set against collaborative nodes in AURA_{LBR} is also the same as Crowds', that is, $N - C$. In other words, all nodes excluding the C collaborative ones in Crowds and AURA_{LBR} may be the real sender from collaborator's viewpoint.

Instead of choosing relay nodes from the whole system, AURA_{LFE} chooses them from the entry table. Different from previous cases, the underlying structured peer network has great impact on the size of anonymity set. Generally the size of the entry table is much smaller ($\log N$ in many schemes) than the system size, so the anonymity set will also be decreased. Again we use Chord as the underlying network. An intuitive result of the $\text{AURA}_{\text{LFE}} - \text{Chord}$ will be $\log_2 N$ since size of finger table in Chord is equal to $\log_2 N$. However, the actual size of anonymity set may be larger. Let S be the ID space of Chord, then each member node in the system should be responsible for $\frac{S}{N}$ IDs and each chosen ID may have $\log_2 N$ possible predecessors. The anonymity set of each node then becomes $\frac{S}{N} \log_2 N$, respectively.

6. **Conclusions and Future Work.** In this paper we proposed AURA, a scalable anonymous relay architecture that can be deployed on top of any structured peer-to-peer networks without modifications. According to the anonymous requirement, an application can be run at different level of anonymity. AURA inherits the excellences of Crowds by

the Crowds-like behavior but at the same time abandons the deployment of a centralized directory server. AURA also provides a path reconstruction mechanism to cope with the failure and departure problems of relay nodes. To fight against the local eavesdropping attack that is undefendable to many existing schemes, a dummy path composition mechanism is designed. Analytical and simulation results of AURA show that the performance on message routing and path construction are competitive to other existing schemes. Through the anonymity analysis, we also show that AURA is powerful when defending attacks from the end servers, collaborative nodes, and even a local eavesdropper.

There are still many works to be done. The implementation of AURA is under construction, and in our plan currently it will be built on top of Chord to test and simulate the performance and resistance to the attackers under various conditions. In this paper, the probability of forwarding p_f is fixed among the whole system. Actually, an adaptive p_f will make the system more flexible. For example, Figure 12 shows that AURA_{DPC} require a higher p_f when the collaborative nodes in the system increased. On the other hand, if a user only wants to hide his identity from the remote server, a lowest value 0.5 is sufficient.

The AURA architecture proposed in this paper can be treated as a prototype. On deployment it can be further customized with specific structured peer-to-peer network to obtain higher performance, scalability, and degree of anonymity. For example, the hybrid method mentioned in Section 4.1 provides a flexible trade-off solution between the cost of node-discovering and the size of anonymity set of $\text{AURA} - \text{Chord}$. With proper design, AURA can be a lightweight solution of anonymous communication over any existing structured peer network.

REFERENCES

[1] A. Pfitzmann and M. Köhntopp, Anonymity, unobservability, and pseudonymity – A proposal for terminology, *Proc. of the International Workshop on Design Issues in Anonymity and Unobservability*, vol.2009, pp.1-9, 2000.

[2] M. K. Reiter and A. D. Rubin, Crowds: Anonymity for web transactions, *ACM Transactions on Information and System Security (TISSEC)*, vol.1, no.1, pp.66-92, 1998.

[3] D. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, *Communications of the ACM (CACM)*, vol.24, pp.84-88, 1981.

[4] U. Moeller, L. Cottrell, P. Palfrader and L. Sassaman, *Mixmaster Protocol – Version 2, Draft*, http://mixmaster.sourceforge.net/, 2003.

[5] G. Danezis, R. Dingledine and N. Mathewson, Mixminion: Design of a type iii anonymous remailer protocol, *Proc. of the IEEE Symposium on Security and Privacy (S&P'03)*, pp.2-15, 2003.

[6] M. G. Reed, P. F. Syverson and D. M. Goldschlag, Anonymous connections and onion routing, *IEEE Journal on Selected Areas in Communications*, vol.16, no.4, pp.482-494, 1998.

[7] R. Dingledine, N. Mathewson and P. Syverson, Tor: The second-generation onion router, *Proc. of the 13th USENIX Security Symposium*, vol.13, pp.303-320, 2004.

[8] P. Druschel, M. F. Kaashoek and A. I. T. Rowstron, Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems, *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, vol.2429, pp.85-93, 2002.

[9] M. Gunes, U. Sorges and I. Bouazzi, Ara – The ant-colony based routing algorithm for manets, *Proc. of the International Conference on Parallel Processing Workshops (ICPP'02)*, pp.79-85, 2002.

[10] I. Clarke, O. Sandberg, B. Wiley and T. W. Hong, Freenet: A distributed anonymous information storage and retrieval system, *Proc. of the International Workshop on Design Issues in Anonymity and Unobservability*, vol.2009, pp.44-66, 2000.

[11] S. El-Ansary and S. Haridi, An overview of structured overlay networks, *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks*, 2005.

[12] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishna, Chord: A scalable peer-to-peer lookup service for internet applications, *Proc. of the ACM SIGCOM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp.149-160, 2001.

[13] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms*, vol.2218, pp.329-350, 2001.

[14] S. Hazel and B. Wiley, Achord: A variant of the chord lookup service for use in censorship resistant peer-to-peer publishing systems, *Proc. of the 1st International Peer to Peer Systems Workshop (IPTPS'02)*, 2002.

[15] G. Ciaccio, Recipient anonymity in a structured overlay, *Proc. of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW'06)*, pp.102-103, 2006.

[16] A. Singh and L. Liu, Agyaat: Providing mutually anonymous services over structured p2p networks, *Tech. Rep. No. GIT-CERCS-04-12*, Georgia Inst. of Tech. CERCS, 2004.

[17] H. Tsai and A. Harwood, A scalable anonymous server overlay network, *Proc. of the IEEE 20th International Conference on Advanced Information Networking and Applications (AINA'06)*, pp.973-978, 2006.

[18] L.-Y. Lee, J.-J. Lee and C.-L. Lei, A universal anonymous system for structured peer-to-peer networks, *TENCON 2007 – 2007 IEEE Region 10 Conference*, pp.1-4, 2007.

[19] R. Dingledine, M. J. Freedman and D. Molnar, The free haven project: Distributed anonymous storage service, *Proc. of the Workshop on Design Issues in Anonymity and Unobservability*, vol.2009, pp.67-95, 2000.

[20] T. Chothia and K. Chatzikokolakis, A survey of anonymous peer-to-peer file-sharing, *Proc. of the IFIP International Symposium on Network-Centric Ubiquitous Systems (NCUS'05)*, vol.3823, pp.744-7555, 2005.

[21] C. Díaz, S. Seys, J. Claessens and B. Preneel, Towards measuring anonymity, *Proc. of the 2nd International Workshop on Privacy Enhancing Technologies (PET'02)*, pp.54-68, 2002.

[22] M. J. Freedman and R. Morris, Tarzan: A peer-to-peer anonymizing network layer, *Proc. of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, pp.193-206, 2002.

[23] W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory IT-22*, vol.22, no.6, pp.644-654, 1976.