J. King Saud Univ., Vol. 19, *Comp. & Info. Sci.*, pp. 39-59, Riyadh (1427H./2007)

An Efficient Overlay Infrastructure for Privacy-preserving Communication on the Internet

Jalal Al-Muhtadi
College of Computer and Information Sciences,
King Saud University,
Riyadh, Kingdom of Saudi Arabia
jalal@ccis.edu.sa

(Received 16 September 2006; accepted for publication 13 December 2006)

Abstract. This paper presents Mist2 (Mist 2nd generation), an overlay infrastructure for privacy-preserving communication. Nodes are arranged in concentric rings, which reduce average overlay hop latencies. Mist2 builds on the strengths of previous work on Mist, and significantly improves on the communication overhead, resilience to router failures, and the distributed nature of the algorithm. Mist2 provides sender, receiver, and sender-receiver anonymity for communicating end users. Compared to previous approaches, Mist2 significantly improves the communication efficiency in latency without sacrificing the level of anonymity. This paper proposes an enhancement to the original Mist, and improves this solution using rings resulting in Mist2. The simulation results clearly demonstrate the improved performance of anonymous communication in Mist2 compared to relevant systems, while maintaining anonymity. Since the proposed ring-based approach optimizes path latencies, it performs better than the other proposed solutions where the overlay hops can potentially traverse large distances.

Keywords: Privacy, Anonymous communication, Routing protocol, Overlay network, Mist.

1. Introduction

The importance of preserving the privacy of an individual on the Internet can only be understated. Cyber-stalkers are able to collect vast amounts of information about unsuspecting users through the use of trojans such as network sniffers and spyware. Parents worry about their children's privacy in Internet chat-rooms and online forums. Through the use of cookies, web servers are able to infringe on a user's privacy by collecting information on the user's browsing patterns and identity. Many researchers have focused their attention on the problem of user anonymity. The attributes of

user anonymity include identity privacy which implies protecting the identity information about a user such as name, passwords, email addresses, telephone numbers, credit card numbers, private keys, geographic location, etc., from inadvertent exposure.

In order to preserve the anonymity of a user, researchers have focused on different aspects of the problem. Specifically, they focus their attention on *sender anonymity*, *receiver anonymity*, and *sender-receiver anonymity* [1]. A system provides sender anonymity if and only if it is not possible for the recipient of a message to identify the original sender. In receiver anonymity, it is not possible to ascertain who the receiver of a particular message is, even though the receiver may be able to identify the sender. Sender-receiver anonymity is the combination of both properties. Some of the proposed solutions place implicit trust on the communication infrastructure [2, 3], whereas in other solutions the infrastructure is oblivious to identity information of the endpoints of a communication [1, 4-8].

This section briefly reviews related research and situates this work in this context. One of the earliest efforts to provide anonymity was [8], where the focus was on hiding traffic patterns to foil traffic analysis using a technique called mixing. Routers communicate with each other using fixed length packets that are sent out at uniform intervals. Protocol messages are mixed in randomly with dummy packets to keep a constant flow of traffic in and out of the routers. In addition, all messages are encrypted with multiple router-specific keys by the sender of the message, who also picks the path the message follows through the network. The causality relationships between messages are not preserved. Onion routing implements this idea by forming an overlay network of Onion routers that do the mixing [6]. Encrypted layers are peeled off (like an onion) at each hop, hiding all routing information except the previous and next hop. Tor [9] is a second-generation Onion Routing system that addresses limitations in the original design by adding congestion control, directory servers, integrity checking, and other enhancements. These schemes provide anonymity to both the sender and receiver of the communication.

Crowds [4], on the other hand, is a solution that specifically targets anonymous web browsing. Crowd routers are dispersed across the Internet and communicate with each other using overlays forming a fully connected graph. Web requests from a sender are routed to a Crowds router and each Crowds router can decide to probabilistically forward them to another Crowds router or send it directly to the web server. If the request is forwarded, some state is stored on the router so that a reverse path is set up for the response from the server to the sender using a mechanism similar to virtual circuits. Using this protocol, the server only sees the IP address of the last Crowd router. The anonymity of the sender is preserved in this case because the originator of the message could equally likely be any one from the whole set of Crowds' users. While Crowds provides sender-anonymity, the overall path latencies are high, since the message can be routed across the Internet many times.

In Hordes [7], the authors augment Crowds by multicasting responses from a web server to all Crowds senders to provide a more efficient solution that results in shorter transmission latencies and requires less work from protocol participants in terms of messages processed. Crowds routers in Hordes need not store state information for reverse path forwarding of responses. In both Crowds and Hordes, the infrastructure is not aware of the sender of the Internet requests.

In P5 [1], the authors generalize the multicast idea to provide sender, receiver, and sender-receiver anonymity. Their overlay network solution organizes its subscribers in a hierarchy of broadcast groups. All communication is sent to broadcast addresses, providing receiver anonymity. Sender anonymity is provided by using mixes. However, instead of having one broadcast group, users use hierarchical group addresses. This provides sender-receiver anonymity, i.e. senders and receivers cannot determine the location of each other. A user reveals a "bit-mask" that specifies their position in a tree-like hierarchy. A user receives messages that are broadcast to all groups at their level as well as lower down in the hierarchy. Messages broadcast to groups lower down in the hierarchy have fewer subscribers and lesser privacy, whereas specifying a higher address adds to communication costs. This allows users to customize trade-off between anonymity and communications efficiency. The P5 infrastructure is oblivious to identity and location information of either or both endpoints of a communication.

Mist (first generation) [5] proposes a hierarchical solution for privacy-preserving communication. In some sense, the architecture creates a "mist" through which users could communicate while obscuring their location and identity information. Users at the leaves of the tree-like hierarchy register with *Lighthouses* (nodes higher up in the tree) through a series of *Mist Routers* along the path. Lighthouses serve as proxies for communication with a particular user. The higher a Lighthouse is in the hierarchy, the greater is the privacy of the user. Lighthouses are tied to geographical locations and choosing different Lighthouses at different levels allows users to trade privacy for communication efficiency.

This paper addresses the shortcomings in the original Mist architecture by proposing a self-organizing overlay network of infrastructure nodes called *Rings*. In the proposed solution, participating nodes form an overlay network of multiple concentric rings. The innermost ring consists of Lighthouses that are maximally connected to each other using overlays. Mist Routers arrange themselves around the Lighthouses into rings that reflect their latency distance from the Lighthouses. All Mist Routers within a ring are within a specific latency range from their Lighthouses. When a user decides to connect to a Lighthouse, the user sets up a route of desired hop-count[(1)] to the Lighthouse traversing several Mist Routers through adjacent rings. This option allows

[(1)] Unless otherwise specified, the hop-count refers to *hops* along the overlay, and not the underlying network layer hops.

the user to customize the trade-off between communication efficiency and privacy. This system is referred to as *Mist2* (Mist 2nd generation) to distinguish it from the previous work on Mist. Mist2 provides more resilience to route failures in the overlay since routers are not restricted to a fixed hierarchy. The average per-hop latency is low since packets are routed between adjacent rings. Rings also provide scalability since Mist2 Routers only need to keep track of neighbors in adjacent rings, as opposed to a Crowds-like approach in which members need to know all other members in the crowd.

Rings expose the underlying latency and topological information and allow users of the system to increase the efficiency of anonymous communication without sacrificing privacy. The proposed approach differs from existing protocols in two important ways. One of the major problems with existing privacy-preserving communication protocols is the excessive overhead in terms of latency and bandwidth [1, 4, 6]. First, by exposing the topology to the protocols and users in Mist2, it is possible to reduce communication overheads significantly (by more than 50% in comparison to a Crowds-like implementation in the simulations). Second, the proposed system focuses on user-to-user communication in the design, which is often overlooked in other protocols that take advantage of group-oriented communication for anonymity. The overhead of implementing user-to-user communication in Crowds and P5 can be extremely prohibitive. Like P5, the proposed system's infrastructure nodes are also oblivious to the identity and location information of communication endpoints.

The rest of the paper is organized as follows. Section 2 presents the Mist2 architecture in detail and shows how Rings are formed. Section 3 presents the user-to-user privacy-preserving communication protocol that is layered on Mist2. Section 4 presents simulation results using CAIDA data and compares the overhead with a Crowds-like implementation. Section 5 concludes the paper.

2. Bootstrapping and Infrastructure

Overlay networks or peer-to-peer networks are becoming increasingly popular for application layer routing in the Internet. Pure peer-to-peer or unstructured peer-to-peer networks [10, 11] tend to show large variances in communication latency, available bandwidth, and throughput. To restrict such unpredictable behavior, structured peer-to-peer routing solutions have been proposed, e.g. Chord [12] and CAN [13].

In the context of anonymous routing, Crowds, Hordes and Tor use unstructured peer-to-peer networks. In Crowds, packets are probabilistically routed between members, and eventually to a web server. Mist [5] proposes a structured anonymous routing architecture, assuming a rigid tree hierarchy that could be too restrictive. To remedy this, a two-way Crowds solution with Lighthouses is considered. Hence, a user

can set up a connection to a Lighthouse just as a Crowds member could set up a reverse path to a web server. Lighthouses would then serve as a communication beacon for that particular user. Tarzan uses a similar approach, where they use a more structured route to NAT boxes, which are similar to our Lighthouses. Hop latencies in Tarzan can be arbitrary as peers are chosen based on the structure of IP addresses, which may or may not correspond to "close" peers.

Mist2 proposes a more flexible architecture by employing the rings as its basic structure. Mist2 networks consist of multiple rings. At bootstrap time, a large number of Lighthouses that belong to different administrative domains and lie within a radius of *R* ms latency form a special ring. This ring is referred to as the *Mist2 Center* (Ring 1). Additional rings are defined based on their members' latencies from the innermost ring (see Fig. 1). Assuming the use of *R* as the radius of a ring, the radius of Ring 1 is *R*, the radius of Ring 2 is 2*R*, and so on. Each Mist2 Router is associated with a particular ring. Mist2 Routers maintain a list of "close" Mist2 Routers in the neighboring rings. While routing packets between rings, Mist2 Routers pick a random router from their neighbor sets. Users decide on a specific route to the Mist2 Center based on a desired hop-count. For example, a user in Ring 4 may choose the following route to Ring 1: 4-3-2-3-4-5-4-3-2-1. Since each hop is constrained in latency as an artifact of the rings, the overall hop latency is expected to be low. Compared to a Crowds-like approach where each hop is half the diameter of the Internet on the average, Mist2 provides significantly more efficient communication.

2.1. Ring formation

Every entity in Mist2 including the Lighthouses and the Mist2 Routers has a pair of public/private keys. Also, every entity is issued an identity certificate containing its identity and its public key. Mist2 depends on PKI or other equivalent key management support. Based on the public key certificates, any entity in Mist2 can be verified by any other entity with a simple cryptographic signature verification.

Our infrastructure allows Mist2 Routers to join the overlay network at any point in time. To join, a Mist2 Router must first prove that it is located in the latency space of a specific ring in the Mist2. Mist2 employs an "anchor" Lighthouse for this purpose. The Mist2 Router that wishes to join the network contacts the anchor Lighthouse with a join request. The anchor Lighthouse then pings the Mist2 Router to find the latency to it and issues a latency certificate containing the identity of the Mist2 Router, and the ring number where the Mist2 Router belongs to, based on the measured latency. All latency certificates issued by the anchor Lighthouse are signed with the private key of the anchor Lighthouse so that the validity of the certificates can be easily checked by other Mist2 Routers. Latency from the anchor Lighthouse to a Mist2 Router may change over the course of the time due to traffic fluctuation or network topology changes. Therefore, all Mist2 Routers are required to renew their latency certificates periodically.

At the same time, the anchor Lighthouse also provides the Mist2 Router with the list of other Mist2 Routers in the neighboring rings. Using this information, the newly joined Mist2 Router can ping other neighboring Mist2 Routers and choose a subset of close neighbors.

Figure 1 shows an example of a Mist2 Router *N* joining the network. First, *N* acquires its identity certificate externally. Using that certificate, *N* contacts the anchor Lighthouse in Ring 1 to get a ring certificate. The anchor Lighthouse pings *N* and determines that *N* belongs to Ring 3. The anchor Lighthouse returns a ring certificate to *N* with the list of Mist2 Routers in Rings 2 and 4. Using this information, *N* pings these Mist2 Routers in Rings 2 and 4 and decides to store nodes {A, E, F} from Ring 2, and nodes {B, D, G, H} from Ring 4 in its neighbor list.

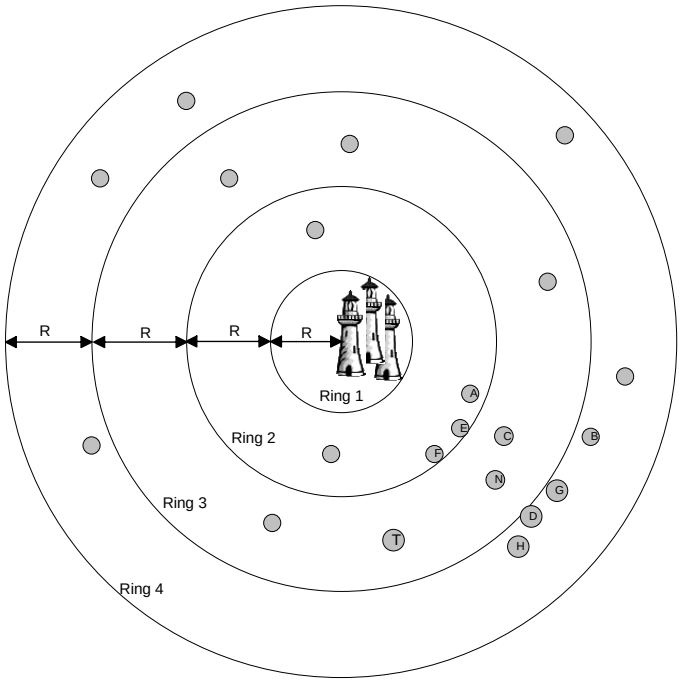

Fig. 1. Ring formation in Mist2.

3. Communication in the Mist2

This section describes how two parties, Alice and Bob, can communicate through the Mist2. Alice and Bob independently register with Lighthouses. The term Lighthouse $_{Bob}$ is used to refer to the Lighthouse that Bob is registered with. When Alice and Bob want to communicate, each has to establish a *Mist2 Circuit* to her or his Lighthouse. The Mist2 Circuit is a special communication channel that provides the required privacy guarantees. When Alice communicates with Bob, they form an end-to-end *Mist2 Connection* that allows communication to take place over the path Alice – Lighthouse $_{Alice}$ – Lighthouse $_{Bob}$ – Bob.

3.1. Lighthouse registration and Mist2 circuit establishment

Registration with a Lighthouse is the first step to communicate in Mist2. The path established from Alice to her Lighthouse must provide her with anonymity. It should be infeasible for Lighthouse $_{Alice}$ and Mist2 Routers along the path to deduce Alice's location or ring number.

Mist2 uses a route establishment technique that resembles the tunnel setup protocol in Tarzan [14]. This routing technique serves two purposes. First, it allows the user to register with a Lighthouse. Second, it allows the establishment of a Mist2 Circuit between the user and its Lighthouse. Alice, for example, can register with a Lighthouse using the following five steps.

Step 1: Alice picks the desired number of hops, *H*, based on her privacy needs (Step 2 will illustrate that the value of *H* is constrained to either even or odd numbers based on Alice's ring). Longer paths will make it harder for Lighthouse $_{Alice}$ to collude with intermediate routers to determine Alice's location. However, longer paths also increase the latency, trading off lower latencies for a higher degree of privacy.

Step 2: Alice computes a route from her ring, Ring *N*, to the inner-most ring, Ring 1, based on the number of hops selected in Step 1. Each hop must be to a neighboring ring. For example, a 7-hop route from Ring 4 to Ring 1 could look like <4,3,4,5,4,3,2,1>. Note that $H = N + 2n-1$, for some non-negative *n*. In this example, N=4, so the only possible values for *H* are {3, 5, 7, 9, …}.

Step 3: Based on the route computed in Step 2, Alice sets up a path to a Lighthouse in Ring 1. This step is similar to the tunnel setup in Tarzan. Alice iteratively advances the Mist2 Circuit to the Lighthouse by contacting successive Mist2 Routers.

In general, all packets in Mist2 have the format shown in Fig. 2. The 'Handle ID' field represents a handle that is unique per Mist2 Router that helps identify the next hop of a particular Mist2 Circuit. The handles are assigned hop-by-hop in order to obfuscate the complete Mist2 Circuit. A value of 0 in this field indicates that no value is assigned yet. How the handle is used is described later in this section. The 'packet type' identifies the type of the packet, which tells the intermediate Mist2 Routers how they should handle the packet. More specifically, Alice's Mist2 Circuit establishment packet will contain '0' for the handle ID, indicating that no handle ID is assigned yet. The type field will contain a value indicating that this is a Mist2 Circuit establishment packet. The payload is the content of the message. *H*(payload) is a cryptographic hash of the payload and is used to verify message integrity.

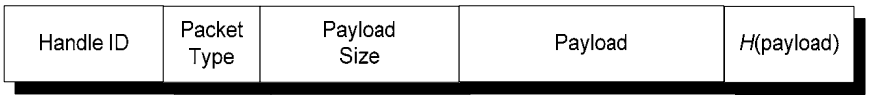

Fig. 2. General format for Mist2 packets.

For example, if the path is <4,3,2,1>, Alice first contacts a Mist2 Router from her inner neighbor list with a circuit establishment message of the first type $E_{key\_next\_hop}$ *(TS* || *ring_after_next)*, where *TS* is a timestamp (to aid in verifying the freshness of the message), '||' denotes concatenation, and "*ring_after_next*" is the next ring in the path. This message is encrypted with the Mist2 Router's public key "*key_next_hop."* The Mist2 Router "*next_hop*," which is in Ring 3 in this case, returns a list of neighbors in Ring 2 (ring_after_next). Alice then picks one at random, and reiterates using the second message type: $E_{key\_last\_hop}$ *(next_hop* || $E_{key\_next\_hop}$ *(TS* || *ring_after_next))* where "*next_hop"* denotes the selected Mist2 Router in the next ring. "*key_last_hop"* is the public key of the last Mist2 Router in the established path and "*key_next_hop"* is the public key of the next Mist2 Router. This process iterates until a complete path from the host to a Lighthouse is established.

The final registration message with the Lighthouse is E_k (M) where:
M = (*Alice* || *TS* || $K_{session}$ || *TKN* || *PP*), where:
Alice is Alice's unique ID in the active information space.
TS is a timestamp to prevent replay attacks.
$K_{session}$ is a random session key to encrypt further communication between the user and her or his Lighthouse. It is also used to add some additional randomness into the encrypted message.
TKN: A token that is used to prove that the user's request to register with a Lighthouse is authentic. The details of this token will be mentioned later.

PP: A predetermined fixed phrase. This is used to ensure that the decryption was

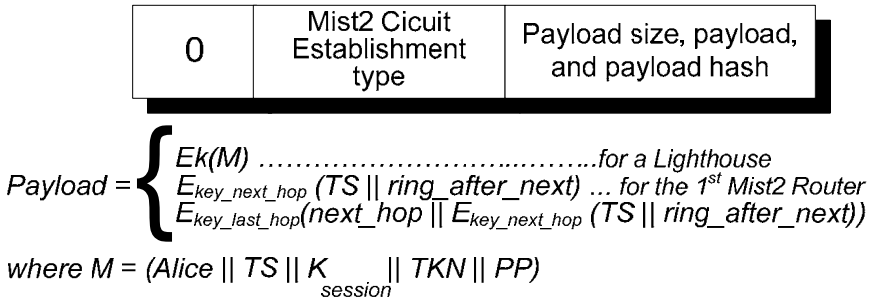

Fig. 3. Alice's Mist2 circuit establishment packet. Three types of messages are used here.

indeed successful. Fig. 3 illustrates the three types of Mist2 Circuit establishment messages. Figure 4 illustrates how handles are generated in Mist2.

Step 4: To complete the Mist2 Circuit establishment, the Lighthouse confirms the registration of Alice by sending back a reply packet. The format of this is:
$<handle\_id, Mist2\_COMM, E_{Ksession}(\text{“OK”} \| TS_2)>$

where $K_{session}$ is the secret key established between Alice and her Lighthouse in Step 3. TS_2 is a timestamp to prevent replay attacks. The packet type is marked as an "Mist2 Communication" packet to distinguish it from circuit establishment messages.

3.2. Authenticating the registration

In order to thwart malicious Lighthouses from falsely claiming that some user registered with them, the user constructs a special token (*TKN*) signed by the user's private key. This token will contain a timestamp and the unique ID of the chosen Lighthouse. This token is sent to the Lighthouse during the Mist2 Circuit setup as described in the previous section. Once the Mist2 Circuit has been established, the Lighthouse uses this token to show that the registration is legitimate. If the timestamp has already been seen before, or if it has expired, the token will be discarded. Naturally, if the signature cannot be verified, the token is also discarded. The format of this token, *TKN*, is as follows:

$$TKN = (User\ ID \,||\, Lighthouse\ ID \,||\, Timestamp \,||\, Expiration\ Time \,||\, S_{User}(User\ ID \,||\, Lighthouse\ ID \,||\, Timestamp \,||\, Expiration\ Time))$$

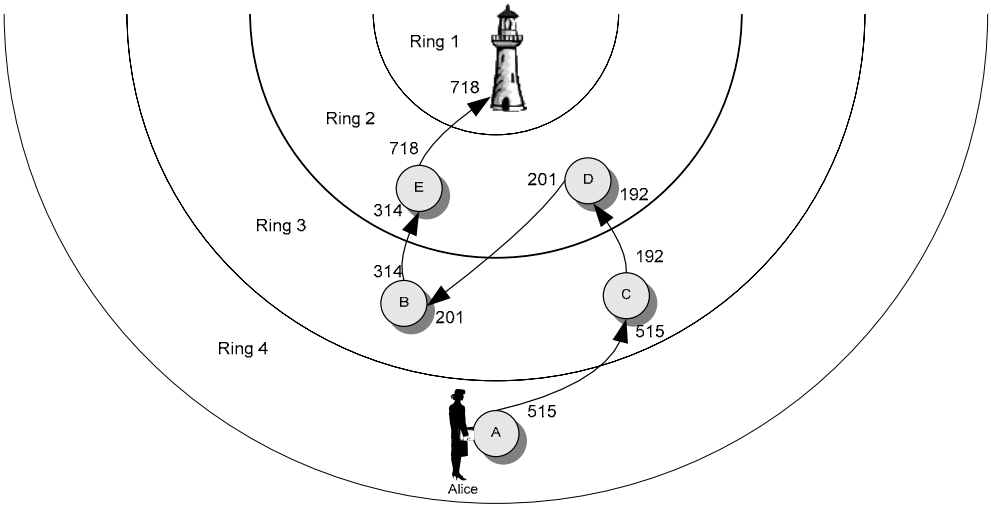

Fig. 4. Handle setup.

The *TKN* contains the unique user ID, the Lighthouse ID (this could be the DNS name), the timestamp (to prevent replay attacks), and an expiration time after which the TKN is not valid. The *TKN* is signed by the user's private key. *TKN* contents do not need to be encrypted because the contents are already known by the Lighthouse anyway. This token can now be used when a Lighthouse shares the registration information with other Lighthouses, to prove the authenticity of a registration.

3.3. Connection between two parties

Connecting between two parties in Mist2 is similar to the original Mist in the most part. Alice and Bob independently register with Lighthouse $_{Alice}$ and Lighthouse $_{Bob}$ respectively. All communication between Alice and Bob must go through their Lighthouses. This communication channel between Alice and Bob is called a Mist2 Connection. Assuming that Bob is trying to initiate communication with Alice, Bob then generates the following message for his Lighthouse:

$$M_{Lighthouse} = E_{K\ session}(COMM\_SETUP \,||\, Alice's\ ID \,||\, TS)$$

Unlike the original Mist, communicating parties do not need to access lookup servers to figure out the Lighthouse of the other party. This will be carried out by Lighthouses. Since handles have been set up in both directions during the Mist2 Circuit Setup phase, this

message will traverse the Mist2 Circuit to Lighthouse $_{Bob}$. This message is encrypted to preserve the integrity and confidentiality of Bob's specific communication request. When the message arrives at Lighthouse $_{Bob}$, it is able to uniquely determine that the message is from Bob based on the arriving handle. It decrypts the message with session key $K_{Session}$ and determines from the *COMM_SETUP* message type that communication must be set up with Alice. It is assumed here that the Lighthouses have a mechanism to communicate information with other Lighthouses. For example, the Lighthouses could form a multicast group to exchange information about registered users. Optionally, lookup services could be used which are accessed and updated by the Lighthouses. The timestamp *TS* is used to prevent replay attacks.

Lighthouse $_{Bob}$ uses asymmetric key encryption to communicate with Lighthouse $_{Alice}$ to determine Lighthouse $_{Alice}$'s handle for Alice. Since this is straightforward, the details of this communication are omitted from this paper. The aforementioned handle can be referred to as dest_handle $_{Alice}$. In Fig. 5, Lighthouse $_{Bob}$ determines dest_handle $_{Alice}$ = 254. Lighthouse $_{Bob}$ then generates a unique handle that Bob can use to address Alice. This handle is referred to as src_handle $_{Alice}$. In Fig. 5 src_handle$_{Alice}$ = 689. Lighthouse$_{Bob}$ sets up a binding of the form <src_handle$_{Alice}$, dest_handle$_{Alice}$, Lighthouse$_{Alice}$>. Fig. 5 shows the binding <689, 254, Y>. All messages from Bob that arrive for src_handle $_{Alice}$ (689) will be tunneled to Lighthouse $_{Alice}$ (Y) and indexed with dest_handle $_{Alice}$ (254). Similarly, Lighthouse $_{Bob}$ will supply the handle for Bob to Lighthouse $_{Alice}$ that will set up a binding of the form <src_handle $_{Bob}$, dest_handle $_{Bob}$, Lighthouse $_{Bob}$> in the same way. Fig. 5 shows this binding as <412, 100, X>. Once Lighthouse $_{Bob}$ and Lighthouse $_{Alice}$ have setup their bindings, they need to inform Bob and Alice of the source handles. Lighthouse $_{Bob}$ sends src_handle $_{Alice}$ to Bob in the following message: $M_{Handle} = E_{K\,session}$*(HANDLE_MSG || Alice's ID || src_handle* $_{Alice}$ *|| TS).* In Fig. 5, this message corresponds to "For Alice use 689." Similarly, Lighthouse $_{Alice}$ sends src_handle $_{Bob}$ to Alice.

Now Bob can send Lighthouse $_{Bob}$ messages destined to Alice by simply using src_handle $_{Alice}$ (689), and Alice can send Lighthouse $_{Alice}$ messages destined for Bob using src_handle $_{Bob}$ (412). This is done to hide Alice's identity from intermediate routers. These intermediate routers are hence unaware of *both* the endpoints of the communication. To communicate with Alice, Bob constructs messages of the following form, where '*M*' is the message for Alice: M_{For_Alice} *= (COMMUNICATION_MSG || src_handle* $_{Alice}$ *|| M)* and sends it towards his Lighthouse. Note that the message passes in the clear, and the use of handles does not disclose the endpoints of the communication. Alice and Bob are now free to choose an end-to-end encryption scheme if desired. Using this method, there is no duplication of encryption by the Mist2. Furthermore, Bob and Alice can have multiple connections open, where each connection is associated with a set of src_handles.

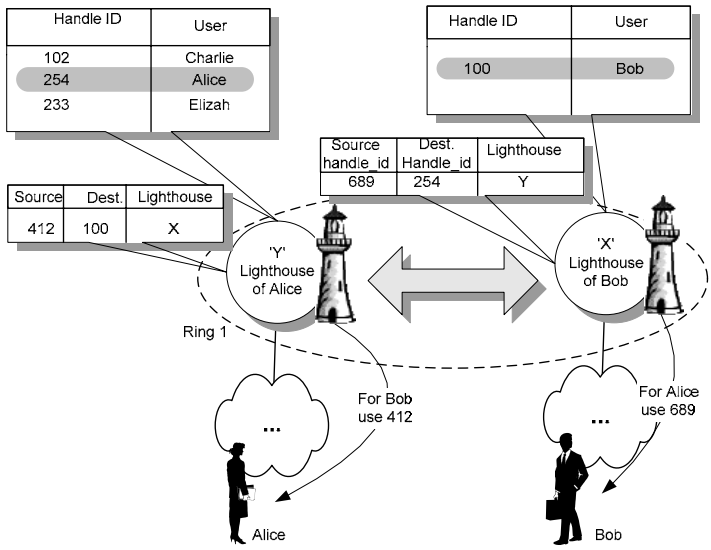

Fig. 5. Mist2 communication setup.

3.4. Hiding from Lighthouses

In the protocol presented so far, Alice and Bob's communication is not anonymous to the Lighthouses. This section describes an enhancement to the protocol to achieve this. This enhancement requires a lookup server in the system that is a member of the Lighthouse multicast group. Hence, the lookup server is aware of all the users registered with Lighthouses in the Mist2. Just like the original Mist [5], the message $M_{Lighthouse}$, which Bob sends to his Lighthouse, needs to include an encrypted token, T_{Alice}, with Alice's ID. This token is encrypted with the lookup server's public key. Hence, $Lighthouse_{Bob}$ is not aware of the identity of Alice.

$$M_{Lighthouse} = E_{Ksession}\ (COMM\_SETUP \mid\mid T_{Alice} \mid\mid TS)$$

When $Lighthouse_{Bob}$ sends T_{Alice} to the lookup server, the lookup server retrieves Alice's ID from T_{Alice} and determine $Lighthouse_{Alice}$. The lookup server generates another token $T_{Lighthouse\text{-}Alice}$, encrypted with $Lighthouse_{Alice}$'s key, which contains Alice's identity. $Lighthouse_{Bob}$ uses this token instead of Alice's ID with $Lighthouse_{Alice}$ to determine $dest\_handle_{Alice}$. The rest proceeds as before. Hence, Bob can route packets to Alice without

$Lighthouse_{Bob}$ knowing that the packets are for Alice. Similarly, $Lighthouse_{Bob}$ will provide src_handle_{Bob} to $Lighthouse_{Alice}$, but without disclosing Bob's identity. The anonymity of the connection relies on there being multiple users registered with a Lighthouse at any given time. This is because $Lighthouse_{Bob}$ is aware of all the users registered with $Lighthouse_{Alice}$ even though it does not know the endpoints of each connection. To relax this assumption, one solution would be to not use a multicast group, and instead rely on a lookup service for Lighthouses to determine other users' Lighthouses. The anonymity of the connection is compromised if the two Lighthouses involved in the communication collude. The anonymity is also lost if a Lighthouse colludes with the lookup service. However, the assumption here is that this is not trivial and that Lighthouses are operated by various trusted individuals or organizations.

4. Implementation

This section describes a proof-of-concept implementation of building a Mist2 communication infrastructure in the context of a Wide Area Network. Specifically, the implementation assists in studying two aspects of Mist2 communication behavior:

- How does Mist2 compare with a Crowds-like solution in terms of communication overhead?
- What is the effect of changing the number of rings, as well as changing the ratio of Mist2 Routers to regular routers in a given topology, with respect to the overheads?

In addition to developing a feel for the overheads, this work presents an analysis of privacy properties of the Mist2 framework, specifically in terms of sender anonymity, receiver anonymity, and sender-receiver anonymity.

Since it is not possible to obtain access to a large number of international hosts on the Internet that can act as Mist2 Routers and run the Mist2 communication protocol, the choice was made to simulate the behavior of the proposed system using data that reflects the connectivity map of the real Internet. Specifically, the data for the Inter-AS (Autonomous System) connectivity map is used, which is available through CAIDA as a part of the Skitter project[(2)] [15]. Rings are simulated on this topology map of a snapshot of the Internet[(3)]. To estimate the overheads in a simplified manner, latency is assumed to correlate with geographic distances, which is justified partly by studies such as [16, 17].

[(2)] The data used in this research was collected as part of CAIDA's skitter initiative (ITDK0204), http://www.caida.org. Support for skitter is provided by DARPA, NSF and CAIDA membership.
[(3)] The data available to us provides a snapshot of the whole Internet in April 2002.

The simulation is described in more detail in Section 4.1, followed by an analysis of privacy in Section 4.2.

4.1. Simulating Mist2 and Crowds using Skitter data

The simulation uses the data collected by the CAIDA Skitter project as the basis for studying the overheads of the proposed communication framework and comparing it with a simulation of a Crowds-like solution. Skitter data reflects 1,224,733 IP addresses and 2,093,194 IP links, (immediately adjacent addresses in a traceroute-like path). This data was collected from 21 skitter monitors probing approximately 932,000 destinations spread across over 75,000 (70%) of globally routable network prefixes.

The Skitter project also provides an aggregate view of the network as a graph of ASes and their interconnections (or peering sessions). Each AS in this graph approximately maps to an Internet Service Provider (ISP). Some ISPs administer more than one AS, but it is not typical according to their findings. Each IP address is mapped to the AS responsible for routing it. This is done by examining Border Gateway Protocol (BGP) routing tables collected by the University of Oregon's "Route Views" project [18]. Their aggregate graph consists of 11,122 Autonomous System (AS) nodes and 35,752 peering sessions. Since they could not provide a geographic location for 123 ASes, the data used contains 10,999 ASes and 34,209 peering sessions. A partial visualization of this graph is presented in Fig. 8 and 9, which show routes picked by Mist2 and a Crowds-like solution for the same host-Lighthouse pair. Using the graph of ASes and peering sessions, the Mist2 protocol is simulated as follows:

1. First, a "Ring 1" (which will contain the Lighthouses) needs to be selected. To do this, an AS, that is rich in connectivity, is selected (and referred to as the "anchor"). Other ASes that are within one ring radius of the anchor are detected. The anchor Lighthouse and its neighbors form potential candidates for Lighthouses in the simulation. The simulation designates (uniformly at random) a subset of these candidates as Lighthouses. Ideally the Lighthouses should be close to the latency center of the Internet. The rationale behind picking Lighthouses on "backbones" with high connectivity is to decrease the communication overheads between Lighthouses. Geographic distances could be a close approximation of this property. After analyzing the skitter AS graph, two potential latency centers in the topology are found, one in North America and the other in Europe. For this simulation, the North American center was chosen as the anchor.
2. Second, the number of rings to be simulated is chosen by specifying a geographic ring-radius. Once the number of rings is fixed, Dijkstra's algorithm is deployed to generate "All-Pair-Shortest-Paths" from the Lighthouses to decide to which ring a given AS belongs. Once this is done, the simulation selects the actual Mist2 Routers. An important point to remember here is that Mist2 Routers and

Lighthouses are implemented on hosts in these ASes and not on actual routers (i.e., application-level routing is used). The ratio of Mist2 to non-Mist2 Routers is then selected, after which the simulation selects the ASes to contain Mist2 Routers at random uniformly.

3. Each Mist2 Router that was picked in Step 2 now runs Dijkstra's shortest-path algorithm to find its neighbors. It makes a list of neighbors in adjacent rings, as described previously, and stores them in its routing table.
4. In order to simulate communication patterns between a Mist2 sender and receiver, the average hop length parameter in the simulation is fixed. The average path-length corresponds to the number of rings a packet visits on its path from the source or destination, to or from its Lighthouse. The simulation picks a sender and receiver at random and finds paths between: (1) the sender and its Lighthouse, and (2) the receiver and its Lighthouse according to the following algorithm:
   The sender (or alternately the receiver) picks a hop length from a geometric distribution whose average is the path-length. This is the same distribution as Crowds path lengths. Then, it generates a series of flags that correspond to whether it should route to a Mist2 Router at one-hop distance in a ring towards Ring 1, or away from it, so that it eventually ends at a Lighthouse in Ring 1 with exactly path-length number of hops. Next, the simulation searches the neighbor lists that were generated in Step 3 and creates a path with Mist2 Routers picked at random from each ring. If such a path can be found, the simulation computes the overhead by adding the geographical distances. Otherwise, the simulation tries again a finite number of times. Sometimes, it is possible not to find a path, as this depends on the number of Mist2 Routers in the system. In this case, the user may choose to use a smaller path length and try again. Note that this overhead of finding a path is only incurred once per session, at route set-up.
5. Once both sender and receiver can find a path to a Lighthouse in Ring 1, the simulation adds the distances between the Lighthouses to obtain the overhead of communicating with that path in terms of geographic (great-circle) distance. This is repeated for many different sender-receiver pairs in the simulation to obtain an average distance overhead.

Next, Crowds-like behavior is simulated on top of the Mist2 framework as follows:

1. The average path length is fixed to be exactly the same value as in Mist2, and the probability of whether to forward or not in the Crowds probabilistic forwarding algorithm is fixed to (1/path_length). This will generate paths with the expected number of hops as the path-length.
2. The simulation generates sender-receiver pairs at random, as in Mist2, and each sender and receiver uses Crowds-like probabilistic forwarding to generate a path to any Lighthouse. This is achieved by making the last hop in the Crowds simulation

path always end in a Lighthouse picked at random. Note that for each corresponding sender-receiver pair in the Mist2 and the Crowds simulation, the number of hops was fixed for that pair. This was done to provide a fair comparison between the two approaches, and was achieved by using the same pseudorandom generator for both simulations.

The results of the simulation are as follows. Fig. 6 shows how varying the ring radius affects average path distances. The ring radius is changed from 100 km to 9000 km to observe the change in the average path distances. For low ring radii (< 500 km), the number of rings are very high, and this in turn increases the number of hops required to reach Ring 1 (note that even if the desired number of hops may be 10, a Mist2 Router in Ring 21, will have to traverse at least 20 hops to reach Ring 1). In the simulation, this behavior is observed until a ring radius of 500 km. For ring radii beyond 500 km, increasing the ring radius results in longer inter-ring hops, which is why a general increase in average path lengths is seen. The drops that appear at some places in graph (for example, for ring radii 5000, 7000 and 9000 km) are because the number of rings reduces. When the number of rings reduces, the average hop length increases. However, there are fewer rings between a host and Ring 1, which results in lower distance routes on the average.

In Fig. 7, the number of Mist2 Routers varies from 1,000 to 10,000. The figure shows how the average path lengths are much lower than a Crowds-like solution. The figure also includes a curve for Mist2-fixed-hops, in which the number of host to Lighthouse hops is fixed to 10.

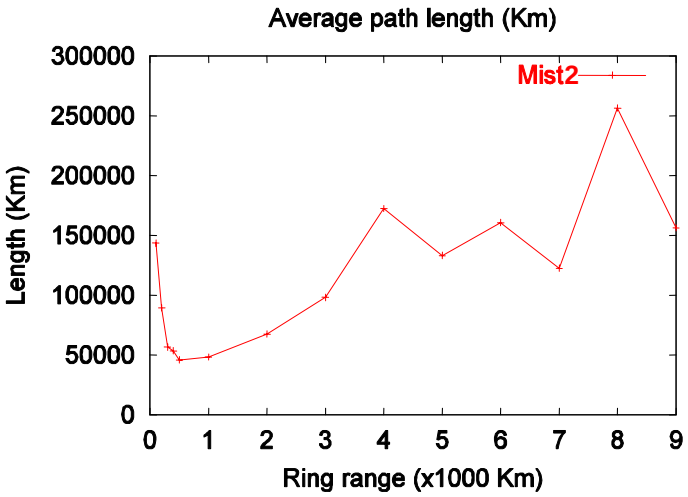

Fig. 6. Comparing average path lengths for various ring radii.

The other two curves correspond to a routing probability of 0.1, which provides 10 hops on the average. This is done to provide a fair comparison between Mist2 and a Crowds-like approach. However, while using Mist2, the user will be able to specify the number of hops to the lighthouse, which would then result in slightly better performance. This improvement is illustrated in the Mist2-fixed-hop curve.

Fig. 8 and 9 provide a visualization of the CAIDA AS Graph using the simulation code, and illustrate two sample paths for the same host-Lighthouse pair using Mist2 and a Crowds-like solution. Both paths are fixed at 10 hops. As one would expect, the hops for Crowds are large. Since Mist2 is structured using Rings, Mist2 follows a path that is much more efficient. In this case, the router traverses the following rings: <9 8 7 6 7 6 5 4 3 2 1>. In this instance, the path lengths for Crowds and Mist2 were 53,272 km and 15490 km respectively.

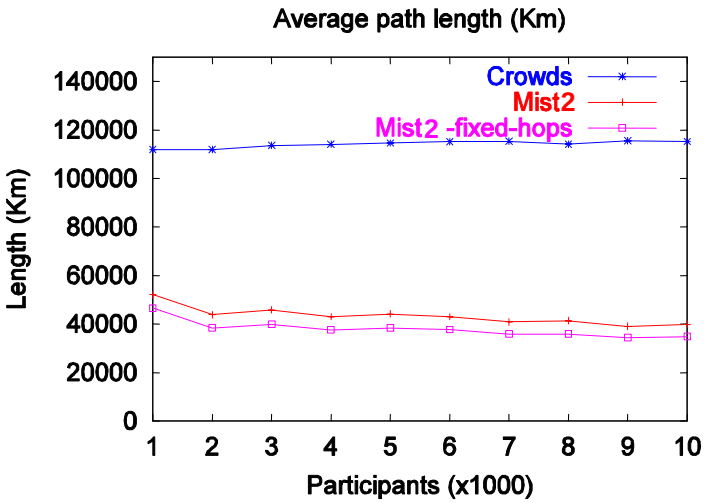

Fig. 7. Comparing average path lengths for Crowds and Mist2 for varying number of participants.

4.2. Privacy analysis of Mist2

Mist2 provides shorter routes because our routing is topology aware. It uses Rings to reduce hop latencies. However, this does not compromise a user's anonymity. A router cannot tell if your communication originates in its neighboring ring, or if it is merely relaying your packets on a much longer path. Since paths can traverse rings in both directions, an intermediate router cannot make any assumptions on which direction the packet originated. Traffic analysis is outside the scope of this paper. Several other approaches [1, 7] discuss how "mixing" and "cover traffic" can be added to foil traffic analysis. Mist2 can be augmented with mixing and cover traffic to counter traffic

analysis. However, many privacy scenarios on the Internet do not really require this functionality and, hence, this paper focuses mainly on providing endpoint user anonymity. Tarzan [14] adds another layer of security to foil traffic analysis by creating Onion data packets. This prevents attacks where malicious routers can reroute data packets or compare packets and short-circuit routes. Tarzan approach is extremely heavyweight, since each data packet would have to be encrypted with several rounds. In Mist2, a malicious router can do two things: either drop the packet or short circuit a route between itself and another malicious router that it knows along the path. The former case is not important since a user would reestablish a route if packet acknowledgments were not received. In the latter case, the number of hops may be reduced. While packets are not lost, this may be seen as a violation of a user's requested level of privacy (i.e., desired hops to the Lighthouse). However, a user picks this desired level of privacy to specifically "tolerate" such routing misbehaviors. In this manner, Mist2 provides a good tradeoff, where a user's level of privacy may be reduced, while relieving the user from the overhead of creating Onion data packets. Of course, Mist2 can be easily augmented to use Onion routing packet, if a user needs the added protection.

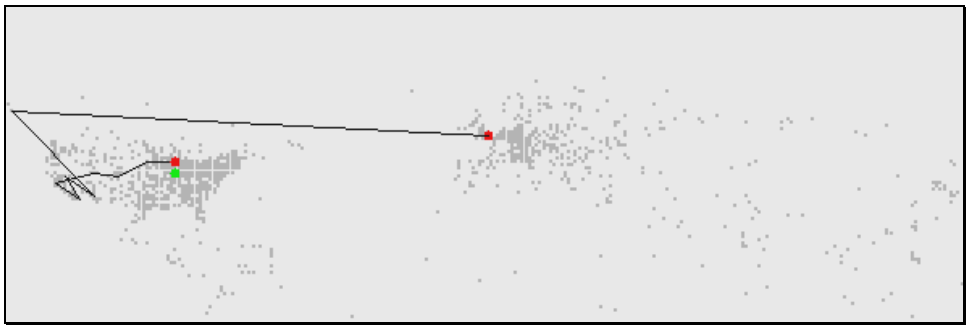

Fig. 8. CAIDA AS graph with a sample path generated by Mist2.
Small dots represent ASes. Large dots represent the communication endpoints.

4.3. Resilience to failure

Both Mist (first generation) and Mist2 employ a structured peer-to-peer approach to reduce communication latency while providing anonymity. Evaluations show that the

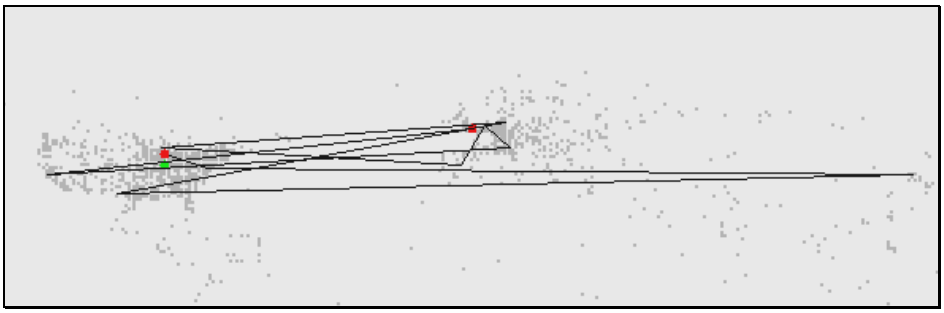

Fig. 9. CAIDA AS graph with a sample path generated by a Crowds-like solution.

performance of Mist and Mist2 is comparable. However, Mist2 provides more resilience to route failures in the overlay since routers are not restricted to a fixed hierarchy (as the case in the original Mist). Furthermore, the center ring in Mist2 consists of a cluster of Lighthouses. If a specific Lighthouse fails, then another Lighthouse can be selected and used for resuming communication. The ring-based structure of Mist2 provides better scalability since Mist2 Routers only need to keep track of neighbors in adjacent rings, rather than all routers in the system.

5. Conclusion

The paper describes an efficient infrastructure for privacy-preserving communication called Mist2. Mist2 uses a self-organizing overlay network of rings to significantly reduce communication overheads in comparison to a Crowds-like implementation. The expectation here is that other overlay implementations that are not topology-aware, like the Tor network [9], exhibit similar behavior to the Crowds-like implementation, since each overlay hop can potentially span the diameter of the Internet. While the proposed approach provides more efficient routes, it does not sacrifice anonymity. Moreover, through the use of rings, routers in a particular ring only need to be aware of routers in adjacent rings, thereby making our approach scalable to a Crowds-like solution. Hence, Mist2 provides a scalable, efficient, and distributed approach to providing privacy preserving communication on the Internet.

References

[1] Sherwood, R.; Bhattacharjee, B. and Srinivasan, A. "P5: A Protocol for Scalable Anonymous Communication". *IEEE Symposium on Security and Privacy*, 2002.

[2] "SafeWeb." http://www.safeweb.com.

[3] "Anonymizer." http://www.anonymizer.com.
[4] Reiter, M. and Rubin, A. D. "Crowds: Anonymity for Web Transactions". *ACM Transactions on Information and System Security (TISSEC)*, 1 (1998).
[5] Al-Muhtadi, J.; Campbell, R.; Kapadia, A.; Mickunas, D. and Yi, S. "Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments". *International Conference of Distributed Computing Systems (ICDCS 2002)*, Vienna, Austria, 2002.
[6] Reed, M.; Syverson, P. and Goldschlag, D. "Anonymous Connections and Onion Routing". *IEEE Journal on Selected Areas in Communication, Special Issue on Copyright and Privacy Protection*, 1998.
[7] Shields, C. and Levine, B. N. "A Protocol for Anonymous Communication over the Internet". *7th ACM Conference on Computer and Communications Security (CCS)*, N.Y.: ACM Press, 2000, pp. 33–42.
[8] Chaum, D. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". *Communications of the ACM Transactions on Information and System Security*, 24 (1981).
[9] Dingledine, R.; Mathewson, N. and Syverson, P. "Tor: The Second Generation Onion Router". *13th USENIX Security Symposium*, 2004.
[10] "Gnutella." http://www.gnutella.com/.
[11] "Kazaa." http://www.kazaa.com.
[12] Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M. F. and Balakrishnan, H. "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications". *ACM SIGCOMM*, San Deigo, CA, 2001.
[13] Ratnasamy, S.; Francis, P.; Handley, M.; Karp, R. and Shenker, S. "A Scalable Content Addressable Network". *ACM SIGCOMM 2001*, San Diego, CA, 2001.
[14] Freedman, M. J. and Morris, R. "Tarzan: A Peer-to-peer Anonymizing Network Layer". *9th ACM Conference on Computer and Communications Security*, Washington, D.C., 2002.
[15] "CAIDA's Skitter Data." http://www.caida.org/tools/measurement/skitter/.
[16] Francis, P.; Jamin, S.; Jin, C.; Jin, Y.; Shavitt, D. R. Y. and Zhang, L. "IDMaps: A Global Internet Host Distance Estimation Service". *IEEE/ACM Trans. on Networking*, 2001.
[17] Shankar, N.; Komareddy, C. and Bhattacharjee, B. "Finding Close Friends over the Internet". *ICNP*, Riverside, California, 2001.
[18] "University of Oregon's Route Views Project". http://www.antc.uoregon.edu/route-views/.

بنية تحتية فعالة لحماية الخصوصية أثناء الاتصال على الإنترنت

جلال الهندي

كلية علوم الحاسب والمعلومات، جامعة الملك سعود،
الرياض، المملكة العربية السعودية

(قدّم للنشر في ١٦/٠٩/٢٠٠٦م؛ وقبل للنشر في ١٣/١٢/٢٠٠٦م)

ملخص البحث. يُقدِّم هذا البحث شرحاً لنظام Mist2 والذي يُمثل بنية تحتية لحماية الخصوصية أثناء الاتصال على الإنترنت. في هذا النظام تكون الأطراف المتصلة موزعة على حلقات دائرية، تشترك في المركز وتتفاوت في طول نصف القطر. نظام Mist2 هو تطوير لنظام Mist بحيث يوفر عدة فوائد إضافية منها تقليل العبء الاتصالي، وتقصير طول مسارات نقل البيانات، ومقاومة الأخطاء في توصيل البيانات، وتوفير قدر أكبر من الخصوصية للمُستَقبِل والمُرسِل في آنٍ واحد. كما يتسم النظام المُقترَح بفاعلية عالية، وسرعة في الأداء. هذا وقد أشارت نتائج محاكاة النظام المُقترَح إلى سرعة أداء النظام مقارنة بالأنظمة الأخرى المشابهة مع الحفاظ على نفس القدر من حماية الخصوصية. كما أشارت النتائج أن سرعة الأداء أفضل بشكل ملحوظ من الحلول التي تعتمد على نقل البيانات عبر مسارات طويلة.