Dynamic Multilayer Routing to Achieve Location-Hiding

Hakem Beitollahi, Geert Deconinck
Katholieke Universiteit Leuven, Electrical Engineering Department, Kasteelpark Arenberg 10, Leuven, Belgium
{Hakem.Beitollahi and Geert.Deconinck} @esat.kuleuven.be

Abstract

Overlay networks are used as proxies which mediate communication between an application and its users without revealing the application’s location (IP address). The capability that users can communicate with an application without knowing its location is called location-hiding. Although recent years have provided little literature about location-hiding such as Tor or anonymous web publishing, most Internet activities where anonymity is desired require only sender and relationship anonymity; thereby location-hiding needs more academic effort.

This paper proposes a novel architecture to achieve location-hiding. We describe the design of a dynamic multilayer routing (DMR) where users can communicate with an application without knowing any information about its location (its IP address). The essential factors of DMR protocol are multi-layering, reconfiguration and host-diversity. The goal of DMR is to overcome or reduce several drawbacks of static structure based techniques.

Through analytical analysis, this paper provides a detailed study of DMR architecture and shows that DMR is completely strong against penetration attacks. Our analysis shows that attackers have a negligible chance (e.g., $10^{-8}$) to penetrate the architecture and disclose the application’s location.

1 Introduction

How can users communicate with an application without revealing the location of the application? This paper answers this question. The capability that users can connect to an application without having any knowledge about its location (IP address) is known as location-hiding or application hiding, and the essence of that is indirect communication. Location-hiding was introduced as resistant to distributed DoS attacks since these attacks depend on the knowledge of their victim’s IP addresses [7, 2, 14, 3, 4, 13, 12]. Location-hiding has also been recommended for preserving the anonymity of the services which need to resist censorship such as for dissidents or journalists publishing information accessible from anywhere.

A complementary problem, hiding user identity, has been well studied as anonymous routing since the early 1980’s. In anonymous routing, users may wish to make their communication anonymous. For example in online voting, users may not wish to reveal their identity. Onion Routing [10] is at the heart of several prior works on anonymous routing. Using Onion Routing, a user can connect to an application through a set of relays and thereby hide its location. Layered encryption ensures that each hop in the relay network can only decrypt the address of its successor in the relay chain. Furthermore, many researchers exploit peer-to-peer overlays in anonymous communication, including Tarzan [6], AP3 [8], MorphMix [11], and Cashmere [16]. However, the problem of location-hiding is vice versa; i.e., we are not interested to hide location of senders (users), instead we are interested to hide location of receivers (application). The first key difference between these two problems is that there are many users in the anonymous routing problem, while there is only one application in the location-hiding problem. The second key difference is that in the anonymous routing problem (user-hiding), users initiate the communication; as a result, some schemes such as Onion Routing require that senders construct a route to the receiver beforehand. This is infeasible in location-hiding problem. The key differences show that the solutions to the anonymous routing problem cannot apply directly to the location-hiding problem. However, they can give us insights to design solutions to the location-hiding problem.

Overlay networks have been proposed to mediate communication between applications and their users without disclosing application’s location [2, 3, 4, 7, 13, 14]. The techniques based on static structures such as proposed architecture by the SOS, cannot hide application’s location, because attackers can monotonically penetrate the structure and find location of the application.

This paper presents dynamic multilayer routing (DMR) architecture which invalidates the information that has been extracted by the attackers. In contrast to static approaches, DMR makes location-hiding feasible against penetration at-
attacks. Multi-layering, reconfiguration and host-diversity are the three key factors in designing the DMR architecture.

This paper evaluates analytically the DMR architecture. We develop a Markov model of the architecture to characterize the dynamic behavior of the system, exploring in particular how attacks affect the ability of DMR architecture. This paper only focuses on penetration attacks which exploit the connection structure of the architecture, and use directed penetration in an attempt to reveal the information stored on a victim host. The analysis shows that the proposed protocol is strongly secure against penetration attacks. This paper does not consider smart traffic analysis attack where attackers can observe whole or large portion of network traffic.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes the attack model. Section 4 presents dynamic multilayer routing architecture. Section 5 provides the mathematical analysis of the scheme. Section 6 discusses overhead and important issues of the architecture. Finally, section 7 concludes the paper.

2 Related Work

Many researchers use location-hiding technique indirectly to protect internet applications from DoS attacks. They explore the use of overlay networks to mediate communication between an application and its users. The application (target) chooses randomly few overlay nodes as secret nodes. These secret nodes are responsible to deliver users’ traffic to the target. Overlay network must route any legitimate packet towards the secret nodes via an algorithm that guarantees not disclosing the location of secret nodes. Secure Overlay Services (SOS) [7] is one of the first approaches that present this idea. Mayday [2] generalizes SOS, and proposes an expanded set of overlay routing and filtering mechanisms, allowing a tradeoff between security and performance. Like SOS, Mayday’s filtering mechanisms also relies on a secret authenticator that may be globally transferable across different clients. Internet Indirection Infrastructure (i3) [14] is another architecture that uses location-hiding technique. I3 uses the Chord overlay [15] network to hide an application’s location. In Fosel [3] and OPL [4] architectures, location of green nodes must be remain secret to have successful architecture. In all architectures that we mentioned, secrecy of hidden nodes is the success key of the approaches; since once hidden nodes’ locations are revealed, the architectures are crashed. All mentioned architectures are vulnerable to penetration attacks, and as they are based on static structures, attackers can monotonically penetrate the system and reveal the address of hidden nodes.

Tor [5] is a distributed low-latency anonymous communication that has deployed in 2004 to provide hidden services. It was introduced to protect hidden services from distributed DoS attacks. The hidden service design is based on connecting two circuits - one of them created by the user, the other by the hidden server - on a commonly agreed Tor relay (rendezvous point). The rendezvous point acts as message relay by forwarding outgoing messages from the user-side circuit to the server-side circuit and vice versa.

3 Attack Model

We focus on penetration attacks, which include a wide range of attacks that can reveal the information stored on the victim host, or allow attackers to eavesdrop network traffic of a host. Such attacks can be done remotely by exploiting vulnerabilities on a host. In our attack model, adversaries are outside the overlay network, and have all information publicly available to users.

As this is the first version of DMR architecture, we only consider above attack model. In other words, this paper only consider host compromising attacks, and thus does not characterize other forms of attacks such as traffic analysis attack [9]. Making architecture robust against traffic analysis attack and evaluating architecture against other forms of attacks are our future work.

4 DMR Architecture

The goal of dynamic multilayer routing (DMR) architecture is to allow communication between an application and its users without revealing the application’s IP address. Often overlay networks are proposed as proxy networks to mediate communication between the application and its users without revealing the application’s location. Figure 1 shows a generic proxy network-based location-hiding scheme. As shown in figure 1, application is hidden behind the proxy network. On one side, a set of proxy nodes is connected to the application and on the other side of the proxy network, a selected set of proxy nodes known as edge proxies publish their IP addresses, providing application access to users. Any user that wants to connect to the application, accesses an edge proxy, then the edge proxy routes the user’s traffic through the proxy network toward the application; thereby the IP address of application is kept secret.

In DMR protocol, a user makes a connection with the application through a sequence of layers which periodically are reconstructed. Each layer of proxies constructs a layer of proxies for itself and only accepts traffic from that layer. Each layer only knows its predecessor and successor layers, and a compromised layer as explained later cannot jeopardize the architecture. The construction of d-layer architecture is developed piece by piece in the following subsections.
4.1 Design Rationale

First, the application selects randomly a few overlay nodes as its proxies. These chosen proxies are called first defense layer. This layer is the innermost layer and as we want to construct \( d \) layers, let us to call it layer 0. The application informs the nodes of this layer of their task; hence if a packet reaches this layer and is subsequently verified as coming from valid layer (as we see next), the proxies of this layer forward the packet to the application.

Now, layer \( d \) constructs a layer of proxies for itself. In fact, proxies of layer \( d \) with an agreement algorithm select randomly few nodes as their proxies. This is layer \( d-1 \) and we call it child of layer \( d \) (and correspondingly layer \( d \) will be layer \( d-1 \)'s parent). Similarly, proxies of layer \( d \) inform their children of their task; i.e., if a packet arrives to layer \( d-1 \) and is verified as coming from its child, layer \( d-2 \), the proxies of layer \( d-1 \) forward the packet to their parent, i.e., layer \( d \).

In the same way, any layer constructs iteratively its child layer; i.e., layer \( i \) constructs layer \( i-1 \). This procedure is continued until \( d \) layers of proxies is constructed. Figure 2 shows final architecture of DMR. The outermost layer is called layer 1 which its parent is layer 2 and iteratively parent of layer 2 is layer 3. Upon arriving a packet to a layer, that layer is responsible to verify that it has come from its child and in the case of validity, it must forward the packet to its parent. It is important to know that any layer only knows its child layer and its parent layer. Hence, layer \( i \) does not know layer \( i+2 \) and layer \( i-2 \). Consequently it is important to know that only layer \( d \), the innermost layer, knows application’s IP address. On the other side of proxy network, the proxies of outermost layer layer 1, publish their IP addresses to the edge proxies and ask them if they have requests for the application, forward the requests to them. Letting each layer selects proxies of next layer, makes DMR highly scalable because a layer only has to manage its local environment. Independent of the system size, a layer only cares about a relatively small number of other nodes at any time. In the rest of this paper, when we say a layer, we mean proxies of layer.

4.2 Routing Action

DMR is constructed on top of an overlay network where the overlay network runs on top of existing Internet infrastructure. The routing mechanism from a user to the application is as follows. Once a user delivers its traffic to an edge node, the edge node routes user’s traffic through overlay via several hops (depend on routing algorithm of overlay) to the proxies of layer 1. Layer 1 routes user’s traffic through overlay to layer 2 and so on; finally user’s traffic reaches layer \( d \) (the innermost layer) which delivers user’s traffic to the application. The application site replies to users through DMR tunnel but in the opposite direction, i.e., the application site delivers its traffic to layer \( d \) which layer \( d \) delivers it to layer \( d-1 \) and so on.

4.3 Local Communication between two Successive Layers (parent and child)

Child proxies communicate periodically with parent proxies to check relationship status. Child proxies with the period of \( T_s \) send status query to parent proxies and ask them "are you still our parents?". Parent proxies reciprocally confirm their relationship and answer "Yes" to their query. Child proxies keep contact information (IP addresses) of parent proxies as long as parent nodes answer status-query. Child nodes become orphan nodes when they do not receive any respond from their parent nodes within two successive periods. An orphan node cleans up parent’s contact information and erases (destroys) contact history with parent nodes (e.g., delete cookies, temporary files and so on). This means that if an orphan node is compromised by attackers (e.g. hacking or eavesdropping), it cannot jeopardize location of parents.

On the other hand, if a parent layer does not receive status-query from its child layer within two successive periods, it is no longer interested to its child layer and does not respond to it anymore; i.e., the child layer becomes orphan. In this situation, the parent layer selects randomly few other nodes as their proxies (a new child layer).
new child layer makes new proxies for itself and thus iteratively a sequence of new layers from parent layer towards edge proxies is reconstructed as explained above.

4.4 Architecture Reconfiguration

For necessity of reconfiguration, let us to discuss how the application’s location is concealed. As explained above, the IP addresses of edge proxies are clear to public and also to attackers; thus attackers start from an edge proxy. They use host compromising mechanisms to take control of the edge proxy. Once the edge proxy is compromised, the location of all its neighbors is exposed; thereby the IP addresses of outermost layer’s proxies are revealed. By knowing location of outermost layer’s proxies, attackers try to compromise an outermost layer’s proxy and thereby expose the location of layer 2’s proxies. Iteratively, by compromising a current layer’s proxy and then revealing location of next layer’s proxies, attackers can penetrate the structure. In fact by compromising a sequence of exposed proxies along a path from an edge proxy to the application, attackers can penetrate the structure and eventually expose the application’s location.

To tackle this problem, DMR uses reconfiguration technique. By reconfiguration, DMR changes location of proxies of each layer periodically. The period which new proxies replace old proxies is called reconfiguration rate. In fact, by reconfiguration, proxies of each layer migrate periodically from one host to another host inside the resource pool. It means that proxies of each layer change location (change IP address), thereby we say new proxies (with new IP addresses) replace old proxies. Proxy migration allows exposed proxies to move to locations unknown to attackers, and thereby the information gathered by attackers become invalidated. Reconfiguration starts from innermost layer and like a domino game the old layers of proxies are destroyed one by one from innermost layer to outermost layer. Instead, the new layers of proxies are replaced. In fact, the tunnel is rebuilt completely, \( T' = (d', d' - 1, \ldots, 2', 1') \). This procedure is continued permanently with reconfiguration rate. While reconfiguration rate is sufficiently fast, DMR protocol guarantees that application’s location is kept secret.

4.5 Avoiding correlated vulnerabilities

We show in the above, with reconfiguration, DMR can effectively hide application’s location. However, this is true under assumption of uncorrelated host vulnerabilities. Sometimes, hosts’ vulnerabilities may be correlated. Those hosts that use similar software components (e.g. communication interfaces), operating systems (OS), hardware, and even similar configurations may have similar vulnerabilities. We call this, correlated host vulnerabilities. As proxies run on top of hosts, correlated proxies refer to correlated hosts as well. As a result of correlated hosts, compromising a host (proxy) may significantly increase the chance (speed) of compromising other hosts (proxies) which share similar vulnerabilities. For instance, if compromising first host takes one week, compromising correlated hosts may only take few hours. In this situation, even reconfiguration cannot hide application’s location. In other words, reconfiguration can hide application’s location if and only if done faster than compromising correlated hosts. I.e. if compromising correlated hosts takes few hours, reconfiguration rate must be in few hours too. However, when compromising a correlated host takes only few minutes, reconfiguration cannot hide application’s location at all. So, we should not look solution to this problem in probing reconfiguration rates. The best solution to this problem is "avoiding correlated vulnerabilities" which explained in the following.

Surly, all hosts in resource pool are not correlated. Several hosts use different software components, OS, hardware and other attributes. Thereby, we group hosts into domains. Within a domain, hosts use similar software components, OS and so on. In other words, hosts belong to a domain share similar vulnerabilities. Across domains (different domains), hosts differ in software components, OS, and other attributes, thereby two hosts from two different domains are uncorrelated. The number of domains is a measure of host diversity in the architecture. Each layer is randomly selected from a domain which two successive layers are not selected from same domain. In fact, a layer that selects proxies of its child layer is aware from its domain; thereby it selects surely child proxies from another domain. In fact, never two successive layers are selected from identical domain. For instance, suppose hosts in resource pool are classified into four domains: domain 1, 2, 3 and 4. The Application site selects proxies of layer d from domain 1. Proxies of layer d-1 are selected from another domain (2, 3 or 4). Once proxies of layer d-1 were selected from domain 3, proxies of layer d-2 are selected randomly from domain (1, 2 or 4) and so on.

4.6 Some additional points

- Proxies of each layer publish their public keys to child proxies. Child proxies first encrypt message with parent’s public keys and then forward it to parent proxies.
- When new proxies replace old proxies, the in-flight packets on the old proxy might be lost, and thus may cause temporary service disruptions. However, reconfiguration procedure takes only few seconds which it is negligible for daily or hourly, i.e. "high" reconfiguration rate. However, if a message lost during reconfiguration time, the user can retransmit the message.
5 Analytical Analysis

First, we model system state of each node and second, we present Markov model of DMR architecture and analyze it. By the Markov model, we study capability of DMR for Location-hiding. Moreover, this section analyzes architecture properties such as routing-depth and reconfiguration rate on location-hiding. We also analyze effects of correlated host vulnerabilities on location-hiding. Let us to call number of layers as routing-depth.

5.1 System State of a node

A host has two states: compromised and intact. A host is called compromised when attackers take control over it. In other words, when attackers hack a host or they can observe the host’s traffic (eavesdropping attack), the host is compromised. A healthy host that is not compromised is intact host. Based on host state, an overlay node can fall in three states: intact, exposed, and compromised. An overlay node run on intact host is called intact node, while when it run on compromised host is called compromised node. A node is exposed when its location is known to attackers, therefore it is subject to future host compromised attacks. A compromised node leads to expose location of neighbor nodes. Since attackers can steal information from compromised node, including the location information of all its neighbor nodes, all the nodes that are neighbor with a compromised node are exposed. Figure 3 shows transitions between these states.

An intact node becomes exposed when one of its neighbors becomes compromised. An exposed node becomes compromised when attackers take control of it. There is no direct transition from intact state to compromised state, because, first, location of node is disclosed (exposed state), then it will be subject to attacks and then finally it will be compromised. A compromised node that takes resource recovery, but does not change location, returns to exposed state. While a compromised node that takes resource recovery and simultaneously changes its location, returns to intact state. An exposed node that changes its location returns to intact state.

5.2 Analysis of DMR architecture

The goal of attackers is to discover the IP address of application protected by DMR architecture. The strategy is to explore the structure of architecture and compromise proxies along a routing path towards the application. First, let us to analyze DMR architecture without reconfiguration.

DMR Without reconfiguration

As location of edge proxies is well known to public, edge proxies are always exposed. Attackers start from an edge proxy and try to compromise the edge proxy. Once the edge proxy is compromised, the IP addresses of layer 1’s proxies are exposed. Now, attackers try to compromise a proxy of layer 1 (one proxy from each layer is enough to penetrate the structure). Once a proxy of layer 1 is compromised, the location of all proxies of layer 2 is exposed. By compromising a sequence of exposed proxies along a path from an edge proxy to the application, attackers can penetrate the architecture, and finally expose the application’s location. The state diagram that results for DMR architecture without reconfiguration is shown in figure 4. As can be seen, first, state of the edge proxy is “exposed” and state of all layers is “intact”. Thereby, the system begins in “exposed” state and, upon an edge proxy is compromised with failure rate of $\lambda_0$, state of edge proxy transits from “exposed” to “compromised”; meanwhile state of layer 1’s proxies transits from “intact” to “exposed” with the same rate, $\lambda_0$. Upon a proxy of layer 1 is compromised with failure rate of $\lambda_1$, state of layer 1 transits from “exposed” to “compromised” and in the same time, state of layer 2 transits from “intact” to “exposed” with the rate of $\lambda_1$. The sequence of “intact $\rightarrow$ exposed $\rightarrow$ compromised” is continued until a proxy of innermost layer is compromised, thereby the location of application is revealed.

The Markov model of DMR architecture without reconfiguration can be illustrated as shown in figure 5.

In this figure, $\lambda_0$ is failure rate of edge proxies, $\lambda_1$ is failure rate of layer 1’s proxies and iteratively, $\lambda_{d-1}$ is failure rate of layer d-1’s proxies. Let us for simplicity, assume

$$\lambda_0 = \lambda_1 = \lambda_2 = \ldots = \lambda_{d-1} = \lambda$$

By using Laplace transforms and then taking inverse Laplace transform, the probability state of $p_i(t)$ for $0 \leq i \leq d-1$ is calculated as follow

$$p_i(t) = \frac{(\lambda t)^i}{i!} e^{-\lambda t} \quad (1)$$
\[ R(t) = \sum_{i=0}^{d-1} \frac{(\lambda t)^i}{i!} e^{-\lambda t} \quad (2) \]

Consequently, the probability that location of application is disclosed, unreliability, would be

\[ Q(t) = 1 - \sum_{i=0}^{d-1} \frac{(\lambda t)^i}{i!} e^{-\lambda t} \quad (3) \]

Figure 6 shows the plot of the probability of revealing application’s location \( Q(t) \) versus time for different routing-depths. The scale of time in this figure and all other figures in this section is day. Failure rate is hold fixed at 0.1 per day. Figure 6 demonstrates that if more layers are constructed between edge proxies and application, the probability of exposing application’s location is decreased. In other words, routing-depth has a strong impact on location-hiding, i.e., the more layers, the better resilience to keep application’s location secret.

**DMR with reconfiguration**

When reconfiguration is taken, as explained above, attackers must start again from an edge proxy. If we suppose reconfiguration rate is \( \mu \), at any point of state diagram represented in figure 4, we return to first state with the rate of \( \mu \). The resulting Markov model of DMR architecture with reconfiguration can be demonstrated as shown in figure 7.

The reliability of architecture in Laplace domain is

\[ R(s) = \frac{d-1}{s + \lambda + \mu} \left( \frac{\lambda}{s + \lambda + \mu} \right)^d, \quad d \geq 2 \quad (4) \]

Unfortunately, there is no explicit inverse Laplace transform for equation (4), thereby we are unable to represent reliability formula in time domain. However, we use Relex Software [1] to calculate \( R(t) \) and \( Q(t) \) of Markov model represented in figure 7.

Figure 8 plots \( Q(t) \) versus time for different routing-depths when reconfiguration is taken. \( \lambda \) is kept fixed at 0.1 per day and reconfiguration rate is 10\( \lambda \). Figure 8 shows how reconfiguration affects the probability of discovering application’s location. It shows that when compromising a host takes on average, 10 days, if reconfiguration is taken once per day \( (\mu = 10\lambda) \), then the probability to disclose location of application even for 3 layers \((d=3)\) would be below 10\%. While if 9 layers are constructed between edge proxies and application, this probability would be around zero \((10^{-7})\). This indicates that reconfiguration within sufficient layers can effectively provide location-hiding.

**Impact of reconfiguration rate**

Figure 9 shows plot of \( Q(t) \) in logarithmic scale versus time for different reconfirmation rates. Routing-depth is kept fixed at 5. This figure indicates that when reconfiguration is slow \((\mu = 1/2\lambda)\), the chance of attackers is higher to disclose application’s location. On the other hand, when reconfiguration rate is sufficiently fast, the chance of
In equation (5), when $\alpha$ is high (i.e. high correlation), the reliability of DMR system is degraded significantly. In equation (5), we assume no reconfiguration is taken. As there is no explicit inverse Laplace transform for situation when reconfiguration is considered, we omit to present $R(t)$ formula when reconfiguration is considered. However, by using Relex software we can calculate $R(t)$ and $Q(t)$ for both reconfiguration and non-reconfiguration states. Figure 10 plots $Q(t)$ versus time when correlated host vulnerabilities is considered. In this figure we hold $\lambda$ fixed at 0.1 per day, $\mu$ fixed at 1 per day. Moreover, we consider high correlation vulnerabilities ($\alpha = 0.1$); i.e. when compromising first host takes on average 10 days, the compromising of correlated hosts takes on average 4 hours. As shown in figure 10, without reconfiguration, curves of different routing paths overlapped. This means that with correlated host vulnerabilities, routing-depth is neutral factor. However, reconfiguration can provide little defense.

Figure 8: $Q(t)$ vresus time for different routing-depths when reconfiguration is taken

disclosing application’s location would be negligible. For instance, when reconfiguration rate is changed from $1/2\lambda$ to $10\lambda$, the chance of attackers to discover application’s location is diminished on average 670 times. In fact, it is clear that reconfiguration rate is a critical factor in DMR architecture. This experiment indicates that reconfiguration rate must be sufficiently fast to have an effective architecture for location-hiding.

Figure 9: $Q(t)$ vresus time for different reconfirmation rates

Impact of correlated host vulnerabilities

As shown in above when a) sufficient layers are constructed (e.g. $d \geq 5$) and b) sufficient reconfiguration rate is considered (e.g. $\mu \geq 2\lambda$), the location of application is guaranteed to remain secret. However, this is true under assumption of uncorrelated host vulnerabilities. When vulnerabilities of hosts are correlated, we have

$$\lambda_0 = \lambda$$
$$\lambda_i = \alpha \lambda, \ 1 \leq i \leq d$$

The reliability $R(t)$ of DMR for correlated host vulnerabilities is

$$R(t) = e^{-\lambda t} + \sum_{i=1}^{d-1} \frac{\alpha^{i-1}}{(\alpha - 1)^i} [e^{-\lambda t} - e^{-\alpha \lambda t}]$$

$$- \sum_{i=1}^{d-1} \sum_{j=1}^{i-1} \frac{\alpha^{i-1}(\lambda t)^j}{j!(\alpha - 1)^{i-j}} e^{-\alpha \lambda t}$$

(5)

6 Discussion

Four important facts can be derived from mathematically analysis represented in last section.

1. Reconfiguration is the heart of DMR architecture. In other words, without reconfiguration, location-hiding cannot be achieved.

2. Number of layers between users and application is an essential factor. In other words, routing-depth plays a major role in location-hiding. The more layers, the better resilience to achieve location-hiding.

3. Reconfiguration rate is a critical factor to achieve location-hiding in DMR architecture. Reconfiguration rate must be sufficiently fast to have efficient DMR architecture.

4. Correlated host vulnerabilities must be avoided; otherwise DMR cannot provide location-hiding. The solution is host diversity.
Overhead of architecture

As pointed above, DMR architecture runs on top of an overlay network. Moreover, we show that routing-depth is a key factor in DMR architecture. However, large routing-depth may incur performance penalty to the application. In a normal overlay network, if lookup a query takes $O(k)$ steps, by applying DMR architecture, lookup a query takes $O(d.k)$ steps, where $d$ is routing-depth. For instance, lookup a query in Chord overlay network takes $O(\log N)$ steps, where $N$ is the number of nodes in the overlay; when DMR architecture is added to provide location-hiding, a query lookup takes $O(d.\log N)$ steps.

Although, DMR slows down the lookup action, it provides effectively location-hiding which is necessary in several applications. In other words, although we miss something (speed of lookup), instead we get something better (location-hiding).

7 Conclusion

This paper presents a novel architecture to achieve location-hiding. Dynamic Multilayer Routing (DMR) is a tunnel constructing from multiple layers of proxies. Each layer only knows contact information (IP address, port, and public key) of successive layers, i.e., its child and parent. The tunnel is reconstructed periodically. This paper evaluates analytically the effectiveness of architecture against penetration attacks. Analytical results indicate that when reconfiguration rate is 10 times faster than penetration rate for a DMR with 9 layers, the probability of disclosing application’s location is nearby zero. Improving DMR to be resistance against traffic analysis attack is our future work.

References