Computer Networks 89 (2015) 14–31

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

U-Sphere: Strengthening scalable flat-name routing for decentralized networks

Jernej Kos [a,*], Mahdi Aiash [b], Jonathan Loo [b], Denis Trček [a]

[a] Laboratory for e-Media, Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia
[b] Networks and Distributed Systems Laboratory, Middlesex University, London, UK

ARTICLE INFO

Article history:
Received 16 September 2014
Revised 16 April 2015
Accepted 9 July 2015
Available online 16 July 2015

Keywords:
Compact routing
Decentralized networks
Security
Privacy

ABSTRACT

Supporting decentralized peer-to-peer communication between users is crucial for maintaining privacy and control over personal data. State-of-the-art protocols mostly rely on distributed hash tables (DHTs) in order to enable user-to-user communication. They are thus unable to provide transport address privacy and guaranteed low path stretch while ensuring sub-linear routing state together with tolerance of insider adversaries. In this paper we present U-Sphere, a novel location-independent routing protocol that is tolerant to Sybil adversaries and achieves low $O(1)$ path stretch while maintaining $\tilde{O}(\sqrt{n})$ per-node state. Departing from DHT designs, we use a landmark-based construction with node color groupings to aid flat name resolution while maintaining the stretch and state bounds. We completely remove the need for landmark-based location directories and build a name-record dissemination overlay that is able to better tolerate adversarial attacks under the assumption of social trust links established between nodes. We use large-scale emulation on both synthetic and actual network topologies to show that the protocol successfully achieves the scalability goals in addition to mitigating the impact of adversarial attacks.

1. Introduction

Over the past few years, online social network services have become ubiquitous and at the same time very much centralized in the hands of a few large providers. Such centralization poses severe security and privacy concerns and researchers argue [1–3] that adopting a decentralized peer-to-peer communication architecture would help in mitigating these security threats as control over personal data would in this case remain in the hands of the users and message forwarding would happen only between trusted friends.

So far, several different approaches to decentralized communication between users have been proposed in the literature [1–5]. A building block of any such system is a routing protocol that enables message forwarding between user nodes. In this regard, achieving all of the following design goals at the same time represents an important but elusive step toward practical solutions:

- *Scalability and low path stretch.* As the protocol must support an ever increasing number of users, the amount of per-node state required for routing must grow sub-linearly, $o(n)$. Otherwise, the routers will be overwhelmed by the protocol's memory and processing requirements. At the same time, path stretch (ratio between the length of the path taken by a given routing protocol and the shortest path in the same network topology) must be kept low since path stretch directly affects data forwarding performance. Besides performance, low stretch is also important from an operational standpoint—solutions having an unbounded stretch lack fate sharing [6] and a failure (or an adversary) far from the path can disrupt

* Corresponding author. Tel.: +386 1 479 8243.
E-mail addresses: jernej.kos@fri.uni-lj.si (J. Kos), denis.trcek@fri.uni-lj.si (D. Trček).

http://dx.doi.org/10.1016/j.comnet.2015.07.006
1389-1286/

communication. Ideally, path stretch should be independent of the network size, $O(1)$.

- *Location independence.* In order for the protocol to be practical, user nodes must be addressable by a single known flat identifier, which must be independent of the node's attachment point in the network topology. This enables any higher-layer applications to rely on a known identifier without having to know anything about the underlying network topology.
- *Tolerance of Sybil attacks.* The protocol must be *tolerant* of insider adversaries that are able to create many interconnected nodes inside the network and use them to disrupt the network's normal operation [7–11]. This must be achieved without relying on any central points of trust and with the assumption that the adversary is able to influence the node identifier prefixes of its nodes.
- *Privacy.* The protocol must protect the privacy of users' social contacts. This means that users should not be able to easily infer the social topology or transport addresses of distant nodes by examining the protocol messages. Also, the protocol must not require nodes to disclose their transport address to other nodes that are not their direct trusted neighbors, as this would also compromise their privacy.

Existing solutions all fall short in at least one of the described design goals. Requiring transport address privacy excludes most of the standard DHT designs as their structured topology requires arbitrary connections between nodes that have no direct trust relations. Additionally, standard DHT designs lack Sybil-tolerance by default [12]. Network-layer DHTs like X-Vine [13] can work over arbitrary topologies and can therefore preserve privacy, but as has been shown in [14,15] they cannot provide bounded path stretch. Also, X-Vine fails to provide Sybil-tolerance when the adversary is allowed to influence the prefixes of its node identifiers. Of the practical systems deployed in the wild, we should highlight two pursuing similar goals. Freenet [16] is a peer-to-peer platform for decentralized communication. It uses its own routing protocol, which does not even guarantee message delivery and has been shown to be vulnerable to attacks [17]. CJDNS [18] is a newer routing protocol based on a network-layer DHT design similar to X-Vine with Sybil-tolerance features removed and as such inherits its mentioned problems.

As we will show, state-of-the-art solutions that can achieve both state and stretch guarantees are all vulnerable to Sybil attacks that target name-to-locator resolution. Our contributions in this paper are therefore as follows:

- We present U-Sphere, a novel location-independent protocol that maintains the low state and low path stretch guarantees offered by distributed compact routing protocols while additionally offering stronger resilience against Sybil adversaries. The protocol is scalable due to its compact $\tilde{O}(\sqrt{n})$ routing state and path stretch independent of the network size, $O(1)$. These goals are achieved via key novel features—instead of DHT-based designs or landmark-based location directories, we embed an unstructured record dissemination overlay into the existing topology. Our construction achieves scalable location-independence and Sybil-tolerance at the same time.
- To evaluate our protocol in a realistic environment, we design a distributed emulation testbed that contains a full protocol implementation covering all described signaling. The testbed is designed to run on a cluster of machines in order to support emulation of large networks. For our experiments, we have used up to 9 of the largest Amazon EC2 instances. Using the testbed, we have run extensive emulations of our protocol, using realistic topologies with more than 6000 nodes and more than 16, 000 links.

The rest of the paper is organized as follows. Section 2 first presents the threat model used in our security analysis, together with all the assumptions and a high-level overview of the proposed protocol. Section 3 focuses on base protocol design while Section 4 presents possible attacks and security mechanisms to mitigate them. The protocol is evaluated in large-scale emulation and we analyze the results in Section 5. We survey the related work and compare our protocol with state of the art in Section 6. The paper concludes with Section 7.

2. U-Sphere overview

This section provides a high-level overview of U-Sphere together with the threat model used in our analysis.

2.1. Threat model and assumptions

The protocol makes certain assumptions about the trust encoded into edges between nodes in the network topology. It is assumed that the established edges are based on real-life trust relationships, previously established out-of-band. As the process of establishing an edge requires the exchange and verification of either public key fingerprints or pre-shared keys, the topology should resemble a social network or a web of trust similar to PGP [19], Freenet [16] and CJDNS [18].

Because of this assumption it should be hard for an adversary to establish trust edges to honest participants as it requires social engineering or compromising existing honest nodes. We do not assume that the adversary is clustered in one part of the network (see Fig. 1 for a visualization of

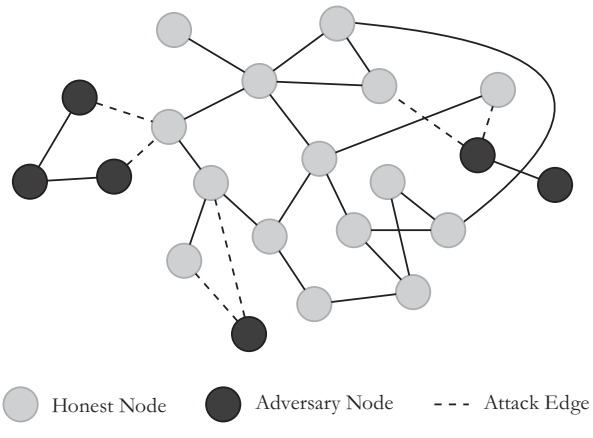


Fig. 1. A visualization of adversary attachment in the assumed threat model. Edges between adversarial and honest nodes are called *attack edges*.

the adversarial attachment topology model). Social engineering or malware attacks that compromise private key material of honest users present a real threat, but protecting against them is beyond the scope of this paper.

The threat model assumes a Byzantine adversary, which means that the adversary can deviate from the protocol in an arbitrary manner, including forging route update messages and generating specific node identifiers for its set of compromised nodes. The adversary has multiple compromised and colluding nodes available inside the network and is allowed to perform a Sybil attack, by introducing additional adversarial nodes and connecting them with existing adversarial nodes in an arbitrary topology. The adversary is only limited in the number of trust edges that he can establish with honest nodes.

2.1.1. Definition of Sybil-tolerance

The protocol aims to achieve *Sybil-tolerance*, which we define in the following manner. An adversary attached to the network topology according to the threat model must not be able to misroute or drop traffic in cases where he is not placed on the shortest path between the source and destination in the network topology. This means that he should not be able to redirect traffic or cause name-to-locator resolution to fail for arbitrary nodes from an arbitrary position in the network topology.

As we will show in Section 6, in existing scalable and low-stretch location-independent routing protocols, an adversary that can influence the choice of its node identifiers is able to disrupt the protocol from any position in the network topology simply by choosing appropriate node identifiers.

2.1.2. Local knowledge and secure size estimation

We assume that each node has only local knowledge of the network topology—a node is only directly aware of its 1-hop neighbors with whom it has established trusted links. In order to be able to adapt to increasing topology sizes, we assume the existence of a secure rough size estimation component running as part of U-Sphere. The estimation does not need to be exact (an order of magnitude is sufficient), and it has to be secure in the sense that an adversary cannot arbitrarily skew the size estimate or cause a denial-of-service attack with little resources. An example of a suitable protocol, based on crypto-puzzles, is presented by Evans et al. [20].

2.1.3. Cryptographic primitives

As the protocol relies extensively on public key cryptography operations that might be expensive to perform for each update message, U-Sphere assumes the use of elliptic curve cryptography based on Curve25519 [21] presented by Bernstein. These primitives have been shown to be very efficient and secure, with lower overheads when compared to RSA [22].

2.2. Protocol overview

We first describe how the protocol performs message routing with location-independent identifiers, and then describe how our specific design of the name-to-locator resolution overlay tolerates possible adversarial attacks.

2.2.1. Location-independent message routing

In order to achieve the performance goals we have specified in the introduction, we base our protocol design on insights from compact routing theory [15,23]. At most $\tilde{O}(\sqrt{n})$ nodes are designated as landmarks in a distributed fashion. By the standard route update process via a path-vector protocol, non-landmark nodes are assigned location-dependent addresses in the form of source routes from nearby landmarks and at the same time learn routes to all the landmark nodes. These addresses alone already enable routing, but in practice, routing always via the landmark nodes will cause the path stretch to become large when nodes are close. To ensure low stretch, each node, via the same path-vector protocol, also learns shortest paths to its closest $\tilde{O}(\sqrt{n})$ nodes. To enable location-independent routing, nodes are placed into groups based on their node identifier prefixes. A name-to-locator record dissemination overlay is constructed for each group in order to be able to resolve node identifiers to current location-dependent addresses, all while keeping the low stretch and state bounds.

2.2.2. Security

As described, the above protocol is not secure. The first attack vector is via the path-vector protocol that is used to learn paths to various nodes in the network. Any intermediate node is able to manipulate route update messages to misroute traffic. In order to prevent this, U-Sphere employs a chained announce delegation scheme where route update messages are cryptographically signed and multi-hop paths are cryptographically protected from being shortened. The second attack vector is the name-to-locator resolution process, which is required to make routing location-independent. Existing location-independent compact routing protocols [15,24] make use of landmark-based location directories that present likely attack targets as adversary-controlled landmarks (which an adversary can generate at will) are able to prevent resolution of arbitrary node identifiers. U-Sphere uses a novel overlay construction that does not require any location directories on landmarks and does not rely on DHT protocols. The overlay is constructed by discovering nearby nodes of the local group and establishing multi-hop overlay links with them. As the links are prioritized based on hop distance, the overlay topology resembles the underlying network with nodes of the other groups removed. This ensures that selection of node identifiers does not influence a node's position in the overlay and thus greatly improves security.

3. The proposed protocol

This section presents the details of U-Sphere, consisting of two complementary components, which together provide efficient Sybil-tolerant and location-independent routing.

3.1. Location-dependent routing

We first present the location-dependent routing component that routes on addresses, which change together with the topology. This component is complemented in the next section, so that the protocol is then able to route directly

on location-independent node identifiers. Here, we first describe how nodes and links are identified in the protocol, what state each node must maintain and then show how the location-dependent routing protocol operates.

3.1.1. Node identifiers

Each node generates a private/public key pair and uses the first 160 bits of the public key's binary representation hashed using SHA-512 as its self-certifying node identifier. This identifier is globally unique among nodes, as creating a duplicate would require either finding a collision for SHA-512 or generating an existing private key, both of which are highly unlikely. However, an adversary can generate an identifier that shares a large common prefix with some other known identifier by trying a large set of public keys, hashing them and selecting the ones that share specific prefixes.

3.1.2. Virtual port identifiers (vports)

Each node establishes direct authenticated transport links with other nodes that it trusts. A locally unique identifier called the virtual port identifier or vport is assigned to each such outgoing link. This identifier is a 16-bit unsigned integer and being only locally unique, the same identifier can easily be used by different nodes to identify different links. Coming from regular network routing, this concept of vports is analogous to interfaces.

3.1.3. Landmarks

U-Sphere requires some nodes to be designated as landmarks. These nodes have no special requirements as far as their operation is concerned—they behave the same as any other node, do not store any additional state and other nodes do not perform any additional queries to them. The only consequence of a node being a landmark is that all other nodes will learn shortest paths to it. Landmark nodes will be used to stitch long-range routing paths through them. In general, a path from source s to destination d via landmark ℓ_d will be stitched from two paths $s \rightsquigarrow \ell_d$ and $\ell_d \rightsquigarrow d$.

Because the protocol requires that all nodes know paths to all the landmarks, the number of landmarks must be limited to $\tilde{O}(\sqrt{n})$ in order to preserve the state bound. This is done in a distributed fashion by having each node decide locally and independently whether to become a landmark or not. Each chooses a number x from the range [0, 1), uniformly at random, and becomes a landmark if $x < \sqrt{(\log n)/n}$ where n is the estimated network size. Following from this, the expected number of landmark nodes will be $n \cdot \sqrt{(\log n)/n} = \sqrt{n \log n}$. By using a Chernoff bound, there will then be $O(\sqrt{n \log n}) = \tilde{O}(\sqrt{n})$ landmarks with high probability. In the context of this paper, an event E occurs *with high probability* if, for any $\alpha \geq 1$, E occurs with probability $\geq 1 - O(n^{-\alpha})$.

Since nodes can join and leave the network at any time, the set of landmark nodes will change through time. To mediate this dynamic, U-Sphere relies on a signal from the size estimation component. Whenever the size estimate changes by a constant factor, a node's state may be flipped and a node becomes or ceases to be a landmark.

3.1.4. Landmark-relative addresses

As mentioned, U-Sphere builds paths via landmarks. To enable construction of such paths, the protocol ensures that each node is assigned a landmark-relative address (L-R address) of length n in the form of $\langle \ell_d, [p_1, p_2, \ldots, p_n] \rangle$. Here, ℓ_d is the node identifier of a landmark node and $p_1, p_2, \ldots, p_n$ is a path of vports identifying links leading from the landmark node to the destination node, enabling a form of source routing. Landmarks themselves have L-R addresses of size zero, as they are always directly reachable by their node identifier.

Given a L-R address, any node s is able to route toward the destination d by first routing via path $s \rightsquigarrow \ell_d$ and then using the source route $\ell_d \xrightarrow{p_1} n_1 \xrightarrow{p_2} n_2 \xrightarrow{p_3} \ldots \xrightarrow{p_n} d$ to reach the destination. Any node can route toward ℓ_d efficiently, because all nodes know the shortest paths to landmarks.

Each node chooses at least one L-R address for itself, based on the closeness (hop count) of landmark nodes in the topology. It may choose more than one address for redundancy. These addresses are location-dependent addresses that change as the topology evolves. We show later how a node can route using location-independent node identifiers, but for now let us assume that each node also knows the destination's L-R address in addition to its node identifier.

It is interesting to quickly analyze the growth of L-R addresses with respect to network size. To see what happens in the worst case, imagine a ring topology where each node has degree 2. If there are n total nodes and $\sqrt{n \log n}$ landmarks, in the worst case all landmarks are clustered one after another and the node with the longest L-R address is located $\lceil \frac{1}{2}(n - \sqrt{n \log n}) \rceil = O(D)$ hops from any landmark, where D is the graph's diameter. However, we show experimentally that in realistic social network topologies and with landmark distributions generated by U-Sphere, L-R addresses are in fact much shorter.

3.1.5. Vicinities

Always routing via landmark nodes can cause high path stretch when the destination node is topologically close to the source. To mitigate this while preserving the per-node state bound, in U-Sphere each node also learns the routes to $O(\sqrt{n \log n})$ topologically closest nodes, based on hop count distance. This set of close nodes of node n is designated the *vicinity* of n and denoted V_n. Using this additional vicinity information, a source node s can route to a destination d directly via $s \rightsquigarrow d$ when $d \in V_s$.

3.1.6. Path-vector route update protocol

To maintain the required routing state described so far, U-Sphere uses a single proactive path-vector protocol. Each node periodically (with period τ_r) announces itself to its neighbors using a route update message. All updates encode the following information:

- *Node identifier* of the originator (the node that generated this update message).
- *Landmark flag*, a Boolean flag indicating whether the originator node is currently a landmark node or not.
- *Forward path*, a path of vports that can be used to forward messages directly to the originator. When receiving an update, each node prepends the vport of the link on which it received the update.

- In case the originator node is a landmark node, the message also includes the *reverse path*. This is a path of vports that can be used to route from the originator toward the current node. Such reverse paths can be used as L-R addresses by nodes, preferring short paths. Before forwarding an update with a valid reverse path, each node first appends the vport of the outgoing link.
- *A sequence number* that is monotonically increasing.

Note that full node identifiers could easily be used instead of just vports in forward and reverse paths. However, we chose not to do this because vports can be more compactly encoded [25] and because their use obscures the nodes' social neighborhood. If node identifiers had been used instead, the social neighborhood of the originator node would be disclosed with each update and this would go against our privacy goal.

When a properly formatted update message is received by a node, it is imported into the routing table only under specific conditions. The conditions are as follows, in order:

(i) If the message originated on the node itself, the route update is immediately discarded.
(ii) If the originator node is not a landmark node and it does not fall into the current node's vicinity, the route update is discarded.
(iii) If the route update contains a new route for a destination not previously seen via a link, the update is accepted.
(iv) If the route update contains a *better* route toward the destination, the update is accepted and the active route is replaced.

The quality of routes is evaluated based on hop count. Any other metric for link quality, latency, "trustedness" or even a composite metric could easily be used instead. Whenever an active route is updated, the route update is also propagated to neighbor nodes. Each node also periodically exports all of its active routes to all neighbors. Due to the above conditions, the flooding of non-landmark announces is limited in scope to a node's vicinity.

On non-landmark nodes, after the route update is accepted and the originator is a landmark node, the local node also performs *L-R address selection*. This process determines its current set of landmark-relative addresses that can be used by other far away nodes to reach it by stitching paths via selected landmarks. The protocol handles the changing of nodes' landmark status gracefully. As the landmark status of the originator is part of regular route update messages, U-Sphere will deal with landmark changes simply by re-evaluating the above conditions for accepting an update based on the changed landmark flag. Due to the second condition, when a node transitions from landmark to non-landmark status, its route updates will no longer be propagated everywhere, but will instead become scope-limited to the originator node's vicinity. Routes at faraway nodes will expire after going $3\tau_r$ without a new update. The reverse will happen when a node transitions from non-landmark to landmark status, as route updates propagate throughout the network. Non-landmark nodes will perform the usual L-R address selection and update their addresses when needed. As landmarks have no other special roles besides being used in L-R addresses, nothing else needs to be done explicitly.

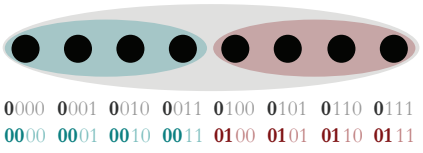


Fig. 2. Size estimate difference by a factor of 2 means a sloppy group prefix p difference of one bit. Shown above is the difference in node groupings between $p = 1$ (gray group) and $p = 2$ (red and blue groups). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.2. Destination L-R address resolution

Until now, we have assumed that when a node wishes to route a message toward some destination, it somehow knows its current L-R address and is able to stitch a proper path. L-R addresses are topology-dependent and can therefore change as new links are established or existing links are removed. In order to be location-independent and route directly on node identifiers, U-Sphere needs to resolve node identifiers into L-R addresses. Achieving this without considering security and path stretch is easy—we could use a DHT overlay to store the mappings. But as mentioned before, we aim higher: We wish to retain the constant-bounded stretch even for the first packet of a flow while not allowing an adversary (capable of choosing node identifiers and launching Sybil attacks) to disrupt name resolution.

The core idea is to group the nodes and create an unstructured overlay embedding for each such group. This overlay is used to disseminate name records containing mappings between node identifiers and nodes' L-R addresses.

3.2.1. Sloppy groups

We use the concept of node "color" groupings from [26] with the adaptation to a more distributed and dynamic setting presented in Disco [15]. Nodes are split into groups based on the value of their node identifiers. Each node takes the first $\lfloor \log_2 (\sqrt{n/\log n}) \rfloor$ bits of its node identifier to represent the identifier of its group. Given a uniform random distribution of node identifiers among honest nodes, this gives us, with high probability (again by a Chernoff bound), $\sqrt{n/\log n}$ groups, each of size $\sqrt{n \log n}$.

Groups are "sloppy" because the group identifier depends on the node's own estimate of n that might differ slightly among the nodes. Sloppy grouping is resilient to small changes in n, as unless the estimate differs by a factor of 2, the grouping stays the same. And even in cases when the estimate differs by a factor of 2, this only corresponds to splitting/merging of a group (see Fig. 2). These properties are important from a performance standpoint, similarly to the notion of *consistent hashing*—a small change in n does not result in a lot of group reorganizations.

3.2.2. Extended vicinity

When the simple vicinity definition is used, each node will store $\sqrt{n \log n}$ routes to topologically nearest nodes. Taking into account the division into sloppy groups and assuming perfectly uniform distribution, we can compute the number of expected nodes of each sloppy group in any node's vicinity. Let S be the set of all sloppy groups and $S_c \in S$ the

set of nodes in sloppy group with identifier c. Then, for any node m and any group identifier c, it follows:

$$\begin{aligned} E[|V_m \cap S_c|] &= E[|V_m|] \cdot P[m' \in S_c] \\ &= \sqrt{n \log n} \cdot E[|S|]^{-1} \\ &= \sqrt{n \log n} \cdot (\sqrt{n / \log n})^{-1} \\ &= \sqrt{\frac{n \log^2 n}{n}} = \log n \end{aligned}$$

This gives us $\log n$ expected nodes of each sloppy group in any node's vicinity. But in practice, topology is also a factor when considering a node's vicinity and it can happen that some vicinities contain less nodes of a certain sloppy group—breaking the perfectly uniform distribution. This is undesirable because U-Sphere assumes that each node has at least one node of each group in its vicinity for proper operation—and having more is better both for redundancy and, as we will see, for raising the probability that the name record dissemination overlay is connected.

In order to ensure that each node has a properly balanced vicinity regarding sloppy groups, we introduce an *extended vicinity*. In addition to the already mentioned criteria for including the nearest $\sqrt{n \log n}$ nodes, U-Sphere also accepts route updates for nodes that may be outside the "normal" vicinity (because they are too far away) but which belong to sloppy groups that are currently under-represented (have less than $\log n$ routes stored). This additional condition for accepting route updates enables us to balance the representation of sloppy groups in each node's vicinity, thus increasing redundancy. It should be noted that even with the extended vicinity, the per-node state bound is not violated. Since the additional state per sloppy group is less than $\log n$ and there are $\sqrt{n / \log n}$ sloppy groups in expectation, the state remains bounded by $\tilde{O}(\sqrt{n})$.

3.2.3. Dissemination overlay construction

After assigning a sloppy group to each node and ensuring that each node has members of all sloppy groups in its extended vicinity, we have to enable that all nodes within a given group learn each others' L-R addresses. To do this, a dissemination overlay topology is constructed for each sloppy group, connecting all of its members and enabling them to exchange L-R address updates (see Fig. 3). Overlay construction proceeds in the following steps on each node a:

(i) The node a maintains a list of sloppy group neighbors together with their L-R addresses. These are the nodes that are in the same sloppy group as a and are close in hop distance in the social topology.
(ii) To discover suitable nodes, node a checks its extended vicinity set for any nodes whose sloppy group identifier matches theirs.
(iii) Upon discovering new members, it sends them record update messages.
(iv) When receiving record update messages from nodes that a does not have in its extended vicinity but belong to the same sloppy group (based on comparing group prefixes), it may establish back-links to them. Each node will establish up to $\log^2 n$ back-links, where the available link slots will be prioritized based on hop distance.

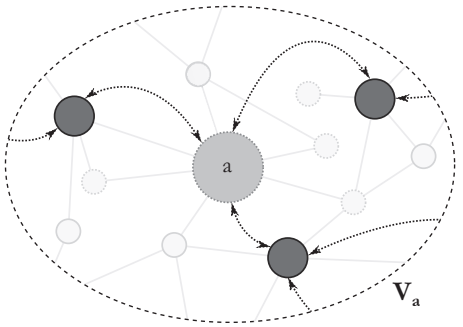


Fig. 3. Dissemination overlay construction for node a. Dark nodes belong to the same sloppy group as a and are part of its extended vicinity, V_a. Dashed lines represent overlay links while light lines represent direct links.

The above algorithm favors establishing links between nodes that are close and well connected in the network topology. It is also simple to implement in a dynamic setting. Neighbors are updated incrementally as part of extended vicinity maintenance in the location-dependent routing component. Back-links are established based on incoming record update messages and expire when no updates have been received through them for some specified period of time. No landmark-based location directories are required for maintenance or for bootstrapping.

3.2.4. Name/locator record update protocol

Once the overlay topology has been established, sloppy group members are able to exchange name record update messages. The aim of these messages is to disseminate up-to-date L-R addresses of all sloppy group members. Each name update message contains the following attributes, cryptographically signed by the originating node in order to prevent modification while in transit:

- *Node identifier* and *public key* of the node that is originating the update message and whose current L-R addresses are included in the update.
- *Timestamp* in originator-local time, which must be monotonically increasing.
- *Sequence number* that is used in case multiple updates are emitted with the same timestamp.
- A list of *currently active L-R addresses* for the originator node.

Each node emits name update messages for its own set of active L-R addresses whenever this set changes (triggered by route updates from landmark nodes). It also transmits the updates periodically to its sloppy group neighbors, with period τ_s. Name update messages describing all the currently known mappings are transmitted whenever a new sloppy group overlay neighbor is detected (either via the extended vicinity or when a new back-link is established). Whenever a name update message with a valid cryptographic signature is received by a node, it is imported only if it meets all of the following criteria:

(i) The node originator belongs to the same sloppy group as the receiving node, based on the receiving node's sloppy group prefix.

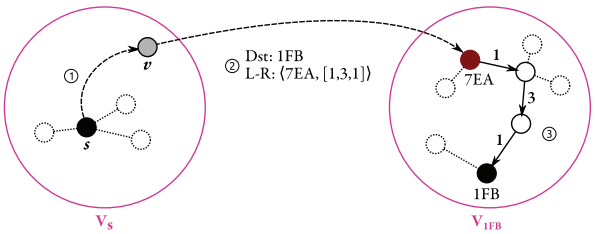


Fig. 4. Routing from node s toward node 1FB in three steps via landmark 7EA when s does not know the destination L-R address: **(1)** L-R address resolution via $v \in V_s$ that is in the same sloppy group as 1FB, **(2)** once the L-R address is known, message is routed toward the designated landmark, **(3)** after the landmark receives the message (and if no shortcutting occurs), source routing is used to reach the destination.

(ii) Name record for this originator either does not yet exist or the newly received name record is more fresh as determined by its timestamp and sequence number attributes.

Stale name records are periodically expired in order to ensure that only fresh and valid records remain. Since the timestamp is cryptographically signed, an adversary cannot propagate stale information.

3.3. Routing decisions

Now that we have both the location-dependent routing component and the name to L-R address resolution component, we can describe how the routing process looks like and how it achieves low path stretch. When a source node s wants to route a message toward some destination d the following scenarios are possible:

- In case d is a landmark node, s can route directly based on d's node identifier as all nodes will know shortest paths to any landmark node.
- Also, when d is part of the (extended) vicinity of s, a shortest path is known and can be taken.
- The last case occurs when s does not know d's current L-R address. In this case, s computes the sloppy group identifier of d by taking a properly sized prefix out of d's node identifier. Then, it searches its extended vicinity for the closest node v that is also a member of d's sloppy group. Due to the construction of extended vicinities such a node will exist and due to exchange of name records via the overlay it will have knowledge of d's current L-R address. Therefore s first routes the message to v that updates the message with a proper L-R destination address and routes it toward its designated landmark. The landmark then uses source routing to reach the destination d. A visualization of this last scenario can be seen in Fig. 4.

Besides using landmark-based source routing, path stretch can be further improved by using shortcutting as in [15,24]. Whenever a node on path $s \rightsquigarrow d$ is encountered that has an active route toward d, this route is followed instead of routing via the source route provided in the L-R address. As the evaluation will show, this leads to improved path stretch at almost no additional cost, while also reducing link congestion. We provide formal proofs of path stretch and routing state bounds in Appendix A and also confirm this behavior in our experimental evaluations.

4. Securing U-Sphere

In this section we analyze the security of U-Sphere and present additional mechanisms to secure the routing protocol.

4.1. Signed route updates

The path-vector protocol works as described, but is inherently insecure when dealing with an adversary defined by the threat model. Without additional protection, any node is able to forge route updates and specifically target paths in route update messages so that they appear to be shorter than they actually are. In this way, adversarial nodes are able to gain control over traffic that would not normally pass through them, which goes against our security goals.

To address this, two modifications to the original path-vector protocol are introduced. First, the originator's *public key* is added to route update attributes and the attributes then get signed by the route originator with its private key. Since a node identifier is self-certifying it is easy for any node to verify that the route update was actually signed by the node listed as the originator and discard any invalid updates. However, this still does not solve the problem that nodes can shorten paths listed in route updates. For example, if a node receives an update with path $[n_1, n_2, n_3]$ where n_3 is the originator node, it can simply truncate the path so that it becomes just $[n_3]$. In this way, the path now appears shorter and will therefore be preferred in routing decisions. One cannot simply sign path attributes as each node that receives and re-announces the update needs to append itself to the end of the path.

To combat this, U-Sphere uses *signed announce delegation chains*. This is a mechanism where each node must explicitly delegate route update announcement privileges to a neighboring node in order for that node to be able to export the route update on originator's behalf. Using node's main key pairs (denoted $\text{Pub}_A/\text{Priv}_A$) for this operation would disclose social topology information in the same way as using node identifiers instead of vports would. This is why U-Sphere establishes security associations (SAs) between neighboring nodes, assigning key pairs to specific links. Nodes on both ends of a link generate and exchange the public SA-Pub_{AB} part of the key. These keys are then used to delegate announce privileges (see Fig. 5) and discard any updates containing paths that fail chain verification. In order to make correlation between SA-derived keys and nodes harder, nodes can use multiple SAs for the same link and rotate them periodically.

4.2. Name resolution

The security of name resolution is based on the sloppy group overlay construction. Each node selects $\log n$ neighbors that share its sloppy group prefix from its extended vicinity to form the overlay. Additionally it also establishes up to $\log^2 n$ back-links to nodes that contacted it. As all the overlay neighbor links are prioritized based on low hop distance in

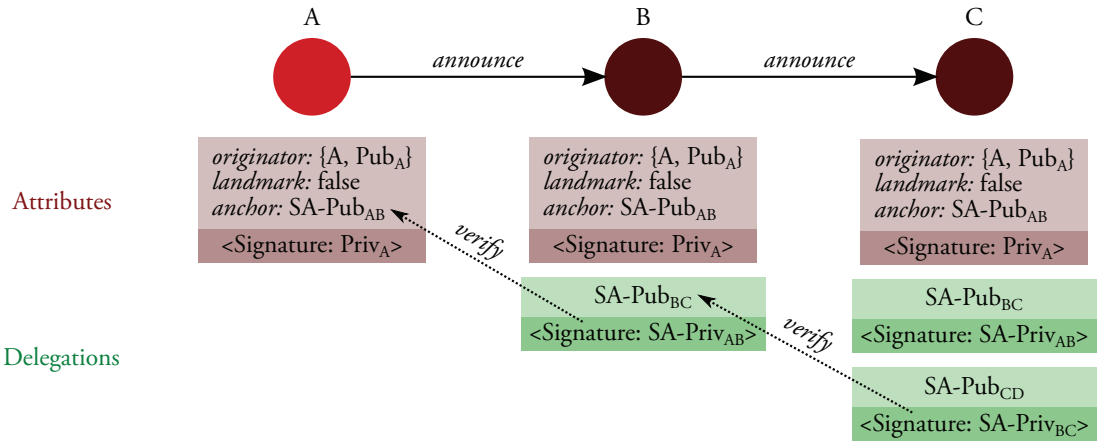


Fig. 5. An explanation of how signed announce delegation chains work. For every neighbor B, the originator A sets the *anchor* attribute to a public key that is part of the security association between nodes A and B, and for which node B knows the private key. It then signs the attributes with its main private key Priv_A. Each following node in the chain receiving the route update then delegates the announce privilege to each neighbor node to which it propagates the update. In this way, a chain of signatures is established and nodes in the middle cannot change the path as they do not know the private keys of previous links—for example node C cannot remove the links $A \rightarrow B$ or $B \rightarrow C$ without invalidating the route update message.

the network topology, a node is more likely to choose close trusted nodes as its neighbors. An adversary is not able to influence this selection by choosing appropriate node identifiers, because due to the prioritization, he also needs to be close in the network topology. And being close requires gaining trust from other users.

The most effective attack would be for an adversary to establish trust edges close to a targeted node, but this requires social engineering of very specific edges in the social vicinity of the target. And even in this case, assuming that the target is well connected to other honest nodes, some of the overlay links may also be established with honest members of the sloppy group. And because of update record distribution, one honest link is enough to ensure that the node learns name records for its sloppy group. Denial of service on specific links is the next logical step, but the adversary cannot learn transport addresses of the link endpoints unless he engineers trust from their users.

Another attack on name resolution is possible when a node chooses a nearby adversarial node as its relay during destination location resolution in its extended vicinity (first step in Fig. 4). In this case, the relay might drop the message and it will never be delivered. To combat this problem, nodes should keep track of how well the relays are performing and deprioritize them for future routing. This is possible as the nodes are free to choose any member of the destination's sloppy group in their extended vicinity as an address resolution relay.

4.3. Landmarks

In contrast to existing scalable low-stretch location-independent routing schemes, U-Sphere does not rely on any special state (like location directories) being maintained on landmark nodes. As an adversary can designate any of its nodes as landmarks, this would give him another attack vector for targeting name resolution.

But an adversary can still cause the nodes in the network to have to hold increased routing state by introducing many landmark nodes into the system. This could be mitigated by requiring crypto-puzzle solutions attached to landmark announces. Even so, the only advantage that an attacker would gain by having more landmarks is that honest nodes near attack edges would be more likely to choose adversarial nodes for its L-R addresses. This might not present an issue because as soon as a message enters the vicinity of a node, it may use shortcutting to be delivered directly to its destination without being routed over the landmark at all. But in any case, nodes are able to choose any nearby landmarks (and may opt explicitly for more trusted ones) and multiple L-R addresses, which together with measurement of how well each of them performs, can mitigate this problem.

5. Evaluation

The following section examines how the U-Sphere protocol performs in practice—specifically we test whether it achieves all the goals outlined in the beginning, namely low per-node state, low path stretch and tolerance of Sybil attacks. Before discussing the results, the methodology used to evaluate the protocol is briefly described.

5.1. Methodology

The evaluation methodology is based on the principles of large-scale emulation [27]. We find this approach the most suitable because it enables us to study a concrete implementation in greater detail than it would be possible had we used only simulation. Specifically, we can measure how message complexity changes through time for both routing components and observe the effect of churn. To ease the testing of U-Sphere implementation, a testbed has been developed, allowing realistic experiments on large topologies. The testbed is general and it can be used—besides performing scientific experiments—as an automated integration test tool during

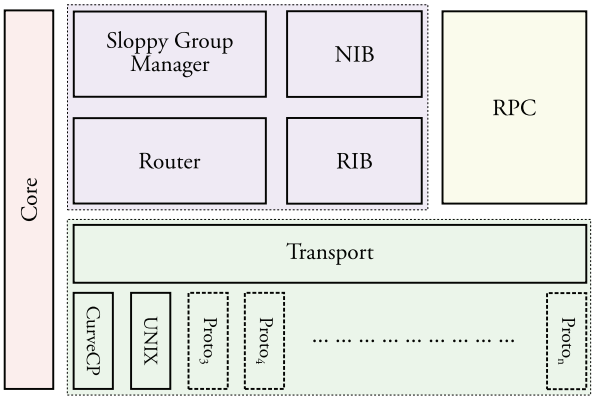


Fig. 6. Components of a U-Sphere node. *Core* (red) implements the ASIO event loop, the *Transport* (green) component handles I/O operations with remote nodes while the *Social* (blue) component handles message routing. NIB and RIB represent the name and routing information bases, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

protocol development. In this section we describe how the testbed environment is implemented and how it is used to run test scenarios.

5.1.1. Protocol implementation overview

U-Sphere implementation is designed as a set of C++ libraries, each library covering one component (see Fig. 6 for an overview of the components and their relations). All communication between components and all I/O operations are implemented by using asynchronous events via the Boost.ASIO library. The implementation is designed for use in multi-threaded applications.

The *Social* component contains a full implementation of U-Sphere as described in Section 3. To exchange messages between nodes, the *Transport* component provides an abstraction layer that exposes inter-node links using a simple message-based API. It currently supports both the CurveCP protocol [28] and UNIX sockets for underlying communication between node processes. Messages are efficiently serialized using Google Protocol Buffers and then dispatched via the underlying transport. The Transport component is modular, so additional transports can be easily implemented. The protocol implementation includes security association establishment between peers and chain verification in route update messages.

5.1.2. Testbed

The testbed is implemented as an application that uses the above libraries for running multiple U-Sphere nodes (implementation instances). Its main role is thus the provisioning of emulated nodes and support for execution of test scenarios on the emulated network. To enable emulation of large networks, the testbed is designed to work on a cluster of machines. For our experiments, we have run the testbed on up to 9 of the largest (`c3.8xlarge`) Amazon EC2 instances connected together into one network segment.

Scenarios are able to invoke test cases at specific points in time. Each test case is designed to be executed in the distributed testbed environment on selected emulated nodes.

Table 1
Topology datasets used for experiments.

Topology	Degree			Vertices	Edges
	Min	Avg	Max		
synthetic-hk	4	6.0–7.98	14–309	16–4096	48–16,362
hyperboria	1	3.97	66	687	1365
as-733-a	1	3.67	592	3015	5539
as-733-b	1	4.29	1460	6474	13,895

This testbed enables us to evaluate the protocol performance under a number of different scenarios on large topologies. The protocol implementation, together with the testbed environment, is available as an open source project [29] licensed under GPLv3.

5.2. Results

In this section we present the results of evaluating U-Sphere under different scenarios and topologies. To generate the synthetic topologies, we have initially used the Barabàsi–Albert preferential attachment model [30] which can generate scale-free networks. But since the B–A model generates topologies with a low clustering coefficient, to obtain more realistic topologies, we have used the Holme–Kim tunable clustering modification [31] where the triangle formation probability parameter has been set to 0.2, giving us degree distributions as shown in Table 1. In order to test the protocol on realistic datasets we have chosen the following additional topologies extracted from actually deployed systems:

- **hyperboria** Topology of the largest deployed CJDNS [18] network (Hyperboria). This topology has been aggregated from multiple observation points around the network.
- **as-733** Two topologies extracted from Internet BGP routers, representing a network of autonomous systems. We use the dataset from [32]. Topology *as-733-a* is from 1997-11-08, while the larger *as-733-b* is from 2000-01-02.

The datasets used in our experiments are available together with the mentioned open source testbed environment. Statistical information about all the used topology datasets is shown in Table 1.

5.2.1. Path stretch

To perform path stretch measurements, two random destinations are selected for each node and *ping* RPCs from source to destination node are made. We record the length of the path taken by actual messages routed using U-Sphere and compare it to the shortest path between the two destinations in the network topology. An important observation in these measurements is that the average size of a node's extended vicinity in 4096-node topologies is only 184 nodes, with about the same amount of landmarks. Together, this amounts to around 8% of nodes, which means that to reach most destinations, direct paths are not known and name resolution needs to be performed (thus the messages need to make the full resolution process as shown in Fig. 4).

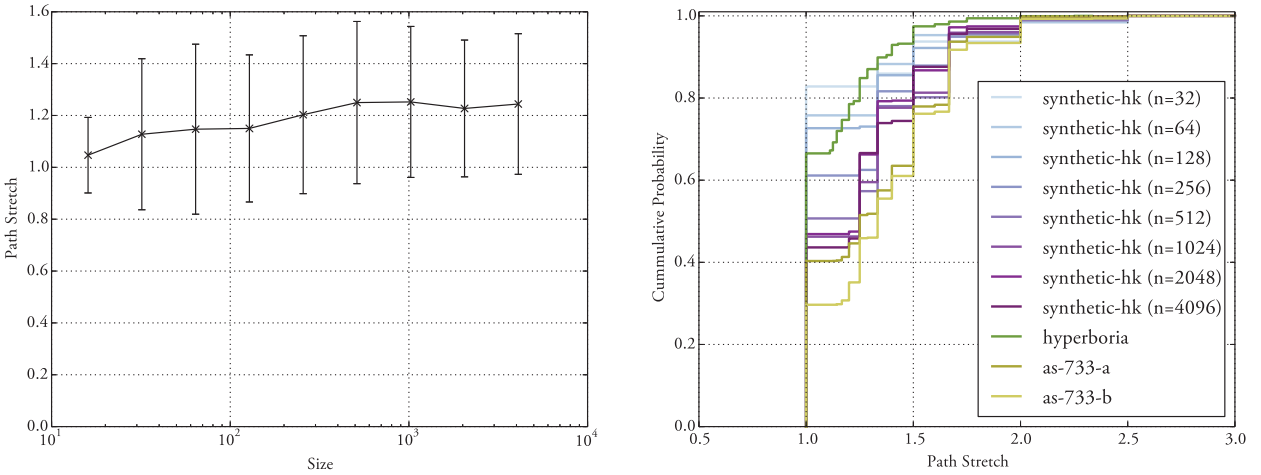


Fig. 7. Path stretch in dependence of varying topologies. Left figure is the average patch stretch plotted using logarithmic scale for topology *synthetic-hk* and the right one shows path stretch distribution with varying topologies (for *synthetic-hk* with increasing topology sizes). Error bars show standard deviations of path stretch.

Results of the measurements can be seen in Fig. 7. We show both, the average path stretch as it changes with increasing number of nodes in the generated topology and the distribution of path stretch over all measured paths. We can see that the path stretch remains low throughout the tests, with averages ranging from 1.05 to 1.25. The maximum path stretch obtained over all topologies is 3.0 and in all cases it covers only a small ($< 0.05\%$) percentage of paths. Stretch distribution is also similar on the three realistic topologies, with stretch being ≤ 2.5 in the *hyperboria* topology and ≤ 3.0 in both *as-733* topologies.

We have also measured variations in path stretch where we examined the influence of community structure (1 through 16 sparsely interconnected communities) and node degrees (average degrees ranging from 2 to 64) on path stretch. The results show that community structure does not affect path stretches, while doubling the average node degree causes a small increase at small degrees but has no noticeable effect when degrees get larger.

5.2.2. L-R address lengths

We mentioned earlier that L-R addresses can grow on the order of graph diameter in the worst case. To test what happens during protocol operation on realistic topologies, we have measured the length of primary and secondary node L-R addresses with increasing topology sizes (secondary address is an additional address chosen by a node for redundancy in case the primary landmark fails). As can be seen in Fig. 8, address lengths are short in practice and only grow with $\log n$ on emulated topologies. Note that the shown average can be lower than 1 as landmarks are considered to have a L-R address length of zero.

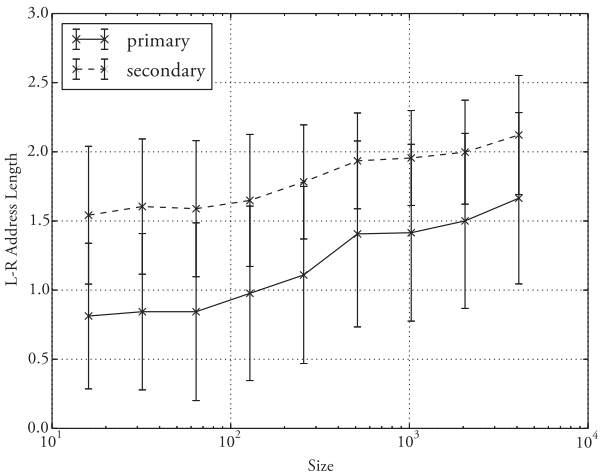


Fig. 8. Average length of primary and secondary L-R addresses with increasing topology sizes (note the logarithmic scale). Error bars show standard deviations of path address length.

5.2.3. Link congestion

Since the protocol uses landmarks to stitch together paths to distant nodes, we expected that some paths would become more congested than others. We compared the link congestion of paths selected by U-Sphere to link congestion encountered when using shortest paths instead. To measure link congestion in the emulated network realistically, the scenario instructed each node to keep track of the number of RPC *ping* packets traversing each of its links. Then we used the same test as when computing path stretch—each node routed a *ping* RPC to two randomly selected nodes and the destination node sent back a reply. After all measurements were complete we gathered the link congestion counters from all links and compared the values to a routing protocol that instead used only shortest paths between the same source-destination pairs.

As can be seen in Fig. 9, U-Sphere does indeed show increased link congestion for a small ($< 0.1\%$) percentage of links when compared to a shortest-path routing protocol. Values on the *x*-axis represent the number of times a link has been traversed and distribution is over all links. Some links are highly congested even in the case of a shortest-paths protocol because the *as-733-a* topology contains small amount

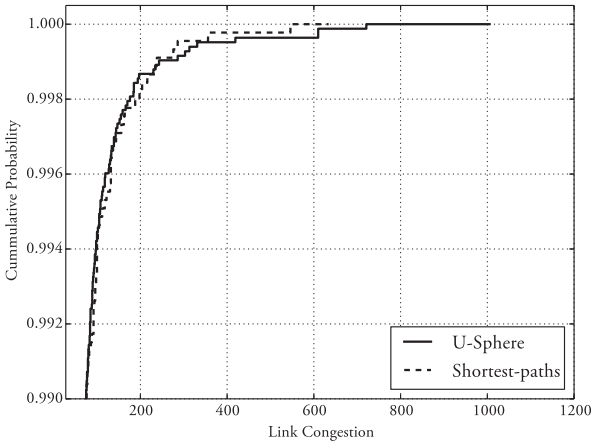


Fig. 9. Comparison of link congestion on topology *as-733-a* with 3015 nodes. The plot is zoomed into the region where the most difference between the two can be seen.

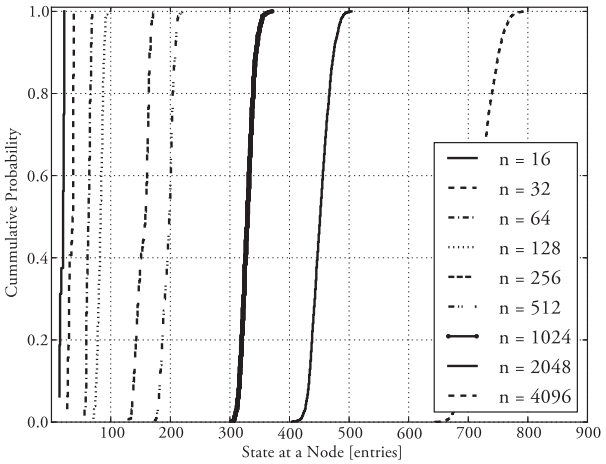


Fig. 11. Combined state distribution over nodes for varying topology sizes.

of bridges between communities which require repeated use of a few links.

5.2.4. RIB and NIB state

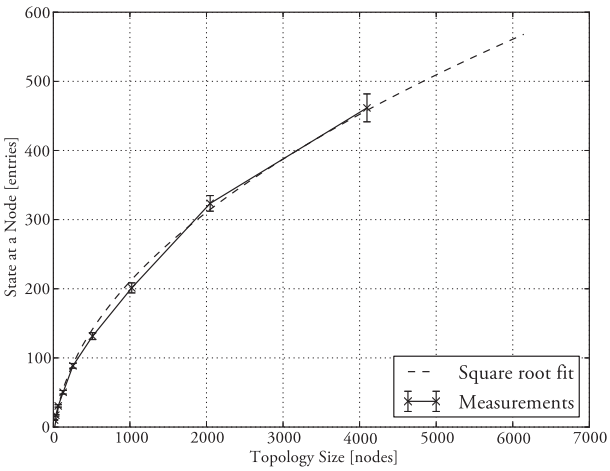
Next, we measured the amount of state held by U-Sphere nodes in their routing tables and name-to-locator resolution databases. Fig. 10 shows the growth of average routing state per node in the number of stored entries. Here we see that name-to-locator mapping state seems to increase only on every second measurement. This is due to the way nodes are partitioned into sloppy groups as the number of sloppy groups only increases after the size estimate doubles. The growth in state follows $\tilde{O}(\sqrt{n})$ and looking at combined state distribution (see Fig. 11) we can see that state is also evenly distributed among all the nodes with no apparent long tails that would indicate higher state on some nodes. The same state distributions have also been measured for *hyperboria* and both *as-733* topologies.

5.2.5. Message complexity

An important factor in the evaluation is also the U-Sphere's message complexity. We measure the amount of control messages required to establish and maintain all routing state through time. Routing announce period τ_r and name-to-locator mapping announce period τ_s greatly affect the measured message complexities. In all our experiments we use $\tau_r = 30\,\text{s}$ and $\tau_s = 600\text{ss}$.

Fig. 12 shows the number of updates (RT for *route updates*, SG for *sloppy group name-to-locator mapping updates* and SA for *security association updates*) that are transmitted per second per node on average. Note that this is not necessarily the number of messages because multiple records can be grouped in the same message in order to improve throughput and reduce the number of transmitted messages, which is why counting the update records is a better measure.

The initial rise in transmitted records corresponds to the node initialization phase where more and more nodes are started in the emulated network. After the initial phase, the number of records stabilizes as no new nodes are started and



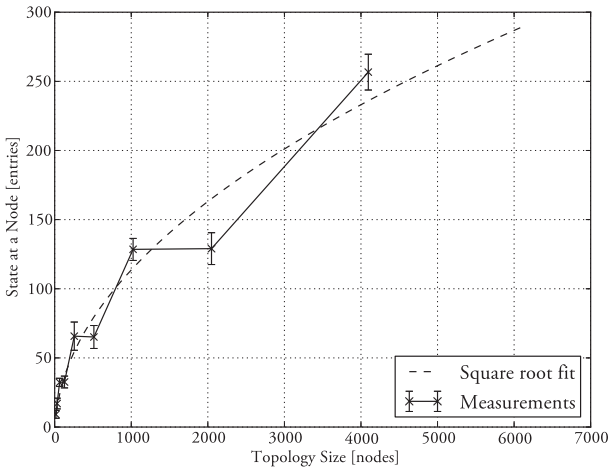


Fig. 10. Average routing table size (left) and name-to-locator resolution database size (right) at a node with increasing topology sizes (topology *synthetic-hk*). Dashed line represents a fit of $a\sqrt{x} + c$ over the measurements. Error bars show standard deviations of routing state size.

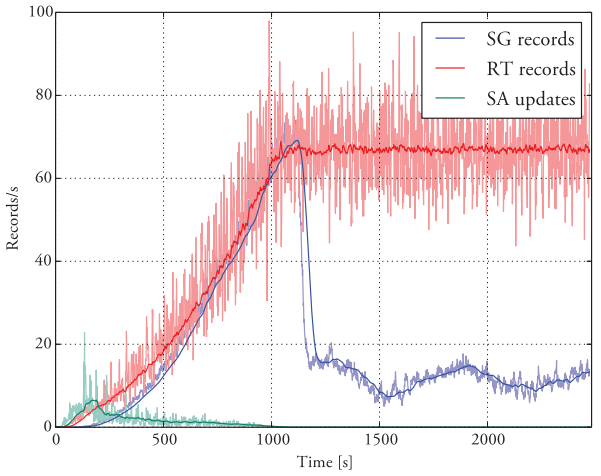


Fig. 12. Number of records transmitted per second per node (topology *synthetic-hk*, $n = 2048$). Shown are route updates (RT), name-to-locator mapping updates (SG) and security associations (SA).

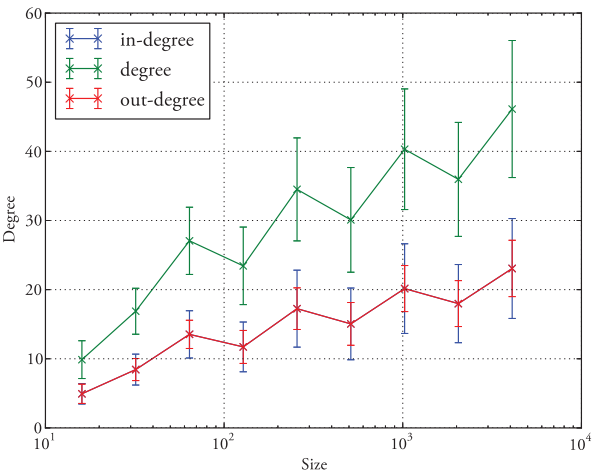


Fig. 14. Node degrees in the sloppy group overlay topology under varying number of nodes (note the logarithmic scale). Error bars show standard deviations of node degrees.

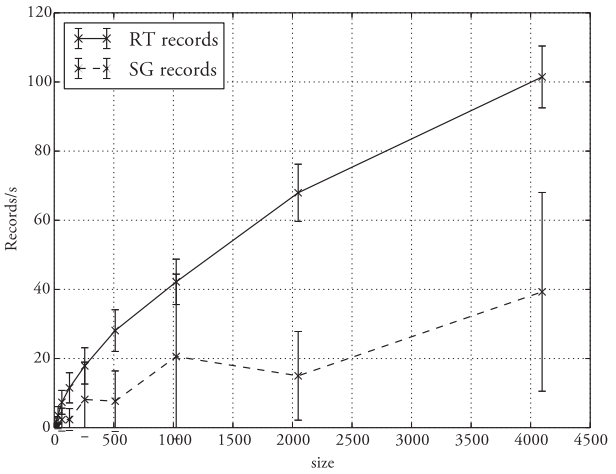


Fig. 13. Average records (route and name-to-locator mapping update) transmitted per second per node with increasing topology sizes. Averages are counted only after all the nodes have started. Error bars show standard deviations of record transmission rates.

the state converges. Initial overhead in sloppy group messages corresponds to the establishment of the sloppy group overlay. After the overlay is established, only periodic announces are enough for its maintenance, so the number of updates is reduced. Fig. 13 shows how the message complexity scales with increasing topology sizes. We have also experimented with varying the node degree in the social topology. As expected, the average message complexity for route updates grows linearly with average node degree—each neighbor has to periodically update all its links.

Another important factor in the scalability of the protocol is the degree of nodes in the sloppy group overlay topology. It is important because we do not want to create too many copies of the update messages, but at the same time we want to increase the probability of successful delivery. We measured the average in, out and combined degrees of nodes in the overlay topology. Results, seen on Fig. 14, show that the degrees grow with $\log n$ which ensures that the scheme scales well in practice.

5.2.6. Sybil attacks

The last scenarios include adversarial Sybil nodes that attempt to interfere with the normal operation of the routing protocol. We generate the Sybil topologies according to the threat model. First we use the already mentioned Holme–Kim model to generate a base topology. Then we randomly attach Sybil nodes to honest nodes in order to establish a predefined percentage of attack edges. Sybil nodes are also randomly interconnected among themselves. We generate multiple topologies for each percentage of attack edges and show the standard deviations. Multiple scenarios are used to evaluate the limits of Sybil-tolerance:

Scenario A Instructs all Sybil nodes to forward name-to-locator records only for other Sybil nodes. Records of honest nodes are simply dropped by a Sybil node.

Scenario B Same as in Scenario A, but additionally all Sybil nodes designate themselves as landmarks.

Scenario C In this scenario, Sybil nodes also interfere with data forwarding. All messages that do not originate from another Sybil node are dropped. We should note that U-Sphere does not protect data forwarding by itself.

For scenarios A and B, we measure the percentage of honest resolved node pairs. A resolved node pair (s, d) means that node s knows the L-R address of node d, given that both nodes are members of the same sloppy group. A value of 100% means that all honest nodes are able to resolve L-R addresses for all the other nodes in their sloppy group. In scenario C we measure the percentage of successful *ping* RPC calls where each node pings two randomly selected other nodes. We vary the percentage of attack edges so that up to 7% of all edges in the topology are attack edges. Fig. 15 shows that the protocol successfully defends against Sybil attackers interfering with name resolution even when the fraction of attack edges is high and even when all Sybil nodes designate themselves

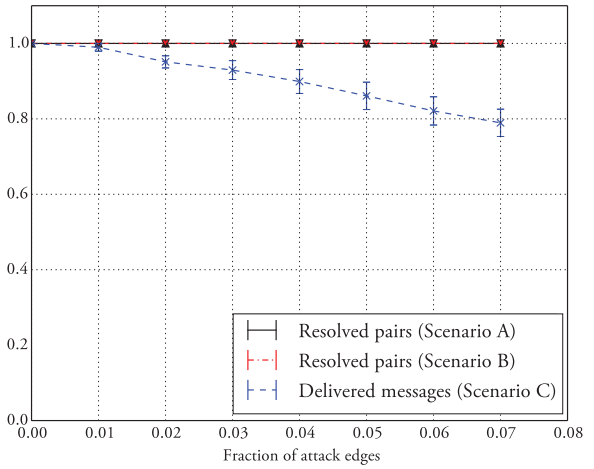


Fig. 15. Fractions of resolved pairs and delivered messages vs. the fraction of attack edges under different Sybil scenarios. Error bars show standard deviations.

Table 2
Related distributed routing protocols.

Protocol	Flat	Low stretch	Low state	Sybil-t.
CJDNS [18]	✓	–	✓	–
X-Vine [13]	✓	–	✓	Partial[a]
VRR [33]	✓	–	–	–
Disco [15]	✓	✓	✓	–
U-Sphere	✓	✓	✓	✓

[a] Sybil-tolerant under the assumption that the adversarial nodes' identifiers are randomly distributed.

as landmarks. As there are no specific defenses for ensuring deliverability of forwarded data messages, it can be seen that the protocol does much worse with data forwarding when percentage of attack edges increases.

6. Related work

In this section we survey related routing protocols and approaches from different fields of research and compare them with U-Sphere based on the protocol design and evaluation results. Table 2 shows an overview of compared protocols.

6.1. Compact routing

Compact routing protocol research dates back to the seminal work of Thorup and Zwick [34] who have first proposed an algorithm that can guarantee $O(1)$ (at most 3) path stretch with $\tilde{O}(\sqrt{n})$ per-node state for the location-dependent case. Abraham et al. [23,26] have later shown that very similar guarantees can also be obtained for the location-independent case. All of these designs, however, assume a static network topology and centralized routing table construction for all nodes participating as routers in the network. This is in contrast to a distributed routing protocol like U-Sphere, that one could use in practice where there is no such central point.

Motivated by supporting efficient routing on small embedded devices with heavily constrained resources (such as wireless sensors), Mao et al. [24] presented S4, a distributed compact routing protocol building on the theory of Thorup and Zwick. The S4 protocol aims to deliver bounded per-node routing state and bounded path stretch. It routes on location-dependent names, where the name-to-locator resolution is provided by a consistent hashing database (known as the *location directory*) over the set of landmark nodes. This resolution step can arbitrarily increase path stretch for the first packet of a flow and additionally it has been shown in [15] that S4 can sometimes violate the per-node state bound of $\tilde{O}(\sqrt{n})$ by a large margin.

Addressing some of S4's deficiencies, Singla et al. [15] have presented a protocol called Disco, that can route on location-independent node identifiers and at the same time guarantee constant path stretch and $\tilde{O}(\sqrt{n})$ bounded per-node routing state for arbitrary topologies. In addition to a landmark-based location directory a DHT-like ring overlay was introduced for the dissemination of name-to-locator mapping records in order to distribute the load away from landmark nodes. The location directory is still used to bootstrap and repair the overlay.

A different route is taken by Caesar et al. [33] with a protocol called VRR (Virtual Ring Routing). Each node has both physical neighbors (based on network topology) and virtual neighbors (based on DHT key identifiers). The routing protocol is a network-layer DHT where routing is performed greedily by looking up ever closer location-independent identifiers in key space. This protocol does not provide any guarantees regarding path stretch and routing state—evaluations in [14,15] have shown that it can experience high stretch in practice and that some nodes can also exhibit very high state.

In contrast to U-Sphere, all of the above location-independent protocols are vulnerable to Sybil attacks where an adversary is allowed to arbitrarily choose node identifiers and can introduce many adversarial nodes into the network. In S4 and Disco, the point of attack is the location directory—adversarial nodes can choose specific identifiers in such a way that the consistent hashing scheme will store specific name-to-locator mappings on them and can then proceed to censor them at will (see Fig. 16 for an overview of this attack). In Disco, an additional attack can be mounted on the DHT-like ring overlay, which again uses adversary-influenced node identifiers for its structure. Disco and S4 also do not protect the paths in route announces and they can thus be shortened, enabling the adversary to be seemingly placed on shortest paths and giving him the power to control more traffic. VRR, being based on a DHT construction, is vulnerable to routing table poisoning with specifically chosen node identifiers [12]. The reliance on node identifiers for structure in all these protocols means that such attacks can be mounted from any position in the network topology. In U-Sphere, an adversary is limited in influence to its trusted neighborhood.

6.2. Securing routing in DHTs

Whānau [11] is a general one-hop DHT protocol bundled with a social network based Sybil-detection scheme and a scheme for preventing identifier clustering attacks. Routing tables are built by nodes performing $\sqrt{e}$ short random walks (where e is the number of all edges) and sampling end nodes

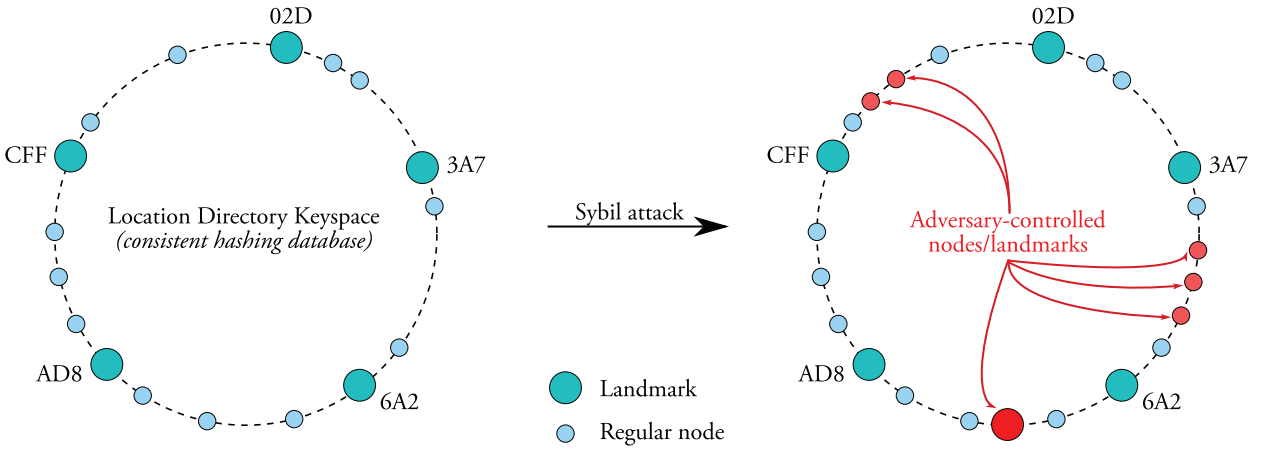


Fig. 16. Sybil attack on the location directory needed by S4 [24] for name-to-locator resolution and Disco [15] for sloppy group overlay bootstrapping. Since the adversary is able to control the m-bit prefixes of its nodes by doing $O(2^m)$ computation [12], it can get hold of almost arbitrary positions within the overlay. It can then use this fact to deny the querying nodes access to certain records, causing the name-to-locator resolution to fail for arbitrary identifiers.

to construct routing tables. It uses layered node identifiers in order to combat clustering attacks. The downside of the protocol is the requirement that all routing tables need to be recalculated once the network topology changes, making it unsuitable for dynamic networks.

R5N [35] is a DHT routing protocol that combines random walks with greedy recursive routing. Its design is based on the recursive version of the Kademlia protocol [36]. When routing, a message is forwarded randomly for the first few hops, before it switches to standard greedy DHT routing. This enables the protocol to de-correlate the message from the source node's vicinity and reduce the chance that the adversary is present on the routing path. Since it uses random routing for the first few hops, the protocol only delivers messages with certain probability and messages must thus be retried multiple times to ensure delivery. As such, there is no bound on path stretch.

X-Vine [13] is a DHT-based protocol for routing over social network topologies. Its security properties are based on limiting the amount of routes that can be established over a node's edges. It, however, does not guarantee any bounds on stretch and as authors specify in their paper, it also assumes that node identifiers (even those of the adversary) are randomly distributed in identifier space. When they are not, similarly to VRR, poisoning attacks can be mounted against the protocol in order to control specific paths.

6.3. General Sybil-detection protocols

There exists a number of protocols that aim to protect general decentralized systems from Sybil attacks. None of these schemes are routing protocols by themselves and therefore cannot be directly compared to U-Sphere or other routing protocols, but we include them here for completeness.

SybilGuard [9] and the improved SybilLimit [37] are among the first protocols to use social networks for Sybil defense. Each node first performs $\sqrt{e}$ (where e is the number of edges in the social topology) random walks of length $O(\log n)$ and remembers the last edge in the walk (the tail). Honest nodes then have intersecting tails with high probability and so any nodes that cannot find such an intersection are considered adversarial, and are not accepted into the network.

Gatekeeper [38] aims to improve the guarantees provided by SybilLimit on graphs that exhibit random expander properties. It accepts only $O(\log n)$ adversarial nodes per attack edge. The protocol is based on a ticket distribution scheme that uses multiple randomly selected nodes as ticket distribution sources in order to validate nodes. SybilInfer is a centralized protocol by Danezis and Mittal [10] which detects Sybil nodes by using Bayesian inference. It requires complete knowledge of the social topology and is thus not appropriate as a distributed protocol.

6.4. General attacks on routing protocols

In this paper we have focused our attention on strengthening scalable location-independent routing protocols against Sybil attacks which are common in real-world decentralized systems and social networks [39,40]. In addition to Sybil attacks, there exist several classes of routing attacks against decentralized routing schemes. We explore them in this section and show that most control-plane attacks are actually mitigated by mechanisms employed by U-Sphere.

Many of the routing attack studies in the literature originate from routing in mobile ad-hoc networks (MANETs) [41]. This is due to the fact that MANETs are fully decentralized in nature, without any central points of trust, and this makes them especially difficult to secure. While MANETs have some unique characteristics like a highly dynamic topology and severely resource-constrained devices, we can draw parallels between them and decentralized peer-to-peer communication networks that we address in this paper. In both, new nodes may attach themselves at any point in the topology and there is no central authority that would regulate admission into the network. Both require scalable protocols that can handle increasing numbers of nodes without consuming too many resources. In addition to MANETs, routing attacks are also important when talking about securing BGP,

the routing protocol used for managing the global Internet routing table [42].

The following general classes of attacks are the most commonly analyzed in regard to routing protocol security. For each of them, we describe state-of-the-art mitigations and compare them to the U-Sphere protocol.

Flooding attack In a flooding attack, an adversary attempts to prevent normal network operation by continuously generating control traffic that must then be processed by other routers, consuming their resources. State-of-the-art mitigations generally focus on identifying and rate limiting flood traffic, most successfully using statistical analysis [43]. It is also possible to perform such an attack on the U-Sphere network, which does not implement any kind of mitigations. But, as all the control messages are cryptographically signed and routing paths in messages are protected by the signed announce delegation chain mechanism, flooding nodes may be detected and their links revoked, requiring the adversary to gain trust of new users that he can exploit for propagating flood traffic. In order to detect flood traffic in the first place, the mentioned existing mitigation methods may easily be added on top of U-Sphere.

Blackhole attack In a blackhole attack, an adversary propagates false routing information and causes the traffic of legitimate nodes to be redirected to nodes under his control, where he can drop it. In a U-Sphere network, an adversary is unable to generate valid route announcements for non-existent links or modify existing announcements to make the paths appear shorter due to the signed announce delegation chain mechanism. The adversary is still able to drop control and data traffic that passes through his nodes and so the blackhole attack cannot be prevented in cases where the adversary is already on the shortest path between the source and destination.

Link withholding attack In a link withholding attack, an adversary fails to propagate routing control traffic. In U-Sphere, this attack is possible, but in case alternate paths exist, it is not particularly disruptive as data traffic will be routed around such link-withholding nodes.

Link spoofing and path truncation attacks In a link spoofing attack, an adversary attempts to advertise fake (non-existent) links with honest nodes in order to redirect traffic. S-BGP [42] and SPV [44] present solutions for these attacks in case of the global Internet routing table with an existing public-key infrastructure (PKI). In U-Sphere, we focus on a fully decentralized solution without any PKI in place. A U-Sphere node cannot spoof a link or truncate the path without access to the neighbor's private key as all the routing announces are cryptographically signed by the originator and paths in route updates are protected by signed announce delegation chains.

Replay attack In a replay attack, an adversary propagates old but otherwise valid control messages in order to create routing issues as the information is no longer correct. This attack is prevented by U-Sphere through the use of cryptographically signed timestamps in the control messages.

Wormhole attack In a wormhole attack, a pair of colluding adversarial nodes use an out-of-band network to capture valid control messages at one location in the network and reply them in another location, basically creating a virtual network link which may now be the shortest path. Solutions for MANETs require either tightly synchronized clocks or access to GPS locations [45], both of which are not feasible in a network like U-Sphere where various transports may be used. This is one of the hardest attacks to prevent as the adversary actually creates a shorter path by using an out-of-band network. As such, U-Sphere is also unable to prevent this attack.

As we have shown in this section, some of the mitigation strategies used for securing MANETs and the global Internet routing table may also be reused in U-Sphere, while others are unnecessary due to the protocol's existing security features.

6.5. Limitations of U-Sphere

As any practical protocol, the U-Sphere routing protocol is not a silver bullet. In its current form, the protocol requires online presence of user nodes. This is suitable for use as a network routing protocol, similar to CJDNS or X-Vine, but it can be an issue when building decentralized social network services where not all users can be online all the time. Concepts from delay/disruption tolerant networks could be used to enable delivery of messages in such scenarios. Targeted attacks, where the adversary targets a single user or her friends, cannot be prevented by U-Sphere. This is due to the fact that the protocol relies on having trusted friends as one-hop links. Also as shown, the protocol cannot resist attacks where the adversary is located on the shortest path between two nodes and then selectively drops data traffic. Handling such cases requires some sort of fault detection mechanism. Using redundant paths combined with techniques from reputation management and data-plane fault localization could be possible as U-Sphere supports some path flexibility by routing via different landmarks.

7. Conclusion

In this paper we have proposed U-Sphere, a novel location-independent routing protocol that is tolerant of Sybil adversaries and is scalable to large topologies due to compact routing state and low, constant bounded, path stretch. In contrast to other state-of-the-art solutions, our contributions therefore address all of the goals outlined in the introduction, namely (a) location-independent routing is achieved by using a fully distributed overlay that can bootstrap itself without the need for landmark-based databases, making it robust against Sybil attacks; (b) scalability in the form of low message overhead, low per-node state, bounded by $\tilde{O}(\sqrt{n})$, is achieved together with $O(1)$ path stretch; and (c) privacy is ensured by not disclosing transport addresses to any non-trusted nodes and by using non-unique identifiers and rotating public keys for identifying links between nodes.

We have performed emulation-based experiments on different network topologies, including real datasets of existing networks. All of these experiments confirmed that the above goals have been successfully met.

Acknowledgments

The authors have been supported by the following institutions: Jernej Kos by the Slovenian Research Agency (grant 1000-11-310153) and the WiNeMo COST Action (STSM grant IC0906-160913-035608-35608); and Denis Trček by the Slovenian Research Agency (research program Pervasive computing P2-0359).

Appendix A. Proofs of stretch and state bounds

In this section we provide proofs for our stated path stretch and state bounds. Path stretch is the ratio between the length of paths taken by U-Sphere and the length of the shortest possible path in the network topology. We measure state in the number of entries in the routing and name resolution tables. These proofs do not assume any adversarial attacks.

Theorem 1 (Path Stretch Bound)**.** *After routing state convergence, U-Sphere routes the first message with path stretch* ≤ 7.

Proof. There are several scenarios that can result in lower path stretch—if the source s knows a destination d's L-R address, if d is a landmark node, if $d \in V_s$ or in case of shortcutting. In these cases it is easy to see that the stretch will be lower, so we omit these scenarios and focus instead on the last scenario where name-resolution via a relay sloppy group member v in the extended vicinity is needed and therefore $d \notin V_s$ and d is not a landmark node. In this case, the full path required will be $s \rightsquigarrow v \rightsquigarrow \rightsquigarrow \ell_d \rightsquigarrow d$ (see Fig. 4) where $v \in S_d$ and by construction of extended vicinities $v \in V_s$. ℓ_d is the landmark closest to d and each intermediate path segment $\rightsquigarrow$ is the shortest path learned via the path-vector protocol. Let $d(a, b)$ represent the shortest distance metric between nodes a and b in the network topology. To aid with the proof we first provide the following lemma.

Lemma 1.1. *Shortest distance between* ℓ_d *and* d *is at most twice the distance between* s *and* d*:* $d(\ell_d, d) \leq 2d(s, d)$.

Proof.

$$\begin{aligned} d(\ell_d, d) &= d(d, \ell_d) && \text{(undirected graph)} \\ &\leq d(d, \ell_s) && (\ell_d \text{is} d'\text{s(closest landmark)} \\ &\leq d(d, s) + d(s, \ell_s) && \text{(triangle inequality)} \\ &= d(s, d) + d(s, \ell_s) && \text{(undirected graph)} \\ &\leq d(s, d) + d(s, d) && (\ell_s \in V_s \wedge d \notin V_s) \\ &= 2d(s, d). \end{aligned}$$

□

Now we can derive the upper bound for each segment of the full path a message must traverse. For the first segment $s \rightsquigarrow v$, the following holds true:

$$d(s, v) \leq d(s, d) \quad (v \in V_s \wedge d \notin V_s)$$

For the second segment $v \rightsquigarrow \ell_d$, the following holds true:

$$\begin{aligned} d(v, \ell_d) &\leq d(v, s) + d(s, d) + d(d, \ell_d) && \text{(triangle inequality)} \\ &\leq d(s, v) + d(s, d) + d(\ell_d, d) && \text{(undirected graph)} \\ &\leq d(s, d) + d(s, d) + d(\ell_d, d) && (v \in V_s \wedge d \notin V_s) \\ &\leq d(s, d) + d(s, d) + 2d(s, d) && \textit{Lemma}1.1 \\ &= 4d(s, d). \end{aligned}$$

For the third segment $\ell_d \rightsquigarrow d$, the following statement holds true by simple application of Lemma 1.1:

$$d(\ell_d, d) \leq 2d(s, d).$$

Now, by combining all three path segments into the final path, we see that its length is $\leq 7d(s, d)$. This result shows an upper bound on path stretch of 7. □

Theorem 2 (State Bound)**.** *After routing state convergence and with high probability, each U-Sphere node must maintain* $O(\sqrt{n \log n})$ *entries in its routing and name resolution tables.*

Proof. Each node must maintain the following state in its routing tables via the path-vector protocol: (a) route entries for each of the landmark nodes; and (b) route entries for its extended vicinity. The protocol has each node become a landmark independently with probability $\sqrt{(\log n)/n}$ (based on a node's size estimate n). There are therefore $O(\sqrt{n \log n})$ landmark nodes in expectation and, by a Chernoff bound, with high probability. For the extended vicinity, each node accepts $O(\sqrt{n \log n})$ route updates from its neighbors.

In addition to routing tables, each node must also maintain name resolution state that is required for resolving node identifiers into L-R addresses: (a) node identifier to L-R address mappings for all members of its own sloppy group; (b) links to near members of its own sloppy group; and (c) back-links to members of its own sloppy group. By construction, each node assigns itself into a sloppy group by examining the first $\lfloor \log_2 (\sqrt{n/\log n}) \rfloor$ bits of its node identifier. This results in each sloppy group having $O(\sqrt{n \log n})$ nodes in expectation, and again by a Chernoff bound, with high probability. Therefore, each node must maintain mappings for $O(\sqrt{n \log n})$ other nodes.

Overlay construction requires each node to maintain links to $O(\log n)$ members of its own sloppy group and $O(\log^2 n)$ back-links to members of its own sloppy group. Forward sloppy group link state is shared with the extended vicinity route entries stored in the routing table.

Finally, each node with degree δ must also maintain $O(\delta)$ security associations with its peers. Although, in the worst case, $O(\delta)$ could be larger than $O(\sqrt{n \log n})$, the node only needs to store security associations for the peers that are part of active routes in the routing table—and there are only $O(\sqrt{n \log n})$ active routes at any one time.

Summing up all the different amounts of state needed by different parts of the protocol, we see that U-Sphere requires $O(\sqrt{n \log n})$ per-node state. □

References

[1] I. Baumgart, F. Hartmann, Towards secure user-centric networking: service-oriented and decentralized social networks, in: 2011 Fifth IEEE Conference on Self-adaptive and Self-organizing Systems Workshops (SASOW), 2011, pp. 3–8.

[2] L. Cutillo, R. Molva, T. Strufe, Safebook: a privacy-preserving online social network leveraging on real-life trust, IEEE Commun. Mag. 47 (12) (2009) 94–101. ISSN 0163-6804.
[3] C. Grothoff, B. Polot, C. von Loesch, The internet is broken: idealistic ideas for building a GNU network, in: W3C/IAB Workshop on Strengthening the Internet Against Pervasive Monitoring (STRINT), W3C/IAB, W3C/IAB, London, UK, 2014.
[4] B.C. Popescu, B. Crispo, A.S. Tanenbaum, Safe and private data sharing with turtle: friends team-up and beat the system, in: Proceedings of the 12th International Conference on Security Protocols, SP'04, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 213–220. ISBN 3-540-40925-4, 978-3-540-40925-0.
[5] B.A. Ford, UIA: a global connectivity architecture for mobile personal devices, Massachusetts Institute of Technology, 2008 (Ph.D. thesis).
[6] D. Clark, The design philosophy of the DARPA Internet protocols, in: Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM '88, ACM, New York, NY, USA, 1988, pp. 106–114. ISBN 0-89791-279-9.
[7] J.R. Douceur, The Sybil attack, in: P. Druschel, F. Kaashoek, A. Rowstron (Eds.), Peer-to-peer Systems, Lecture Notes in Computer Science, 2429, Springer/Berlin/Heidelberg, 2002, pp. 251–260. ISBN 978-3-540-44179-3.
[8] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, A. Panconesi, SoK: the evolution of Sybil defense via social networks, in: 2013 IEEE Symposium on Security and Privacy (SP), 2013, pp. 382–396. ISSN 1081-6011.
[9] H. Yu, M. Kaminsky, P.B. Gibbons, A.D. Flaxman, SybilGuard: defending against Sybil attacks via social networks, IEEE/ACM Trans. Netw. 16 (3) (2008) 576–589.
[10] G. Danezis, P. Mittal, SybilInfer: detecting Sybil nodes using social networks, in: NDSS, 2009.
[11] C. Lesniewski-Laas, M.F. Kaashoek, Whanau: a Sybil-proof distributed hash table, in: Proceedings of the Seventh USENIX Conference on Networked Systems Design and Implementation, NSDI'10, USENIX Association, Berkeley, CA, USA, 2010, p. 8.
[12] G. Urdaneta, G. Pierre, M.V. Steen, A survey of DHT security techniques, ACM Comput. Surv. 43 (2) (2011) 8:1–8:49. ISSN 0360-0300.
[13] P. Mittal, M. Caesar, N. Borisov, X-Vine: secure and pseudonymous routing using social networks, in: Proceedings of NDSS 2012, 2012.
[14] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, ROFL: routing on flat labels, SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 363–374. ISSN 0146-4833.
[15] A. Singla, P.B. Godfrey, K. Fall, G. Iannaccone, S. Ratnasamy, Scalable Routing on Flat Names, Association for Computing Machinery, p. 1. ISBN 9781450304481.
[16] I. Clarke, O. Sandberg, B. Wiley, T. Hong, Freenet: a distributed anonymous information storage and retrieval system, in: H. Federrath (Ed.), Designing Privacy Enhancing Technologies, Lecture Notes in Computer Science, 2009, Springer, Berlin/Heidelberg, 2001, pp. 46–66. ISBN 978-3-540-41724-8.
[17] N. Evans, C. GauthierDickey, C. Grothoff, Routing in the dark: pitch black, in: Twenty-third Annual Computer Security Applications Conference, 2007. ACSAC 2007, 2007, pp. 305–314. ISSN 1063–9527.
[18] C.J. Delisle, CJDNS. URL: https://github.com/cjdelisle/cjdns, 2014.
[19] P.R. Zimmermann, The Official PGP User's Guide, MIT Press, Cambridge, MA, USA, 1995. ISBN 0-262-74017-6.
[20] N. Evans, B. Polot, C. Grothoff, Efficient and Secure Decentralized Network Size Estimation, Springer-Verlag, 2012.
[21] D. Bernstein, Curve25519: new Diffie–Hellman speed records, in: M. Yung, Y. Dodis, A. Kiayias, T. Malkin (Eds.), Public Key Cryptography—PKC 2006, Lecture Notes in Computer Science, 3958, Springer, Berlin/Heidelberg, 2006, pp. 207–228. ISBN 978-3-540-33851-2.
[22] D. Bernstein, N. Duif, T. Lange, P. Schwabe, B.-Y. Yang, High-speed high-security signatures, J. Cryptogr. Eng. 2 (2) (2012) 77–89. ISSN 2190-8508.
[23] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, M. Thorup, Compact name-independent routing with minimum stretch, ACM Trans. Algorithms 4 (3) (2008) 37:1–37:12. ISSN 1549-6325.
[24] Y. Mao, F. Wang, L. Qiu, S. Lam, J. Smith, S4: small state and small stretch compact routing protocol for large static wireless networks, IEEE/ACM Trans. Netw. 18 (3) (2010) 761–774.
[25] P.B. Godfrey, I. Ganichev, S. Shenker, I. Stoica, Pathlet routing, SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 111–122. ISSN 0146-4833.
[26] I. Abraham, C. Gavoille, D. Malkhi, Routing with improved communication-space trade-off, in: R. Guerraoui (Ed.), Distributed Computing, Lecture Notes in Computer Science, 3274, Springer, Berlin/Heidelberg, 2004, pp. 305–319. ISBN 978-3-540-23306-0.
[27] N.S. Evans, C. Grothoff, Beyond simulation: large-scale distributed emulation of p2p protocols, in: Proceedings of the Fourth Conference on Cyber Security Experimentation and Test, CSET'11, USENIX Association, Berkeley, CA, USA, 2011, p. 4.
[28] D.J. Bernstein, Curvecp: usable security for the internet. URL: http://curvecp.org, 2011.
[29] J. Kos, U-sphere implementation and testbed. URL: https://github.com/kostko/unisphere, 2014.
[30] A.-L. Barabasi, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[31] P. Holme, B.J. Kim, Growing scale-free networks with tunable clustering, Phys. Rev. E 65 (2002) 026107.
[32] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD '05, ACM, New York, NY, USA, 2005, pp. 177–187. ISBN 1-59593-135-X.
[33] M. Caesar, M. Castro, E.B. Nightingale, G. O'Shea, A. Rowstron, Virtual ring routing: network routing inspired by DHTs, SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 351–362. ISSN 0146-4833.
[34] M. Thorup, U. Zwick, Compact routing schemes, in: Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '01, ACM, New York, NY, USA, 2001, pp. 1–10. ISBN 1-58113-409-6.
[35] N.S. Evans, C. Grothoff, R5N: Randomized Recursive Routing for Restricted-route Networks, Institute of Electrical and Electronics Engineers, pp. 316–321. ISBN 978-1-4577-0458-1.
[36] B. Heep, R/Kademlia: recursive and topology-aware overlay routing, in: 2010 Australasian Telecommunication Networks and Applications Conference (ATNAC), 2010, pp. 102–107.
[37] H. Yu, P.B. Gibbons, M. Kaminsky, F. Xiao, SybilLimit: a near-optimal social network defense against Sybil attacks, IEEE/ACM Trans. Netw. 18 (3) (2010) 885–898.
[38] N. Tran, J. Li, L. Subramanian, S.S. Chow, Optimal Sybil-resilient Node Admission Control, Institute of Electrical and Electronics Engineers, pp. 3218–3226. ISBN 978-1-4244-9919-9.
[39] L. Wang, J. Kangasharju, Real-world Sybil attacks in BitTorrent mainline DHT, in: 2012 IEEE Global Communications Conference (GLOBECOM), 2012, pp. 826–832. ISSN 1930-529X.
[40] Z. Yang, C. Wilson, X. Wang, T. Gao, B.Y. Zhao, Y. Dai, Uncovering social network Sybils in the wild, ACM Trans. Knowl. Discov. Data 8 (1) (2014) 2:1–2:29. ISSN 1556-4681.
[41] B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, A. Jamalipour, A survey of routing attacks in mobile ad hoc networks, IEEE Wireless Commun. 14 (5) (2007) 85–91. ISSN 1536-1284.
[42] S. Kent, C. Lynn, K. Seo, Secure border gateway protocol (S-BGP), IEEE J. Selected Areas Commun. 18 (4) (2000) 582–592. ISSN 0733-8716.
[43] S. Desilva, R. Boppana, Mitigating malicious control packet floods in ad hoc networks, in: 2005 IEEE Wireless Communications and Networking Conference, 4, 2005, pp. 2112–2117. ISSN 1525-3511.
[44] Y.-C. Hu, A. Perrig, M. Sirbu, SPV: secure path vector routing for securing BGP, SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 179–192. ISSN 0146-4833.
[45] Y.-C. Hu, A. Perrig, D. Johnson, Wormhole attacks in wireless networks, IEEE J. Selected Areas Commun. 24 (2) (2006) 370–380. ISSN 0733-8716.

Jernej Kos is a computer science researcher, software developer and network engineer with over 10 years of experience. He enjoys working on interesting projects, specifically with backend architecture and low-level details. He has experience with scalable web application development, development of software for embedded devices, routing protocol internals and security protocols. He is currently involved the open source project wlan slovenija, where he has developed a modular platform for configuration, provisioning and network monitoring of large-scale wireless mesh networks, an efficient VPN solution based on L2TPv3 support in the Linux kernel and a big data time series processing library. His current research interests include secure, Sybil-tolerant and privacy-aware decentralized services and novel location-independent compact routing protocols. To this end he has also developed Boost.ASIO C++ bindings for the CuveCP protocol and in the course of his Ph.D. thesis is working on a secure and scalable location-independent routing protocol that can be used in mesh networks and/or for building decentralized social networks. Together with other members of the Laboratory for e-Media he is currently involved with EU project SALUS which aims to deliver a secure PPDR network infrastructure based on LTE and IEEE802.11 technologies. He has successfully applied for and received grants from the Shuttleworth Foundation and the NLnet Foundation.

Mahdi Aiash received his B.Eng. degree in Computer Engineering from Aleppo University, Syria in 2004 and a master degree (M.Sc.) from Middlesex University, London, UK in 2008. Recently, he finished his Ph.D. at the same University. His main research area is in information security and networking.

Jonathan Loo received his M.Sc. degree in Electronics (with distinction) from the University of Hertfordshire, UK in 1998 and his Ph.D. degree in Electronics and Communications from the same university in 2003. Currently, he is a reader (associate professor) at the School of Engineering and Information Sciences, Middlesex University, UK. He leads a research team in the area of communication and networking. His research interest includes network architecture, communication protocols, network security, embedded systems, video coding and transmission, wireless communications, digital signal processing, and optical networks. He has successfully graduated 11 Ph.D.s as director of studies in the aforementioned specialist areas.

Denis Trček is with Faculty of Computer and Information Science, University of Ljubljana, where he heads Laboratory of e-Media. He has been involved in the field of computer networks and IS security and privacy for over 20 years. He has taken part in various EU and national projects in government, banking and insurance sectors (projects under his supervision totaled to approx. one million EUR). His bibliography includes over 100 titles, including monograph published by renowned publisher Springer. He has served (and still serves) as a member of various international bodies and boards (MB of the European Network and Information Security Agency, etc.). His interests include security, trust management, privacy and human factor modeling. He is a member of the IEEE.