A New Approach to Name-Based Link-State Routing for Information-Centric Networks

Ehsan Hemmati[1] and J.J. Garcia-Luna-Aceves[1,2]
[1]Department of Computer Engineering, University of California, Santa Cruz, CA 95064
[2]Palo Alto Research Center, Palo Alto, CA 94304
ehsan@soe.ucsc.edu, jj@soe.ucsc.edu

ABSTRACT

The Link State Content Routing (LSCR) protocol is presented, which supports routing over multiple paths to named content using link-state information. LSCR uses two types of link-state advertisements (LSAs): a *Router LSA* that contains information about links connected to each router, and an *Anchor LSA* that carries information regarding a name prefix and the router that advertises being attached to that name prefix, also called an anchor of the prefix. *Anchor LSA*s are propagated selectively based on a diffusing mechanism. In contrast to prior content routing solutions based on link-state information, LSCR allows routers to establish multiple routes to name prefixes, without requiring each router to know about all the instantiations of each prefix. LSCR is shown to avoid permanent routing loops and to have better performance compared to traditional link-state routing protocols when a name prefix is replicated at multiple sites in the network.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing protocols;
C.2.6 [**Internetworking**]: Routers

General Terms

Theory, Design, Performance

Keywords

Information-centric networks; name-based content routing; link-state routing

1. INTRODUCTION

The current Internet architecture was designed many years ago to address the communication needs prevailing at that time, and focused on the need to share limited, expensive, and static computer resources. Since then, the Internet usage pattern has shifted from a host-centric model to a flexible content-oriented model in which users and content are distributed and mobile.

ICN'15, September 30–October 2, 2015, San Francisco, CA, USA.
© 2015 ACM. ISBN 978-1-4503-3855-4/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810156.2810173.

A number of Information-Centric Network (ICN) architectures have been developed to address the increasing demand of user-generated content [9,13,19,27] in the Internet. ICN architectures are based on location-independent content naming rather than location-oriented host addressing. ICN architectures are implemented based on name resolution and name-based content routing to provide cost-efficient, mobile, and scalable content distribution.

The most promising ICN architectures can be characterized as Interest-based. Content Centric Networking (CCN) [1] and Named Data Networking [3] are the most prominent Interest-based ICN architectures today. In such architectures, content providers, i.e., producers, create named data objects (NDOs), and publish name prefixes associated with these content objects. A name prefix (or simply a prefix), is a location-independent routable name that is associated with a set of NDOs and advertised in the network by the publisher. A content consumer asks for an NDO or name prefix by sending a request, called an *Interest*, which is routed along content routers toward a publisher of the requested content. The producers or any caching sites that have a copy of the requested NDOs satisfy the Interests and send back the corresponding *Data* packets.

Each content router in the network forwards Interests according to its *Forwarding Information Base* (FIB), which stores the next hops towards name prefixes. A routing protocol is needed to maintain the routes needed to populate the FIB entries. An important feature of many such architectures is the caching of named data objects that can be done everywhere and for everything in the network. Designing a reliable routing algorithm that cope with adding and deleting prefixes in caches is challenging task.

Section 2 reviews prior work on name-based content routing for ICNs. As this summary reveals, no prior work has been reported based on link-state information that enables routers to know only subsets of the replicas of name prefixes to support name-based content routing to the nearest prefix replicas.

Section 3 presents **LSCR** (*link state content routing*), a name-based content routing protocol that uses link-state information and information regarding the nearest replicas of prefixes. LSCR relies on router names instead of IP addresses. Like other link-state routing protocols for ICNs, LSCR uses a flooding mechanism to propagate link-state information regarding physical link characteristics and build a map of the network topology at each router. However, instead of flooding publisher information, as it is done in NLSR [22] and OSPFN [23], LSCR propagates publisher information by diffusing the information selectively, based on a distributed computation of preferred publishers. LSCR supports both flat and hierarchical naming schemas; therefore, it can be implemented in different ICN architectures with different naming policies.

Section 4 shows that LSCR computes multiple paths to name prefixes and prevents permanent routing-table loops. A routing table lists the shortest distances to the nearest publishers of NDOs or name prefixes through one or multiple neighbors. Section 5 analyzes the communication, storage, and time complexities of LSCR, traditional link-state routing, and loop-free distance-vector routing.

Section 6 presents the results of simulation experiments comparing LSCR with a link-state approach similar to NLSR and OSPFN [22, 23]. LSCR produces less communication and computation overhead when the number of replicas in the network is more than three. This result follows from the fact that LSCR disseminates only partial publisher information.

2. RELATED WORK

Several ICN architectures have been proposed to handle name resolution and routing [7, 31]. In general, these architectures adopt content routing approaches that adapt traditional routing algorithms designed for networks in which a destination has a single instance [15]. This results in the need to disseminate information about all replicas of prefixes, rely on flooding requests to the entire network, build a spanning tree, use source routing, or use directories and DHTs for redirection. The rest of this section summarizes examples of these various approaches.

Directed Diffusion [24] was one of the first proposals for name-based routing of content. Requests for named content (called interests) are flooded throughout a sensor network, and data matching the interests are sent back to the issuers of interests. DIRECT [30] is similar to directed diffusion and provides named-based content routing in MANETs subject to connectivity disruption.

The name-based routing protocol (NBRP) [20] was one of the earliest proposals for name-based routing of content. Name-prefix reachability is advertised among content routers, and path information is used to avoid permanent loops.

CBCB (combined broadcast and content based) routing is another early development on name-based routing of content [12]. CBCB consists of two components. A spanning tree of the network or multiple per-source trees are established, and publish-subscribe requests for content are sent between consumers and producers of content over the tree(s) established in the network.

Data Oriented Network Architecture (DONA) [26] uses name resolution to map the flat names to corresponding IP addresses that can be local or global.

MobilityFirst [4] relies on an external and fast name resolution system called Global Name Resolution Service (GNRS) to map the data object names to network addresses. Name based routing in both DONA and MobilityFirst is accomplished using traditional IP routing and forwarding.

In Publish Subscribe Internet Technology (PURSUIT) [2, 8], each data object has a unique name that is mapped to the publisher. A Topology Manager (TM) in the network, that runs a link state routing protocol to discover the network topology, is responsible to calculate a route between the publisher and the consumer.

NDN [3] has adopted a traditional link-state approach to name-based content routing. NLSR [22] and OSPFN [23] are the two link-state content routing protocols that have been proposed to date. In NLSR, two types of link state advertisements (LSAs), *AdjacencyLSA* and *PrefixLSA*, propagate topology and publisher information in the network. Each router uses topology information and runs an extension of Dijkstra's shortest-path first (SPF) algorithm to calculate the next hops for each router. The router then maps a prefix to the name of its publisher and creates a routing table for each name prefix. NLSR propagates information regarding all replicas of all prefixes in the network and does not provide any mechanism to rank replicas of the same prefix.

A number of content routing approaches have been developed in the context of content delivery networks (CDN) that direct consumers to the nearest CDN sites and support name-based content routing in the CDN using distance information [18, 29]. Other approaches use distributed hash tables (DHT) as the name resolution tool running on top of the existing inter-domain routing infrastructure [5, 6, 25].

CORD [16] combines redirection of content requests with content routing using a publish-subscribe approach based on a virtual DHT of directories listing the location of content.

Recently, DCR [14] was introduced to support name-based content routing using distance information regarding the nearest instances of NDOs or name prefixes. The main difference between DCR and prior approaches to content routing is that routers compute shortest paths to prefixes or NDOs by disseminating distance information only for the nearest instance of a prefix. As the next section describes, LSCR adopts the same general approach using link-state information.

3. LSCR

We make a number of assumption in the description of LSCR. Routers are assumed to operate and store information correctly. Each router receives LSAs from its neighbors correctly and processes them one at a time within a finite time. Link costs can vary in time but cost values are always positive. The link cost assignment and metric determination mechanisms are beyond the scope of this paper.

Each router in the network has a unique name or identifier, which can be flat or hierarchical and a lexicographic value is assigned to the name.

Every piece of content in the network is a named-data object (NDO), and a set of one or multiple NDOs can be represented by name prefix (or simply $prefix$). Prefixes can be simple and human-readable or more complicated and self certifying, or may even be a cryptographic hash of the content.

A router that has local access to all the content associated with a name prefix is called an *anchor* of the prefix. Each anchor of a prefix advertises the prefix as being locally available to the rest of the network. Routers and content can be assigned flat or hierarchical names.

LSCR relies on two basic mechanisms: name resolution and topology-based routing. Like other link-state routing protocols [11], LSCR propagates link state advertisements (LSAs) to create a local copy of the network topology and a mapping schema from name prefixes to router identifiers (ID) at each router. Based on topology and anchor information, LSCR creates a lexicographic ordering among neighbors and calculates multiple routes to the nearest replicas of prefixes, i.e., to the nearest anchors of prefixes.

3.1 Messages and Data Structures

A link between router i and its neighbor n is denoted by (i, n) and its cost is denoted by l^i_n. The set containing router i and its neighbor routers is denoted by N^i. The lexicographic value of the identifier of a neighbor router n is denoted by $|n|$.

LSCR propagate two types of LSAs. A *Router LSA* (RLSA) is used to advertise the presence of a router and the state and cost of its outgoing links. An *Anchor LSA* (ALSA) is used to advertise the existence of name prefixes locally available, which make the advertising router an anchor of the prefixes. A sequence number is associated to each LSA to identify the message and its order.

An RLSA can be initiated by any router that runs LSCR. The ALSA sent by router i is denoted by $RLSA^i$ and consists of the name of the router i, a message sequence number (msn^i), and a list of outgoing links connected to the router i and the cost associated with each link.

ALSAs can be initiated only by anchors of the prefixs and intermediate routers can forward, drop, or hold these LSAs. The ALSA sent by anchor m regarding prefix j is denoted by $ALSA^m_j$, and consists of the name of the anchor (m); and one "$Prefix\ Update$" (PU^m).

PU^m states the prefix name j, the sequence number that is assigned to the prefix by the anchor (usn^m_j), and the "ValidFlag" or *vFlag* indicating if name prefix j is attached to anchor m or detached (vf^m_j).

Anchor m sends just one prefix update per ALSA because of two considerations. First, prefixes have different lengths and can be too large for several prefixes to fit in a single message. Second, every intermediate router that runs LSCR processes a received ALSA based on the prefix and decides whether to forward or hold the ALSA. A router may forward one ALSA with a specific prefix and hold another ALSA that advertises a different prefix from the same anchor.

Router i maintains a Link Cost Table (LT^i) storing the cost of the link from router i to each of its adjacent routers. Each LSCR router exchanges periodic $Hello$ messages with its neighbors to detect the addition or deletion of links and routers, as well as any changes in the link cost.

A predefined parameter defines the time interval between the transmission of two consecutive Hello messages. The link to a neighbor is considered to be down if a router does not receive a Hello message for a specified amount of time (a time-out) from that neighbor. Afterward, the router sends recovery Hello messages to detect the recovery with time intervals relatively longer than normal Hello message interval. A router cannot differentiate between a link being down or the neighbor behind that link having failed. However, this distinction does not affect the protocol, because in both cases the neighbor behind a perceived link failure should not be used to forward traffic. Each router that runs LSCR sends an RLSA at startup and whenever it detects a change in one of its links.

Each LSCR router maintains a $Forwarding\ Table$ (FT^i), storing the set of valid next hops for each destination. The row for destination p in FT^i specifies: the name of the router (p); the sequence number $(rsn(p))$ reported by router p; the Distance List (RD^i_p) consisting of the set of shortest distances from each neighbor router $n \in N^i$ to destination p (rd^i_{pn}); the shortest distance to router p (rd^i_p); and the set of neighbors that are valid next hops toward destination p $(RS^i_p\ )$.

Router i updates FT^i based on RLSAs received from other routers in the network. The RLSA form router k received by router i is denoted by $RLSA^i_k$. The information stored in $RLSA^i_k$ is the router sequence number rsn^i_j, plus the cost of each the links between router j and its neighbors.

Router i stores information about prefixes and their corresponding anchor(s) in its *Prefix Table* (PT^i). The information regarding prefix j is denoted by PT^i_j and consist of the name of the prefix j and the prefix anchor information of prefix j $(PAI^i_j\ )$.

Each entry of PAI^i_{jm} consists of: the anchor m of the prefix j; a "*valid*" flag; vf_{jm} for anchor m, which indicates whether router m advertises prefix j or not; and the sequence number (sn_{jm}) reported by anchor m for prefix j.

Router i updates PT^i based on ALSAs that are received from anchors. The ALSA form anchor m received by router i regarding prefix j is denoted by $ALSA^i_{mj}$ and consists of: the name of the anchor (m), the prefix name j, the sequence number assigned to the prefix by the anchor (usn^i_{mj}), and the *vFlag* indicating if name prefix j is attached to anchor m or detached (vf^i_{mj}).

Router i also maintains a *Routing Table* (RT^i) that stores routing information for each known prefix. The information stored in RT^i regarding prefix j is denoted by RT^i_j, and consist of routing information for the nearest anchor of the prefix j. The routing information includes: the name of prefix j, shortest distance d^i_j to nearest anchor of prefix j, the set of neighbors that are valid next hops for prefix j (S^i_j), the king anchor for prefix j (k^i_j), and a neighbor that is the best next hop in the shortest path to anchor k^i_j of j $(s^i_j \in S^i_j\ )$. The king anchor of prefix j is the anchor with the smallest lexicographic name among those anchors that are at the same shortest distance to j

Router i updates PT^i and RT^i based on the other two tables, LT^i and FT^i, and the information available in ALSAs. The ALSA received by router i sent by anchor m regarding prefix j is denoted by $ALSA^i_{jm}$. The information extracted from $ALSA^i_{jm}$ is the sequence number assigned to the prefix by the anchor (usn^i_{jm}) and the vFlag uvf^i_{jm}.

3.2 Routing to Nearest Replicas

A router calculates the best routes to nearest copies of a prefix in two steps. First, the router calculates valid next hops for all the anchors that advertise that prefix. Second, the router selects some of the neighbors from the previous step as valid next hops to the prefix. For every anchor in the network, the result of the first phase is a directed acyclic graph.

3.2.1 Next-Hop Ordering Condition (NOC)

Every router keeps track of the sequence numbers reported by the routers in the network. Whenever a router receives an RLSA from another router, it checks the sequence number. If the message sequence number is greater than the sequence number stored for that router, the router updates the topology information and also forwards it to its neighbors; otherwise, the router drops the message. Using the sequence number and a termination-detection mechanism similar to the one used in OSPF prevents advertisement messages from circulating in the network forever.

Based on the information received from other routers in the network, router i creates the network topology NT^i and calculates the cost of a path to every destination p in the network from each of its neighbors, as well as router i itself. The router executes Dijkstra's SPF algorithm (or any other shortest-path algorithm) on the network topology to construct a source graph, which constitutes shortest-path trees to every destination from every neighbor. The results are stored in the Distance List of the Forwarding Table (RD^i_p). The router also stores the shortest distance d^i_p.

Router i selects a subset of its neighbors as valid next hops to reach destination p based on the following Next-Hop Ordering Condition (NOC), which we will show prevents permanent routing loops from being created.

NOC: Router i can select its neighbor $n \in N^i$ as a valid next hop to reach destination p if:

$$rd^i_{pn} < \infty \wedge (rd^i_{pn} < rd^i_p \vee (rd^i_{pn} = d^i_p \wedge |n| < |i|)) \quad (1)$$

Router i selects router n as next hop to reach destination p if the neighbor n is closer to the destination or neighbor n and router i are at the same distance to p and $|n| < |i|$.

Figure 1 shows an example network topology. The number next to the node representing a router denotes the distance from the router to destination p.

Figures 2 a,b, and c show the valid next hops for each router to reach destinations p, q, and r, respectively. The arrowheads point to valid next hops for each destination. For instance, router u can be selected as next hop in routers s and t to reach destination p, and u itself can select routers l and f to forward messages to p.

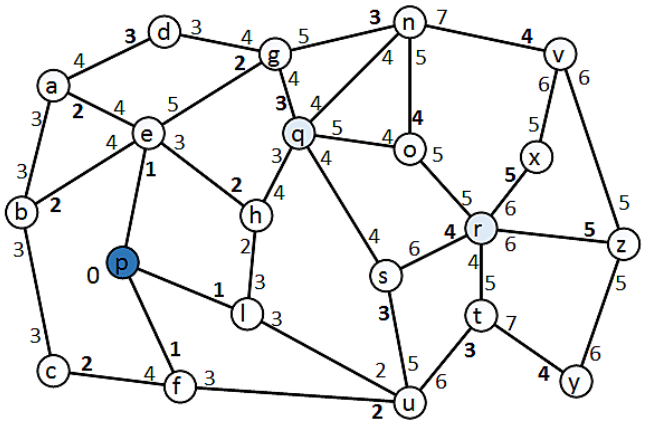

Figure 1: Sample network

Algorithm 1 illustrates how a router updates its forwarding table when it receives an RLSA. We assume that a router waits for a reasonable time after it receives the last RLSA before it executes Algorithm 1, and that the router will not process any other RLSAs while it is executing the algorithm.

3.2.2 *King-Anchor Selection Condition (KSO)*

Based on the information available in an ALSA received by router i, the router looks up its forwarding table and ranks the next-hops for each prefix based on their costs to reach the anchor(s). Each router selects the *king* anchor of a prefix among all the anchors it knows for the same prefix. If two or more anchors are at the same smallest distance from router i, the router selects the lexicographically smallest anchor as the king anchor. Whenever router receives a new ALSA, i.e., an ALSA with the up-to-date sequence number and valid flag equal to 1, the router updates its Prefix Table and calculates the king anchor.

Algorithm 1 Update FT^i

```
Input: $RLSA^i_k$, $LT^i$, $FT^i$
 1: if $rsn^i_k > rsn(k)$ then
 2:     $rsn(k) = rsn^i_k$
 3:     Create the Network Topology $T^i$
 4:     Run Dijkstra's algorithm and Update $rd^i_p$
 5:     for (every $n \in N^i$) do
 6:         Run Dijkstra's algorithm on $n$;
 7:         for (every router $p \in T^i$) do
 8:             Update $rd^i_{pn}$
 9:         end for
10:     end for
11:     for every router $p \in T^i$ do
12:         $RS^i_p := \varnothing$
13:         if $rd^i_p < \infty$ then
14:             for every $n \in N^i$ do
15:                 if $(rd^i_{kn} < rd^i_k) \vee (d^i_{kn} = d^i_k \wedge |n| < |i|)$
    then $RS^i_p = RS^i_p \cup \{n\}$
16:                 end if
17:             end for
18:         end if
19:     end for
20: end if
```

Algorithm 2 illustrates how router i updates its Prefix Table PT^i, when it receives a fresh ALSA from anchor m regarding prefix j.

Algorithm 2 Update PT^i

```
Input: $ALSA^i_{mj}$, $PT^i$
 1: if $usn^i_{mj} > sn^i_{mj}$ then
 2:     $sn^i_{mj} = usn^i_{mj}$
 3:     $vf^i_{mj} = uvf^i_{mj}$
 4: end if
```

A router forwards an ALSA and sets the forwarded flag if the anchor is the king anchor; otherwise it HOLDs the LSA (i.e., it does not propagate the ALSA). If that anchor becomes the king anchor as a result of topology changes or because the current king anchor stops publishing the prefix, then the router restores the ALSA, sets the forwarded flag in the Prefix Table, and forwards the ALSA. Router keeps track of forwarded LSAs and uses this forwarded flag information to avoid sending duplicate ALSAs. Detach ALSAs (ALSA with $vFlag = 0$) are always forwarded, regardless of whether they come from king anchors or not.

KSO: Router i can select anchor m as the king anchor of prefix j (i.e., k^i_j) if the following statement is true:

$$\begin{aligned} &vf^i_{mj} = 1 \wedge \forall [a, vf^i_{aj}] \in PI^i_j,\ vf^i_{aj} = 1 \wedge \\ &[rd^i_m < rd^i_a \vee (rd^i_m = rd^i_a \wedge |m| < |a|)] \end{aligned} \quad (2)$$

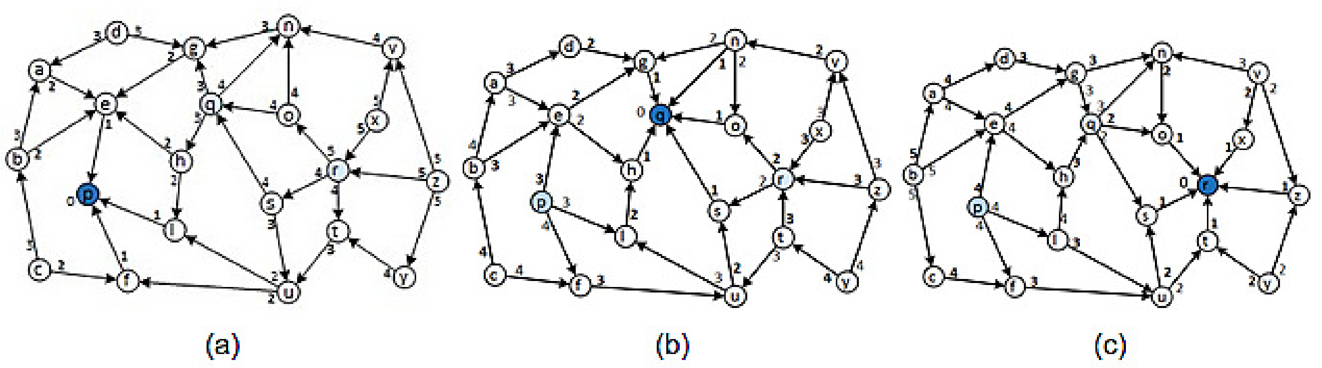

Figure 2: Valid next hops to destinations: (a) destination p, (b) destination q, (c) destination r

The king anchor of a prefix is an active anchor that is the smallest closest anchor among all active anchors of the prefix. The distance to prefix j is the minimum of the distances to anchors advertising j and is equal to the distance to the king anchor of the prefix. Thus,

$$d_j^i = Min\{rd_m^i | m \in PAI_j^i\} = rd_{k_j^i}^i.$$

If no active anchor advertises prefix j or none of the active anchors are reachable, then $PAI_j^i = \varnothing$ and $d_j^i = \infty$ and prefix j is marked as unreachable.

Router i uses Algorithm 3 to select the king anchor of a prefix. After selecting the king anchor, the router selects valid next hops to it. A neighbor can be selected as valid next hop for a prefix if it is valid next hop for the anchor that advertise that prefix and that neighbor is closer to the prefix or it is at the same distance but has lexicographically smaller names.

Algorithm 3 King selection for prefix j

```
Input: $FT^i$, $PT^i$
1: $k_j^i := null; d_j^i := \infty$
2: for ecery $m \in PI_j^i$ do
3:     if $vf_{mj}^i = 1$ then
4:         if $rd_m^i < d_j^i \vee (rd_m^i = d_j^i \wedge |m| < k_j^i$ then
5:             $d_j^i = rd_m^i$
6:             $k_j^i = m$
7:         end if
8:     end if
9: end for
```

3.2.3 *Successor-Set Ordering Condition (SOC)*

The distance from neighbor n to prefix j at router i (d_{jn}^i) is the minimum of the distances to anchors of prefix j through n known by router i:

$$d_{jn}^i = Min\{rd_{mn}^i | m \in PAI_j^i\} \quad (3)$$

Router i selects router n as a valid next hop to prefix j if the following statement is true:

$$d_{jn}^i < \infty \wedge (d_{jn}^i < d_j^i \vee (d_{jn}^i = d_j^i \wedge |n| < |i|)) \quad (4)$$

Router i can select its neighbor n as a next hop toward prefix j if the neighbor is closer to the prefix than router i, or routers i and n are at the same distance from the destination, but neighbor n has a lexicographically smaller name than router i.

Figure 3 shows the final state after executing LSCR assuming routers p, q, and r are all anchors of prefix j. The bold lines in the figure indicate links pointing to the best next hop of each router. For instance, both f and l offer paths of distance two to router u. However, router f is selected as the best next hop at router u toward destination p, because $|f| < |l|$.

Each tuple on a link in Figure 3 represents the closest smallest anchor and distance to that anchor through that link. In this figure, the minimum distance from router d to prefix j is through neighbor g and costs two. Consider router a, router d can reach destination p in three hops via its neighbor a, because a is a valid next hop for that destination and also a is two hops away from prefix j (anchor p) and $|a| < |d|$. These conditions satisfy Eq. 5; therefore, neighbor a is selected as a next hop to reach prefix j.

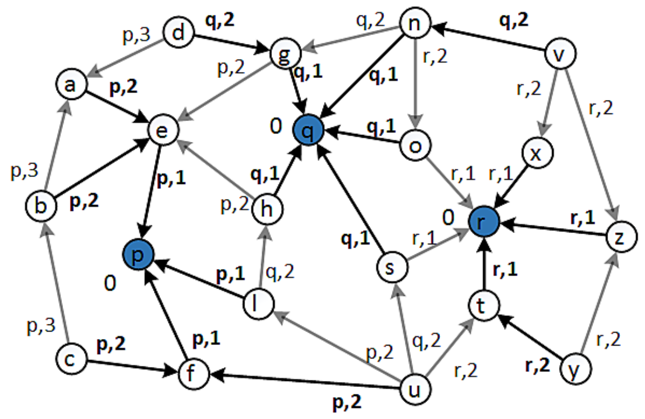

Figure 3: Valid next hops to prefix j

Whenever a router receives an up-to-date LSA indicating a change in the network, it executes Algorithms 1 or 2, depending on whether it receives an RLSA or an ALSA, and then executes Algorithm 4 to update its routing table.

3.3 Naming

The naming schema used in LSCR depends on the ICN architecture in which it is implemented. A hierarchical naming schema such as the one introduced for NLSR can be used for both routers and signaling messages. According this schema, each router is named in the following format: $/ < network > / < site > / < router > /$, where *network* and *site* are assigned based on the network and specific site the router belongs to and *router* is a unique name in that network and site.

LSA messages use the naming schema like:
/<network>/<site>/<router>/LSCR/LSA/TypeID/ <sequence num>. The first part is the name of router initiating the LSA, typeID field distinguishes between *RouterLSA* and *ALSA* and *sequencenum* is the sequence number assigned by the router for that LSA.

Algorithm 4 Update RT_j^i

```
Input: $LT^i$, $FT^i$, $PT^i$, prefix $j$
 1: Ecexute Algorithm 3;
 2: if $d_j^i < \infty$ then
 3:     for (every $n \in N^i$) do
 4:         $d_{min}^n = \infty$
 5:         for (every $m \in PAI_j^i$) do
 6:             if  $rd_{mn}^i < d_{min}^n$ then
 7:                 $d_{min}^n = rd_{mn}^i$
 8:             end if
 9:         end for
10:         if $d_{min}^n < \infty$ then
11:             if $(d_{jn}^i < d_j^i) \vee (d_{jn}^i = d_j^i \wedge |n| < |i|)$ then
12:                 $S_j^i = S_j^i \cup \{n\}$
13:             end if
14:         end if
15:     end for
16: end if
```

4. CORRECTNESS OF LSCR

The following theorems prove that LSCR obtains correct routing entries for routing to the nearest anchor of each prefix, which means that routing tables do not contain any permanent loops. Our proofs rely on known results that sequence numbers can be used to determine that a link-state update is more recent than stored information [11].

We assume that there is a finite number of link-cost and anchor changes up to time t_0, and no more changes occur after that time. We also assume that routers can determine which updates are more recent than others. Also, we assume that every router has correct information about the network topology, which can be be done in many ways for complete or partial topology information [10,11,17].

LEMMA 1. *The king anchor is the same for all the routers along the shortest path from each router to the king anchor.*

PROOF. The proof is by contradiction. Assume that the shortest path P_i from router i to its king anchor of prefix j m_j^i consists of h hops and each router selects its next hop according to Equations 3 and 4.

Let $P_i = \{n_1, n_2, S, n_h\}$, where $n_1 = i$, $n_h = m_j^i$, and $n_{k+1} \in RS_m^n$ for $1 \leq k \leq h-1$. We know that $rd_m^i = rd_{n_k}^i + rd_m^{n_k} \forall k, 1 \leq k \leq h-1$. Assume that router n_p selects router $a \neq m$ as the king router for prefix j; therefore, either $rd_a^{n_p} < rd_m^{n_p}$ or $rd_a^{n_p} = rd_m^{n_p} \wedge |a| < |m|$.

If $rd_a^{n_p} < rd_m^{n_p}$ then $rd_{n_p}^i + rd_a^{n_p} < rd_{n_p}^i + rd_m^{n_p}$. Accordingly, $rd_a^i < rd_m^i$. Hence, router a is the king anchor, which contradict our assumption that m is the king anchor.

If $rd_a^{n_p} = rd_m^{n_p} \wedge |a| < |m|$ then $rd_{n_p}^i + rd_a^{n_p} = rd_{n_p}^i + rd_m^{n_p} \wedge |a| < |m|$. Therefore, $rd_a^i = rd_m^i \wedge |a| < |m|$. Again, router a is the king anchor, which contradict our assumption that m is the king anchor. □

LEMMA 2. *Each router that receives an ALSA from the closest anchor of a prefix advertises that prefix.*

PROOF. There are two types of ALSAs based on the *vFlag* parameter. *Detach LSA*s (i.e., ALSAs with $vFlag = 0$) are propagated in the network using the intelligent flooding mechanism and termination detection is based on sequence numbering. Therefore, every router receives a *Detach LSA* from each anchor, including the closest one.

ALSAs with $vflag = 1$ are propagated using a diffusion mechanism. Based on Lemma 1, all the routers along the shortest path from the king anchor to the router has the same king anchor. Based on the LSA forwarding mechanism, each router forwards an ALSA form the king anchor.

Assume that router i did not receive any ALSA from its new king anchor m. Also assume that the shortest path P_i from router i to the king anchor of prefix j, m_j^i, consists of h hops. Assume that $P_i = \{n_1, n_2, S, n_h\}$ is such a path. King anchor k is the king anchor of $n_p, 1 \leq p \leq h-1$. Also assume that a router $n_m \in P_i$ did not forward the ALSA from its king anchor. This is in contradiction to the forwarding mechanism used for ALSAs. Therefore, router i should have received the ALSA form its king anchor. □

THEOREM 3. *All routers in a network running LSCR must converge to the shortest distance to their nearest anchors of each prefix a finite time after t_0.*

PROOF. Note that there is a finite number of prefixes and there is a finite number of anchors for each prefix. Furthermore, each router processes and forwards each unique LSA only once based on sequence numbers.

Without loss of generality, we focus on a specific prefix j. The prefix can be considered as a virtual node connected to its anchor via a virtual link. Prefix detachment and attachment to the anchor can be considered as a link failure and a link recovery, respectively. For each direction of a link, there is one router (the head of the link) that detects and reports the change in the link in one direction. Therefore, for any link l_i, which can be a physical link or virtual link, each router sends at most one LSA for that link after t_0.

Consider an arbitrary router r_0 that never terminates executing LSCR. That router must send an infinite number of LSA messages after time t_0.

Because the network is finite, there is a finite number of links, and r_0 must process an infinite number of LSAs for at least one link l_f. Because no changes occur after time t_0, router r_0 cannot originate an infinite number of LSAs for any adjacent think after t_0. Furthermore, it is not possible for a router to send an infinite number of messages regarding l_f as a result of receiving an infinite number of LSAs regarding link l_h. It follows that router r_0 can forward an infinite number of LSAs regardingl_f only if it receives an infinite number of LSAs regarding l_f from at least one neighbor.

Accordingly, at least one of the neighbors of router r_0, call it r_1, must send an infinite number of LSAs containing updates for link l_f that makes r_0 process and forward an unlimited number of LSAs. By the same token, neighbor r_1 can send an infinite number of LSAs regarding l_f only if at least one of its neighbors, call it r_2, forwards an infinite number of LSAs regarding l_f to router r_1. However, it is impossible to continue the same line of argument indefinitely because the head node of any link can generate at most one update for that link after time t_0 and the network is finite.

Therefore, LSCR can produce only a finite number of LSAs for a finite number of link or prefix changes and must stop within a finite time after t_0. $\square$

THEOREM 4. *If topology information is correct at each router, no routing-table loops can be formed if NOC is used to select the next hops to the anchor of a prefix at each router.*

PROOF. The proof is by contradiction. Assume that a routing loop L_m for anchor m is formed at time $t_1 > t_0$ when routers update their next-hops satisfying NOC. Assume that $L_m = \{r_0, r_1, S, r_{q-1}\}$ consisting of q routers is such a loop.

We can consider L_m as a path $P_m = \{r_0, r_1, S, r_{q-1}, r_q\}$, where $r_0 = r_q$. Note that $rd^{r_0}_{r_q} = rd^{r_0}_{r_0} = 0$ and $rd^{r_q}_m = rd^{r_0}_m$. Router n_i selects its next hop from $RS^{n_i}_m$ for $0 \le i \le q-1, r_{i+1} \in RS^{n_i}_m$.

Based on NOC, for every router $n_i \in P_m$, it must be true that $rd^{n_i}_{mn_{i+1}} < rd^{n_i}_m$ or $rd^{n_i}_{mn_{i+1}} = rd^{n_i}_m$ and $|n_{i+1}| < |n_i|$. By definition, $rd^{n_i}_{mn_{i+1}}$ is the distance from $n+1$ to m calculated at router i. Routers i and n have the same topology information by assumption. Accordingly, $rd^{n_i}_{mn_{i+1}} = rd^{n_{i+1}}_m$. This results in the following:

$$(rd^{n_{i+1}}_m < rd^{n_i}_m) \vee (rd^{n_{i+1}}_m = rd^{n_i}_m \wedge |n_{i+1}| < |n_i|) \quad (5)$$

Note that $\forall r_i \in P_m, rd^i_m \neq \infty$. A next hop cannot have an infinite distance to the destination, because that would contradict NOC. Therefore, for any two routers $r_u, r_v \in P_m$ and $0 \le u < v \le q$ we have:

$$(rd^{r_v}_m < rd^{r_u}_m \vee (rd^{r_v}_m = rd^{r_u}_m \wedge |r_v| < |r_u|) \quad (6)$$

The equation is valid for any two routers $r_u, r_v \in P_m$ including r_0 and r_q. Hence, it must be true that $rd^{r_0}_m < rd^{r_q}_m$ or $rd^{r_0}_m = rd^{r_q}_m \wedge |r_0| < |r_q|$, which contradict our assumptions. $\square$

LEMMA 5. *For every router i and its neighbor n and for any arbitrary prefix j, it is true that $d^i_{jn} = d^n_j$*

PROOF. The distance of router n from prefix j is the distance of router n from its king anchor: $d^n_j = rd^n_{k^n_j}$. Every router forwards the ALSA from its king anchor; therefore, router i is aware of anchor k^n_j, and so $k^n_j \in PAI^i_j$. From Eq. 3, we have $d^i_{jn} = Min\{rd^i_{mn} | m \in PAI^i_j\} = rd^n_{k^n_j}$. The last equality is derived from Lemma 2, Eq. 2, and the fact that $(k^n_j \in PAI^i_j \wedge k^n_j \in PAI^n_j)$. Therefore, $d^i_{jn} = d^n_j$. $\square$

THEOREM 6. *No routing-table loops can be formed if all routers use LSCR to calculate routes to prefixes and routers have correct topology information.*

PROOF. If the prefix is advertised by just one anchor, based on Theorem 4 no loops can be formed. If more than one anchor advertises a given prefix j, the same conclusion can be derived using Lemma 5 and an argument similar to the proof of Theorem 4. $\square$

5. ROUTING COMPLEXITY

We compute the communication, time, and storage complexities of LSCR and compare it with the complexities of traditional link-state routing and loop-free routing based on distances. The communication complexity of a routing protocol is the number of messages that must be transmitted for the routing protocol to have information required to compute correct routing tables for all the destinations at each router. The storage complexity is the amount of information that must be stored at each router to obtain correct routing tables. The time complexity of a routing protocol is the maximum time needed for all routers to have correct routing information for all destinations.

We use N and E to denote the number of routers and links in the network, respectively. The number of distinct anchors available in the network is denoted by D, the average number of instances of the same destination is denoted by R, the average number of neighbors per router is l, the network diameter is denoted by d, and C denotes the number of distinct prefixes in the network.

Traditional Link-State Routing (LSR):
Both network topology and prefix information are flooded to the entire network when LSR is used. A router that runs LSR sends adjacency LSA and prefix LSAs, one for each local prefix, and each LSA must be sent to all the other routers in the network. Every router stores the complete topology information as well as all instances of all prefixes in the network. Furthermore, the maximum distance between a source and a destination is the network diameter. Accordingly, the time, communication, and storage complexities of LSR are:

$$TC_{LSR} = O(d); \quad CC_{LSR} = O(ERC + lEN); \quad SC_{LSR} = O(RC + E) \quad (7)$$

Loop-free Distance-Vector Routing:
Because of the looping problems of traditional distance-vector routing protocols, the traditional distance-vector routing algorithms cannot be used for routing in ICNs with multi-instantiated prefixes [15]. The Distance-based Content Routing (DCR) protocol solves this problem [14], and maintains routes to the nearest replicas of prefixes that are loop-free at every instant. Hence, we use this protocol as the example of distance-vector routing.

For a given prefix, a router disseminates only its distance to the nearest anchor of the prefix, independently of the number of prefix replicas in the network. Hence, the information a router stores and communicates for a given prefix in DCR is only its distance to the nearest anchor of the prefix, plus the anchor name and the latest sequence number created by that anchor.

As the number of replicas increases, the distances from a router to the nearest replica of a prefix decreases, and it is always the case that the number of hops from any router to the nearest replica of a prefix (x) is at most d hops.

DCR does not incur any routing-table loops, which means that any routing information propagates as fast as the shortest path between its origin and the recipient. Furthermore, the number of messages required for all routers to have a correct distance to a given prefix is $O(E)$, regardless of the number of times R the prefix is replicated. Given that there are C prefixes in the network, the time, communication, and storage complexities of DCR are:

$$TC_{DCR} = O(x); \; CC_{DCR} = O(EC); \; SC_{DCR} = O(C) \quad (8)$$

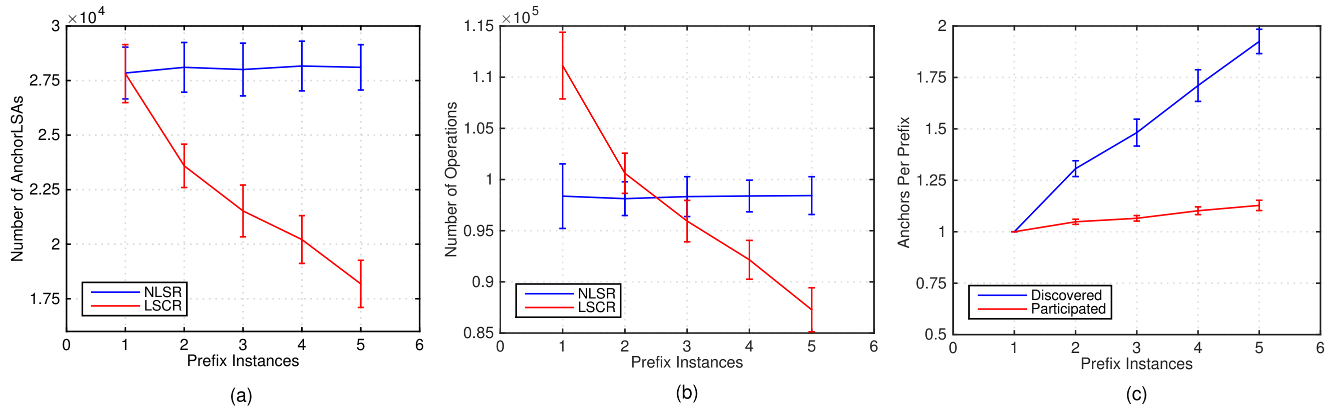

Figure 4: LSCR Performance: (a) average number of LSAs packets sent, (b) average number of operations, (c) average number of discovered anchor and number of anchors selected for routing per prefix per node

Link-State Content Routing (LSCR):
The information required for LSCR to find correct shortest paths to the nearest anchors of prefixes is the complete topology, and the prefix information from the nearest anchor (anchor name and sequence number created by that anchor). Therefore, the storage complexity of LSCR is independent of the number of anchors advertising each prefix.

Given that every router needs the complete topology information, the time required to receive LSAs for all links is the time that a message traverses across the the network diameter d. On the other hand, the distance from any router to the nearest anchor of a prefix can be at most d.

The number of messages exchanged to create the complete topology is $O(lEN)$, just as with traditional link-state routing, given that each of the N routers has l neighbors and LSAs must be flooded. In addition, each router needs to know its nearest anchor to each prefix and also needs to delete anchors and even prefixes due to failures. Based on the diffusion mechanism, LSCR only propagates specific valid ALSAs and the communication complexity incurred for this is just $O(C)$. However, the communication complexity of sending ALSAs for the deletion of a prefix is (CER), where R is number of replicas for a given name prefix.

The storage required by LSCR consists of all the links in the network and all the prefixes in the network. Accordingly, the time, communication, and storage complexities of LSCR are:

$$TC_{LSCR} = O(d); \quad CC_{LSCR} = O(CER + lEN); \qquad (9)$$
$$SC_{LSCR} = O(C + E)$$

6. PERFORMANCE COMPARISON

We compared the performance of LSCR with an optimized version of NLSR using the SCoNET-Sim tool, which is an NS-3 based simulator for content centric networks [28].

To eliminate the differences in performance due to sender-initiated or receiver-initiated modalities, we implemented LSCR and NLSR using sender-initiated signaling, in which control messages are simply Interest messages that carry a payload containing control information. As a result, our implementation of NLSR in the simulation uses a single transmission per LSA, rather than sending LSAs as a result of Interests after neighbor routers determine the differences in their local databases. NLSR propagates LSAs using intelligent flooding.

The scenarios used in our performance comparison assume the The AT&T core network topology, which is a realistic ISP topology [21] consisting of 154 nodes and 184 links. A node has 2.4 neighbors on average, and there are 14 nodes with only one neighbor. Each node has a unique identifier in the simulation model. The existence of a link-level protocol assures that every node detects the loss or recovery of connectivity with its neighbor in a finite time after a router fails to receive the proper control messages a repeated number of times. All messages, link failures, and link recoveries are processed one at time in the order in which they occur and within a finite time. We used a total of 210 content objects and 30 anchors that advertise prefixes in the network. The anchors are selected randomly, and some anchors may have some prefixes in common. On each simulation run, the input event generated was a single link failure or recovery, and a single prefix addition or deletion.

The performance metrics we used to compare NLSR and LSCR were the number of messages, number of events, number of operations, and the average number of replicas per prefix stored in each router. These metrics are measured for five different cases: the initialization process, a link failure, a link recovery, a prefix addition, and a prefix deletion. The number of prefix replicas (prefix instances) is varied from one anchor per prefix to five anchors per prefix.

Figure 4 shows the result of simulations for the number of prefix replicas increasing from one to five. Figure 4-a shows the number of ALSAs sent. As the number of replicas increases, the number of LSAs sent in LSCR decreases, because routers disseminate only LSAs for nearest anchors. The number of operations is the total number of operations performed by each protocol, and is incremented whenever an event occurs or the statements within a for or while loop used in the algorithms that compute shortest paths are executed. Both LSCR and NLSR run Dijkstra's SPF algorithm to find shortest paths to destinations, and in both approaches SPF is run at a node as many times as a node has neighbors.

Figure 4-c illustrates the average number of anchors that LSCR stores for each prefix and the average number of anchors that are participated in routing per prefix per node. As the number of replicas increases the number of replicas

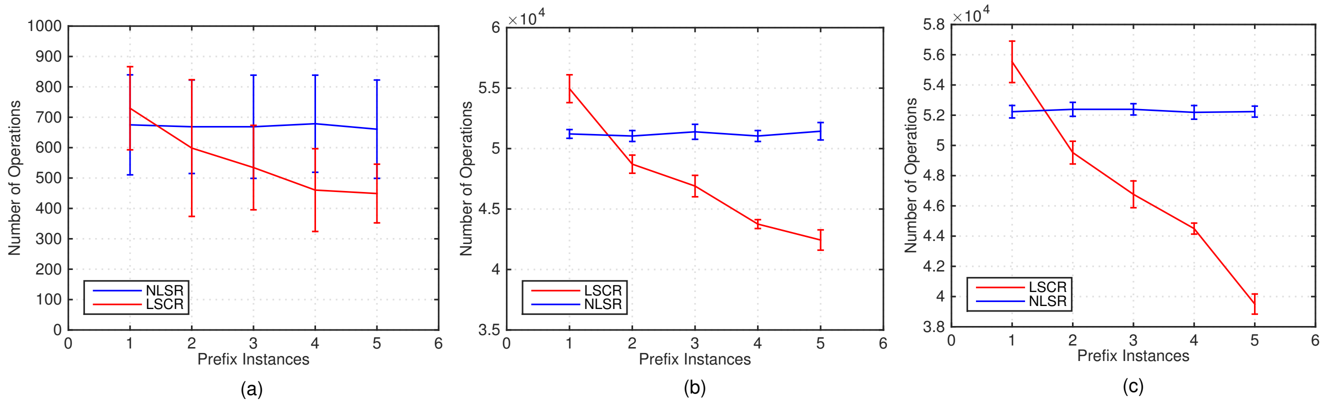

Figure 5: Computational overhead of LSCR: (a) prefix deletion, (b) link failure, (c) link recovery

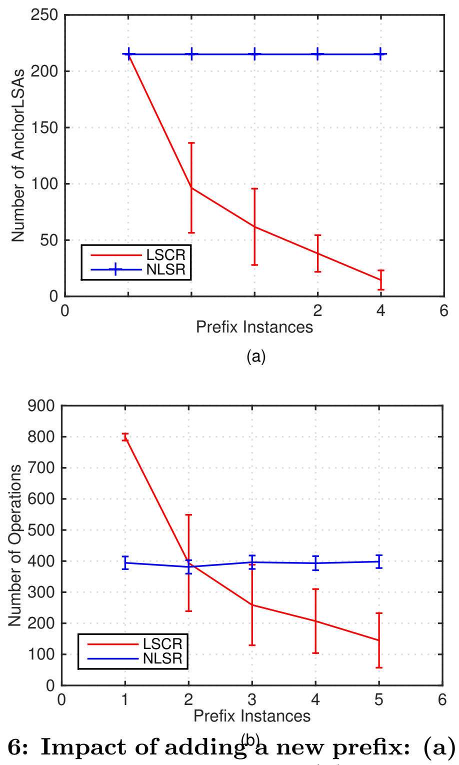

Figure 6: Impact of adding a new prefix: (a) average number of LSAs packets sent, (b) average number of operations

stored in the node also increases but even in a network in which five anchors advertise a prefix, LSCR stores less than two anchors on average, while NLSR stores information for all five anchors.

Figure 5 illustrates the computational overhead of LSCR compared to NLSR for the cases of prefix deletion, link failure and in recovery. A link is selected randomly and deleted from the network. After the deletion, the statistics are measured once the protocols are allowed to converge to the steady state, which means that all messages are processed and no further changes are made to the routing tables. Then the deleted link were added to network, i.e., the link becomes operational, and the statistics are measured after steady state is reached again. In the case of prefix deletion, an anchor is selected randomly and from that anchor a prefix is deleted. The performance are almost the same in all these cases. However, LSCR has better performance when the number of prefix replicas is larger than three.

Figure 6 shows the number of LSAs propagated and the number of operations executed after an anchor advertises a new prefix. In this case an anchor is selected randomly and then it starts to advertise a prefix that is new for that anchor. As the number of replicas increases, the number of LSAs needed to propagate the new anchor information decreases. The number of propagated LSAs in LSCR is almost half the number of LSAs in NLSR when the number prefix instances is two. The computation overhead also decreases in LSCR as the number of replicas for each prefix increases. For instance, the computation overhead of LSCR is half the computation overhead of NLSR when the number prefix instances is four.

7. CONCLUSION

The Link State Content Routing (LSCR) protocol was introduced for name-based content routing in ICNs. LSCR provides multiple paths to the nearest replicas of NDOs or name prefixes. LSCR relies on full-topology information and information about nearest prefix replicas, rather than all prefix replicas. Therefore, its communication and storage complexities are smaller compared to content routing approaches based on the traditional link state approach exemplified by NLSR and OSPFN.

Routers exchange two types of information: Topology information and anchor information. Each router builds a complete network topology based on topology information. A shortest-path routing algorithm is used to calculate the distance to a given destination through each neighbor of a router. Then neighbors are ranked lexicographically based on their distances.

Routers that run LSCR forward anchor information selectively based on a distributed computation of preferred publishers. We showed that routes to prefixes are loop-free once routers have correct topology information, even when the prefixes have multiple replicas.

Although LSCR is more efficient than the traditional link-state approach to name-based content routing, more work is needed to attain higher efficiency. Possible approaches include: communicating partial topology information [10], reducing the frequency with which LSAs have to be sent [17], and improving the way in which routers update anchor information after resource failures.

8. REFERENCES

[1] Content centric networking project (CCN). `http://www.ccnx.org`.

[2] Fp7 PURSUIT project. `http://www.fp7-pursuit.eu/PursuitWeb/`.

[3] Named data network project. `http://www.named-data.net/`.

[4] Nsf mobility first project. `http://mobilityfirst.winlab.rutgers.edu/`.

[5] Publish subscribe internet technology (PURSUIT) project. `http://www.fp7-pursuit.eu/PursuitWeb/`.

[6] Scalable and adaptive internet solutions (SAIL) project. `http://www.sail-project.eu/`.

[7] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, July 2012.

[8] M. Ain, D. Trossen, P. Nikander, S. Tarkoma, K. Visala, K. Rimey, T. Burbridge, J. Rajahalme, J. Tuononen, P. Jokela, J. Kjällman, J. Ylitalo, J. Riihijärvi, B. Gajic, G. Xylomenos, P. Savolainen, and D. Lagutin. D2.3 - architecture definition, component descriptions, and requirements. In *Deliverable, PSIRP 7th FP EU-funded project,*, 2009.

[9] M. Bari, S. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu. A survey of naming and routing in information-centric networks. *Communications Magazine, IEEE*, 50(12):44–53, December 2012.

[10] J. Behrens and J.J. Garcia-Luna-Aceves. Hierarchical routing using link vectors. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 702–710 vol.2, Mar 1998.

[11] D. Bertsekas and R. Gallager. *Data Networks (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[12] A. Carzaniga, M. Rutherford, and A. Wolf. A routing scheme for content-based networking. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 918–928 vol.2, March 2004.

[13] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi. A survey on content-oriented networking for efficient content delivery. *Communications Magazine, IEEE*, 49(3):121–127, March 2011.

[14] J.J. Garcia-Luna-Aceves. Name-based content routing in information centric networks using distance information. In *Proceedings of the 1st International Conference on Information-centric Networking*, ICN '14, pages 7–16, New York, NY, USA, 2014. ACM.

[15] J.J. Garcia-Luna-Aceves. Routing to multi-instantiated destinations: Principles and applications. In *Proceedings of the 2014 IEEE 22Nd International Conference on Network Protocols*, ICNP '14, pages 155–166, Washington, DC, USA, 2014. IEEE Computer Society.

[16] J.J. Garcia-Luna-Aceves, Q. Li, and T. Karadeniz. Cord: Content oriented routing with directories. In *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pages 785–790, Feb 2015.

[17] J.J. Garcia-Luna-Aceves and M. Spohn. Scalable link-state internet routing. In *Network Protocols, 1998. Proceedings. Sixth International Conference on*, pages 52–61, Oct 1998.

[18] J.J. Garcie-Luna-Aceves. System and method for discovering information objects and information object repositories in computer networks, Jan. 9 2007. US Patent 7,162,539.

[19] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: Seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 1:1–1:6, New York, NY, USA, 2011. ACM.

[20] M. Gritter and D. R. Cheriton. An architecture for content routing support in the internet. In *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, USITS'01, pages 4–4, Berkeley, CA, USA, 2001. USENIX Association.

[21] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz. On realistic network topologies for simulation. In *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, MoMeTools '03, pages 28–32, New York, NY, USA, 2003. ACM.

[22] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang. Nlsr: Named-data link state routing protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 15–20, New York, NY, USA, 2013. ACM.

[23] M. Hoque, C. Yi, A. Alyyan, , and B. Zhang. Ospfn: An ospf based routing protocol for named data networking. Technical report, NDN Technical Report NDN, July 2012.

[24] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM.

[25] K. V. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos. On inter-domain name resolution for information-centric networks. In *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*, IFIP'12, pages 13–26, Berlin, Heidelberg, 2012. Springer-Verlag.

[26] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pages 181–192, New York, NY, USA, 2007. ACM.

[27] J. Kurose. Information-centric networking: The evolution from circuits to packets to content. *Computer Networks*, 66:112 – 120, 2014. Leonard Kleinrock Tribute Issue: A Collection of Papers by his Students.

[28] J. Mathewson, M. Barijough, E. Hemmati, J.J. Garcia-Luna-Aceves, and M. Mosko. Sconet : Simulator content networking. In *CCNxCon*, 2015.

[29] J. Raju, J.J. Garcia-Luna-Aceves, and B. Smith. System and method for information object routing in computer networks, June 23 2009. US Patent 7,552,233.

[30] I. Solis and J.J. Garcia-Luna-Aceves. Robust content dissemination in disrupted environments. In *Proceedings of the Third ACM Workshop on Challenged Networks*, CHANTS '08, pages 3–10, New York, NY, USA, 2008. ACM.

[31] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos. A survey of information-centric networking research. *Communications Surveys Tutorials, IEEE*, 16(2):1024–1049, Second 2014.