A Comparison of Name-Based Content Routing Protocols

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

***Abstract*—The first comparison of the performance of name-based content routing protocols based on distance vectors and link-states is presented. The protocols used for this comparison are the Named-data Link State Routing (NLSR) protocol, which is the main representative of name-based content routing based on link states; and the Distance-based Content Routing (DCR) protocol, which is the first name-based content routing protocol based on distance vectors. In the simulation of NLSR, the signaling of NLSR is simplified to minimize the overhead it incurs sending link state advertisements (LSAs), such that a single transmission is need to send an LSA, rather than multiple transmission as is the case with NLSR. The results of simulations show that the ideal version of NLSR requires fewer control messages to react to changes of name prefixes when the number of replicas is very small, and DCR incurs less signaling overhead to react to topology changes or changes in name prefixes when the number of replicas is large.**

***Keywords*-computer network; information centric network; content-based routing;**

I. Introduction

Several Information centric networking (ICN) architectures have been developed as alternatives to the current Internet architecture to address the increasing demand of user-generated content [1]–[4]. The goal of the ICN architecture designs are to provide a cost efficient, scalable and mobile content distribution networking by adopting a content-based model of communication. According to these architectures, messages flow between publishers or caching sites and consumers based on the names of content objects, rather than their addresses. Most of the proposed ICN architectures are based on location-independent content naming and are implemented based on name resolution and name-based content routing. ICN architectures attempt to dissociate content form its producer, so the content can be addressed and matched independent of its source or location.

Section II summarizes several approaches that have been proposed to support content routing based on the names of NDOs that may be replicated in a network. Interestingly, as our summary indicates, all these approaches have been based on algorithms designed for routing to single-instance destinations.

Section III summarizes the operation of the Named-data Link State Routing protocol (NLSR) [5] protocol and the Distance-based Content Routing (DCR) protocol [6]. NLSR is a good representative of content routing based on link states, and DCR is the first example of loop-free content routing based on distance information.

Section IV presents the results of simulation experiments based on ns3 used to compare the performance of NLSR and DCR operating in a realistic topology.

II. Name-Based Content Routing

Name resolution and routing of content are essential in all information centric network (ICN) architectures, and several approaches have been proposed to support content routing based on the names of NDOs that may be replicated in a network. Interestingly, all these approaches are based on algorithms designed for routing to single-instance destinations.

A number of content routing approaches rely on flooding of content requests to cope with the fact that nodes requesting content by name do not know the locations of copies of the content. Directed Diffusion [7] was one of the first proposals for name-based routing of content. Requests for named content (called interests) are diffused throughout a sensor network, and data matching the interests are sent back to the issuers of interests. DIRECT [8] uses an approach similar to directed diffusion for name-based content routing in ad hoc wireless networks subject to connectivity disruption. Nodes use opportunistic caching of content and flood interests persistently within and across connected network components.

A number of approaches are based on maintaining routing information to all replicas of content by means of path-vector algorithms or link-state algorithms. Gritter and Cheriton proposed the Name-Based Routing Protocol (NBRP) [9] as an extension of BGP. The CBCB (combined broadcast and content based) routing scheme for content-based networking [10] is an example of content-routing similar to the receiver-initiated approach to multicasting. CBCB consists of two components. First, a spanning tree of the network or multiple per-source minimal-paths spanning trees of the network are established. Then publish-subscribe requests for content based on predicates are sent between consumers and producers of content over the tree(s) established in the network.

DONA [11] uses flat names for content and either global or local IP addressing and routing to operate. If only local IP routing is used, content requests (FIND messages) gather autonomous-system (AS) path information as they are

forwarded, and responses are sent back on the reverse paths traversed by requests. Within an AS, IP routing is used.

The routing approach in the Mobility First project [12] requires using either network addresses or source routing or partial source routing. Several ICN projects have adopted content routing modalities based on the link-state routing approach (e.g., [13]–[18]). NLSR [19] is the most recent example of name-based content routing based on complete topology information. Routers flood link-state advertisements (LSA) that describe the state of physical links or the name of prefixes of content for which they have local copies.

Some ICN projects (e.g., [18], [20]) adopt content routing modalities based on distributed hash tables (DHT) running in overlays over the physical infrastructure to accomplish name-based routing. A destination is assigned a home location in the DHT that nodes can determine by using a common hash function from the name space of destinations to the name space of nodes in the DHT. DHT nodes can cache known mappings to improve efficiency, and the DHTs are built using underlying routing protocols that discover the network topology.

III. Elements of NLSR and DCR Operation

A. NLSR

NLSR is based on the link-state routing approach to compute shortest paths from each router to every other router and replica of a name prefix. It uses two types of link-state advertisements (LSA): Adjacency LSAs and Prefix LSAs. Based on the information available in LSAs, each node creates a Link State Data Base (LSDB) and ranks its interfaces to forward interests toward a name prefix and fills the FIB.

An adjacency LSA advertises topology information and contains the name of the router, its neighbors, and the list of all active links connecting the router to its neighbors. Each router sends Adjacency LSA at startup and whenever it detects a change in the links to which it is connected. NLSR sends "info" messages periodically to its neighbors to detect changes in the topology. Prefix LSAs advertise name prefixes that are available locally and contains the name of the router and one name prefix. If the router has multiple local name prefixes, it advertises several prefix LSAs, with each such LSA carrying one prefix update. A prefix LSA contains a flag called *isValid* indicating the status of the prefix. A node send the prefix LSA with *isValid=1* if the prefix is registered in that node and *isValid=0* whenever a name prefix is de-registered. An NLSR node receiving this LSA will update the name prefix in the LSDB and update its FIB accordingly.

A router stores the latest version of LSAs it receives or creates in its LSDB and uses this information to find the best next-hops to reach each name prefix and ranks its interfaces. Based on link-state information, each node creates the topology of the whole network and runs an extension of Dijkstra's shortest-path first (SPF) algorithm to calculate multiple paths to each router and ranks the next hops to reach each destination in the network. To calculate the cost of a path using a specific neighbor, the node removes all the links connected to itself except the one that connects the node to that neighbor. Then it runs Dijkstra's SPF algorithm to calculate the distance to reach every destination through that neighbor. This process is repeated for each neighbor. Then NLSR uses this information and prefix information to map a name prefix to the name of a router and creates a routing table entry for each name prefix. Routes calculated by this mechanism are not loop-free and NLSR does not provide any mechanism to rank multiple replicas of the same name prefix.

LSAs are propagated in the network using hop-by-hop synchronization approach (*Sync*) [21] . Each router periodically exchanges its hash of the LSDB with its neighbors. In this way each router can detect the inconsistencies between its own database and the databases of its neighbors and updates its database with the latest LSA information. Sync allows NDN components and applications to define collections of named data in Repositories or *Repo*, called slices. Each router computes a hash tree over the data stored in the LSDB slice and send this root hash to its neighbors. If the root hash values of two neighbors do not match, routers exchange the hash values on the next tree level until they detect the inconsistency. Each router sends special Interest messages, called Root Advise, containing Root Hash Value of its LSDB slice to its neighbor. Neighbor router sends its own root hash value back using Root Advise Reply. If the hash values do not agree the router will recursively request for next level hash values until they find the mismatch in their data bases. Then the NLSR router requests data by sending Content Interest and the neighbor router sends back the data in a Content Reply. The router will update its LSDB with up-to-date information in the repository and will run NLSR algorithm to update the FIB.

NLSR uses a hierarchal naming schema for both routers and LSAs. A router in the network has a name in format of: $/ < network > / < site > / < router >$. 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. Each LSA has the name of format of: $/ < network > / < site > / < router > /NLSR/LSA$ followed by the message specific naming. This naming for Adjacency LSA is $/LsType.1/ < version >$ and for prefix LSA is $/LsType.2/LsId. < ID > / < version >$. $Version$ field indicate the ordering and can be a sequence number. $LsId$ is a unique LSA Id assigned for each prefix.

B. DCR

DCR is the first name-based content routing approach based on distance information that can find routes to any or some replicas of an NDO or name prefix. Routers running

DCR do not need to know the network topology, complete paths to destinations, or all the replicas of a desired content. A router that runs DCR maintains three tables tables for the purposes of routing to the nearest replicas of content: a *link table* stores the list of all neighbors and link cost connecting the router to its neighbors, a *neighbor table* keeps track of routing information reported by each neighbor (including the router itself), and a *routing table* stores routing information for each known prefix.

A router advertising a zero distance to a name prefix is called an *anchor* of the prefix. Each router sends periodic update messages to its neighbors containing a list of updates to the distances to name prefixes. Each update states the distance to a name prefix, the closest anchor for the prefix, and a sequence number assigned to the prefix by the anchor. DCR uses the Successor-Set Ordering Condition (SOC) to select valid next hops in a way that no routing-table loops are created in the network. Based on information reported by each neighbor and SOC, router i can select its neighbor router k as a next hop to reach name prefix j if and only if:

1) Router k reports a new anchor of prefix j that node i does not know or the most recent sequence from a previously known anchor.
2) If router i has a finite distance to prefix j: Router k has shorter distance to j, or routers i k have the same distance to j, but k is a lexicographically smaller name than i. If router i has an infinite distance to prefix j: Router k offers the smallest finite distance to j among all its neighbors, and also has the lexicographically smallest name among neighbors offering the smallest distance to j.

Each router uses SOC to rank its neighbor for each name prefix, whenever it detects a change in link costs or its neighbor table. Router i computes the distance to a prefix as the minimum of distance of paths using neighbors that satisfy SOC. Each router sends the updated routing information to its neighbors using update messages.

The naming schema for DCR depends on the ICN architecture in which DCR is implemented. A flat or hierarchical naming schema can be used for both routers and update messages. If a hierarchical schema is used, each router is named in the following format: $/ < network > / < site > / < router > /$. The update messages can use a naming schema like this: $/ < network > / < site > / < router > /DCR.$". DCR requires a mechanism to compare router names and rank them lexicographically.

IV. Performance Comparison

A. Protocol implementation

We implemented DCR and an optimized version of NLSR using the ns3 simulator tool with extensions for content centric networks [22]. SCoNet supports CCNx v.1 specifications and messages based on the TLV format.

The signaling overhead incurred in the transmission of an LSA in NLSR is much larger than the overhead incurred by DCR, which is based on sender-initiated signaling and incurs a single transmission per update message.

To eliminate the differences in performance due to sender-initiated or receiver-initiated modalities, we implemented DCR 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 them as a result of Interests after neighbor routers determine the differences in their local databases. NLSR propagates LSAs using the intelligent flooding mechanism commonly used in link-state routing. We denote the optimized implementation of NLSR by *i-NLSR*.

In our simulation of DCR, whenever a node receives an update, it checks its information against the information stored in its neighbor table. If it detects any changes in the anchor, distance or sequence number of a name prefix, it updates the information in the neighbor table and schedules a routing update. A router waits to receive updates from other neighbors before changing its routing table. A router reports the updated routing information to its neighbors in its next update message.

For simplicity, our implementation of DCR is such that each update message contains the information regarding all the prefixes known to the router. In practice, only those name prefixes that have changed since the last update message was sent would need to be reported. Each anchor sends a new sequence number on each update message it sends for locally-available prefixes.

For the case of i-NLSR, whenever an LSA is received, the router updates its LSDB and schedules the routing update to rank its interfaces to reach the destination. If the received LSA is an adjacency LSA the router calculates multiple next-hops for each destination and updates its routing table using the NLSR multi-path calculation mechanism described in Section III. If the router receives a prefix LSA, it maps the name prefix to the destination and ranks the next-hops based on routing table information for that destination, calculated in previous phase. If two or more routers advertise the same name prefix, the faces are ranked based on distance to closest router advertising the name prefix. Adjacency LSA and prefix LSA carry the information described in Section III, as specified in [5].

B. Simulation Scenario and Parameters

Network model: In this study, the AT&T core network topology shown in Fig. 1 was used, which is often used as a realistic topology for simulations [23]. The AT&T topology has 154 nodes and 184 links. A node has 2.4 neighbors on average, and there are 14 nodes with only one neighbor. In our simulation model each node has a unique

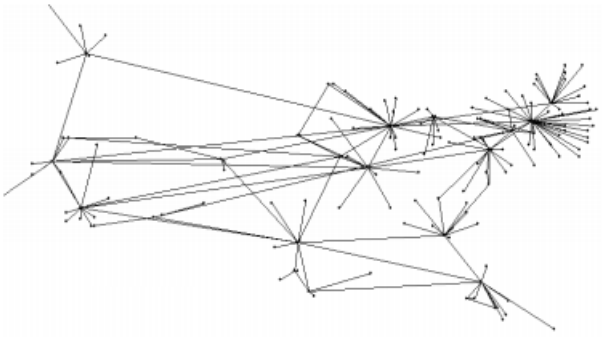

Figure 1. AT&T Network Topology, adapted from [23]

identifier. The hop count is measured as the distance to a destination; therefore, the path cost is the number of hops between a source and a destination. In the implementation, 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 failure, and link recovery are processed one at time in the order in which they occur and within a finite time.

In our scenario, 30 nodes are selected as anchors that advertise 180 unique name prefixes and all anchors have a prefix list of the same size. Each prefix may have more than one anchor. The anchors are selected randomly, and two or more anchors may have some prefixes in common. For instance, a network with an average of three replicas has 540 NDOs and each anchor publishes 18 unique name prefixes.

The simulations were run 20 times and different seeds and several quantities are measured in the network. On each run, the input event generated was a single link failure or recovery, and a single prefix addition or deletion. After each of the events, the protocols are allowed to converge to the steady state, which means all the messages are processed and no further changes are made to the routing tables. The links or prefixes that are deleted or aded are selected randomly. Routers perform their computations in zero time. i-NLSR sent *info* messages every 10 seconds and sends LSA whenever it selects a change in topology or local prefixes. DCR sends update messages each 10-second interval. A neighbor is considered as unreachable if the node does not receive four consecutive *info* messages, in the case of i-NLSR or *update* messages, in the case of DCR.

The simulation started from a topology in which all the links and anchors are operational, and all prefixes are attached to the anchors. The network was in steady state before any changes. All the quantities were measured from the time that a router detected a change, until the protocol converged. The performance metrics are:

- *Messages:* The total number of control messages transmitted over the network. The number of messages for i-NLSR includes the number of Hello messages, adjacency LSAs, and prefix LSAs. In DCR this metric indicates the total number of update messages transmitted as a result of any changes.
- *Events:* The total number of updates that must be processed by the protocol, including changes in neighbor table in the case of DCR and changes in link status or prefix status in the case of i-NLSR.
- *Operations:* The total number of operations performed by each protocol to calculate the routing table. The operation count was incremented whenever an event occurs, and whenever the statements within a loop are executed.

The flooding needed in i-NLSR was implemented in such a way that a router forwards the LSAs toward those neighbors that did not send the LSA. Therefore, the control messages in this study gives us a lower bound of the total number of messages that would be required by NLSR.

For the case of i-NSLR, the Dijkstra-SPF algorithm inserts data into a priority queue. The number of operations required to do so is estimated to be $log_2 N$, where N is the number of nodes. The simulation used the BOOST library with the time complexity of $O(NlogN + E)$ [24].

C. Performance Results

The results of our simulation experiments are shown in Figures 2 to 6. The mean and the standard deviation of the value distribution are given. In each graph, the horizontal axis is the average number of replicas of a name prefix, and the average quantity (number of messages, events, and operations) is presented for each router.

The results for the number of operations and events for DCR are an upper bound, given that our implementation sends all name prefixes, rather than just those that changed since the last update.

Figure 2 shows the results for the initialization phase. Routers do not have any information about the topology, anchors, or prefixes at the beginning, except for those prefixes that are available locally. Routers start sending their local information for a random time after starting the simulation.The results give an upper bound for real events, such as refreshing the LSDBs in all nodes or adding a new node to the network.

In DCR, a router sends information regarding the closest anchor to each known prefix to its neighbors. Therefore, the total number of messages for DCR does not change as number of replicas increases, but the number of prefix LSAs in NLSR is proportional to the total number of prefixes and increases as the average number of anchors per prefix increases. The operation count for the Dijkstra-LS algorithm (replicated at each node) was substantially higher than for the operation count in DCR. NLSR updates the LSDB whenever it receives an up-to-date LSA and schedule routing

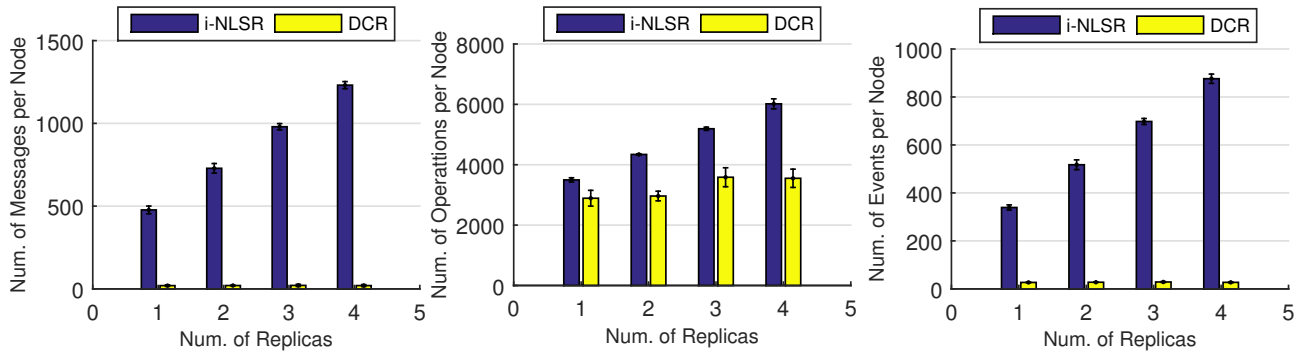

Figure 2. Initialization: average number of messages sent per node, average number of operations done per node, average number of events per node

update; therefore, the number of events per node increases as the number of messages increases.

The results of adding a new prefix to the network are depicted in Figure 3. Routers running DCR exchange more messages than NLSR to converge when the number of replicas is small. However, starting with three replicas, the number of messages is comparable, and the number of messages in NLSR grows with the number of replicas, while the opposite is true for DCR. For the case of prefix deletion, the number of messages needed in DCR remains the same after two replicas, while it keeps increasing in NLSR.

The results for link failures and recoveries are depicted in Figures 5 and 6. Each router that runs NLSR has to process LSA updates corresponding to each direction of a link, unless the link connects to a leaf node in the network. The number of messages exchanged per node is smaller in DCR than in NLSR, independently of the number of replicas of prefixes maintained in the network. A router executing NLSR runs Dijkstra on every node and every single neighbor; therefore, the computation due to running Dijkstra's SPF algorithm is dominant on both link failure and recovery, and the computation cost is higher than in DCR.

V. Conclusions

The performance of DCR and an optimized version of NLSR was analyzed. The simulation results show that reporting distances to the nearest replicas of name prefixes is more efficient than maintaining complete topology information and information about the routers where the various replicas of name prefixes reside. The overhead in NLSR becomes an issue when the average number of replicas per name prefix grows beyond two. The results also show that, for DCR to work efficiently, update messages in DCR should describe updates made to distances to name prefixes since the last update was sent, rather than having each update message contain information about all name prefixes.

Our study suggests various important avenues of fruitful research. First, a routing approach based on link-state information in which routers communicate information about only the nearest prefix replicas should be investigated and its performance should be compared against DCR, which supports routing to nearest prefix replicas using distance information. Second, content routing approaches in which routers are not required to send periodic updates should be investigated for both link-state and distance-vector approaches. The use of event-driven routing updates instead of periodic dissemination of LSAs or distance updates can reduce the signaling overhead due to routing substantially. Third, the importance of sender-initiated signaling mechanisms in CCN and NDN should be quantified.

References

[1] J. Kurose, "Information-centric networking: The evolution from circuits to packets to content," *Computer Networks*, vol. 66, pp. 112–120, Jun. 2014.

[2] M. Bari, S. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 44–53, Dec. 2012.

[3] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, "A Survey on content-oriented networking for efficient content delivery," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 121–127, Mar. 2011.

[4] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks - HotNets '11*. New York, New York, USA: ACM Press, Nov. 2011, pp. 1–6.

[5] A. K. M. Mahmudul-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*. New York, New York, USA: ACM Press, Aug. 2013, p. 15.

[6] J. J. Garcia-Luna-Aceves, "Name-based content routing in information centric networks using distance information," in *ACM Conference on Information-Centric Networking*, Paris, France, September 2014, pp. 24–26.

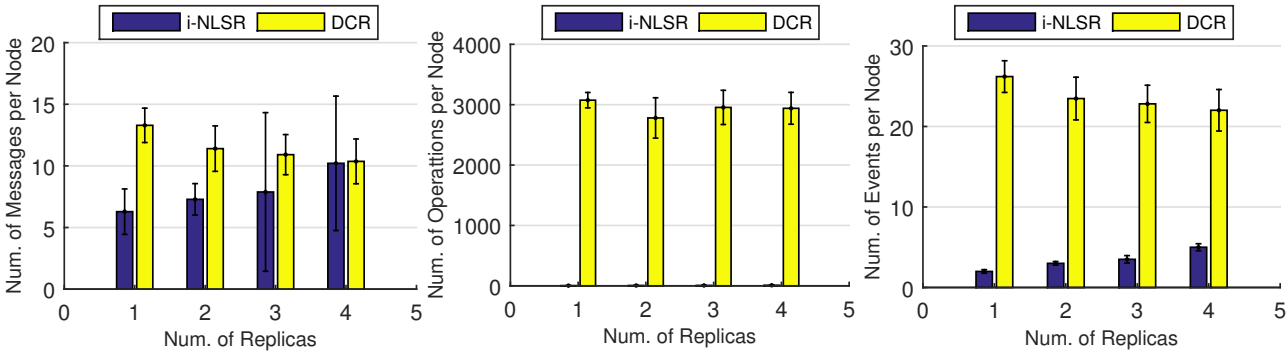

Figure 3. Adding a prefix: Average number of messages sent, operations done, and events processed per node

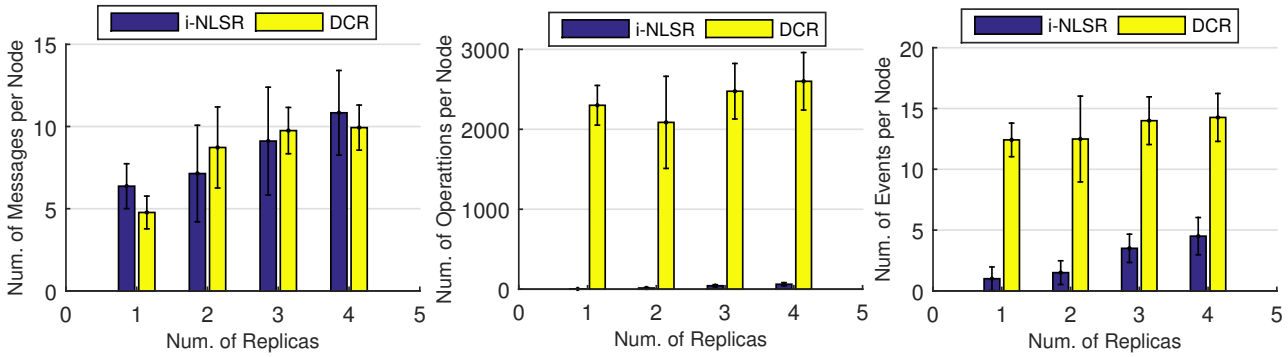

Figure 4. Deleting a prefix: Average number of messages sent, operations done, and events processed per node

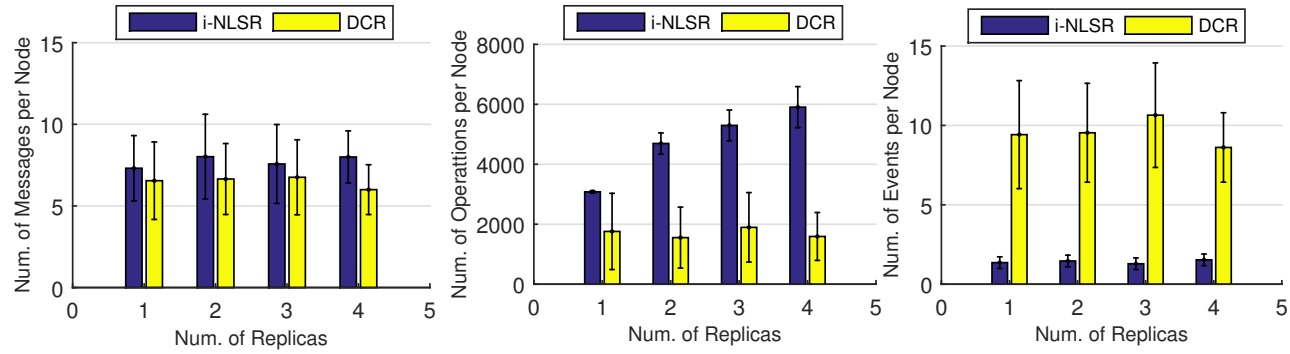

Figure 5. Link failure: Average number of messages sent, operations done, and events processed per node

[7] 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*. New York, New York, USA: ACM Press, Aug. 2000, pp. 56–67.

[8] 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*. New York, New York, USA: ACM Press, Sep. 2008, p. 3.

[9] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet," p. 4, Mar. 2001.

[10] A. Carzaniga, M. Rutherford, and A. Wolf, "A routing scheme for content-based networking," in *IEEE INFOCOM 2004*, vol. 2. IEEE, pp. 918–928.

[11] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, p. 181, Oct. 2007.

[12] Mobility first project. [Online]. Available: http://mobilityfirst.winlab.rutgers.edu/

[13] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies* -

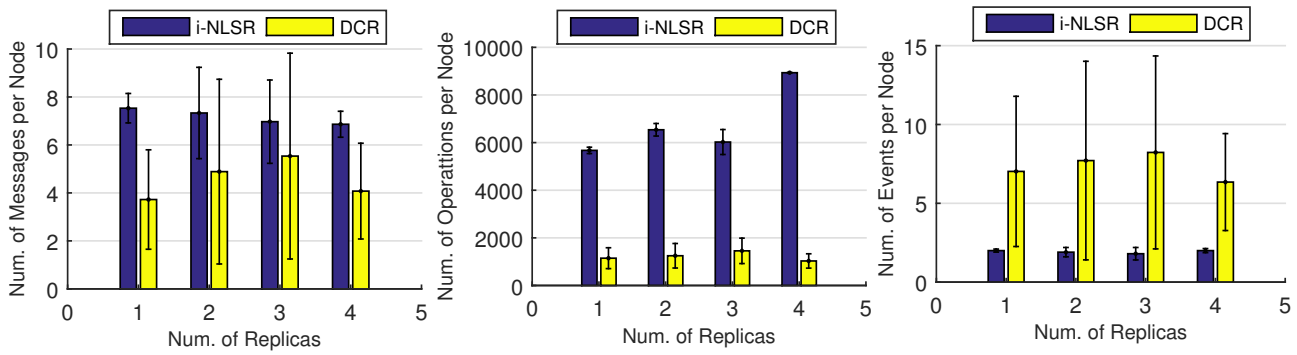

Figure 6. Link recovery: (Average number of messages sent, operations done, and events processed per node

CoNEXT '09. New York, New York, USA: ACM Press, Dec. 2009, p. 1.

[14] Content centric networking project (CCN). [Online]. Available: http://www.ccnx.org/releases/latest/doc/technical/

[15] Content mediator architecture for content-aware networks (COMET) project. [Online]. Available: http://www.comet-project.org/

[16] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini, "CONET," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking - ICN '11*. New York, New York, USA: ACM Press, Aug. 2011, p. 50.

[17] Named data network project. [Online]. Available: http://www.named-data.net/

[18] Scalable and adaptive internet solutions (SAIL) project. [Online]. Available: http://www.sail-project.eu/

[19] A. K. M. Mahmudul-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*. New York, New York, USA: ACM Press, Aug. 2013, p. 15.

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

[21] Ccnx synchronization protocol. [Online]. Available: http://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html

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

[23] 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*. New York, New York, USA: ACM Press, Aug. 2003, p. 28.

[24] J. Siek. Boost graph library: dijkstra shortest paths (BOOST). [Online]. Available: http://www.boost.org/doc/libs/1_57_0/libs/graph/doc/dijkstra_shortest_paths.html