UNIVERSITY OF CALIFORNIA
SANTA CRUZ

FORWARDING AND ROUTING ALGORITHMS FOR INFORMATION CENTRIC AND MOBILE AD HOC NETWORKS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Maziar Mirzazad Barijough

December 2016

The Dissertation of Maziar Mirzazad Barijough
is approved:

Prof. J.J. Garcia-Luna-Aceves, Chair

Prof. Katia Obraczka

Prof. Brad R. Smith

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

Abstract

Forwarding and Routing Algorithms for Information Centric and Mobile Ad hoc Networks

by

Maziar Mirzazad Barijough

It has been shown that the current Internet architecture is not well suited to current and emerging traffic demands. Many content distribution technologies such as content delivery networks (CDN), and peer-to-peer systems have emerged to allow content access by name rather than server location or address. Furthermore, to respond to the increasing volumes of traffic for such applications as video on demand and cloud computing, many efforts have been undertaken to enable caching, content replication, and processing within the network. Such technologies are highly dependent on the distribution channel and perform as an overlay on the current Internet architecture, which results in a number of inefficiencies.

Information Centric Network (ICN) architectures have been proposed as an alternative to the current Internet architecture. The focus of ICN architectures is on caching, replicating, and distributing data by name, independently of their locations.

In this dissertation, the most popular ICN architectures, Named Data Networking (NDN) and Content-Centric Networking (CCNx) are evaluated and problems of such designs in different aspects such as caching, forwarding, and security are introduced. To address the weaknesses of these architectures, we propose a new ICN architectures and compare it with NDN and CCNx using simulation experiments. The results of these experiments show that the proposed new architectures improve network performance in terms of loop detection, forwarding table size, routing table size, processing overhead, and scalability.

Although the aggregation of packets which is used in some ICN architectures might not be very helpful in presence of in-network caching, we show that it can be highly effective where the amount of signaling overhead plays a major role in the performance of the protocol. We propose a new routing algorithm for mobile ad hoc networks, called ADARA, which improves the performance of on-demand routing protocols by aggregating route request packets. Results indicate that aggregating route requests can make on-demand routing more efficient than existing proactive or on-demand routing protocols.

To the love of my life Sara,

To my amazing parents Leili and Ismaeil,

To my magnificent sisters Saharnaz and Sanam.

Acknowledgments

I would like to send my special thanks to Prof. JJ, Garcia-Luna-Aceves, for trusting me and giving me the unrivaled opportunity to work with him as a PhD student. His extreme support, guidance and encouragement, made this journey a lot easier for me. JJ's deep insight and knowledge helped me at every stage of my research and PhD and his sense of humor gave an amazing energy to continue the path I started. It was a pleasure to work with such a smart, motivated, passionate and knowledgeable person. Muchsimas Gracias!

I would like to also thank Prof. Katia Obracazka and Prof. Brad Smith, for participating in my dissertation committee and for their helpful advise and feedback.

Also I would like to thank Prof. Djamshid Tavangarian, Dave Oran, and Zbigniew Sufleta who helped me with their extraordinary guidance and support during my education and career.

I thank my fellow labmates and friends in Computer Communication Research Group (CCRG), Ehsan Hemmati, Turhan Karadeniz, Spencer Sevilla, and James Mathewson, who have been very kind and helpful during my years of PhD.

I would also like to thank my parents Leili, Ismaeil, and my sisters Saharnaz, Sanam for their love, support and encouragement.

Most of all I would like to thank my amazing wife, Sara, for her unconditional support and patience.

Chapter 1

Introduction

Information Centric Networks (ICN) initially introduced by Van Jacobson [14] to address the current Internet traffic requirements. The goal of ICN architecture is to address content distribution and delivery opposed to host-to-host communication done by IP. Also in Information Centric Network, content are requested by their names rather than their address or location.

Information Centric Networks has gained attention of many Internet and Networking researchers and many projects have been defined to address different aspects of such networks.

In-network caching is one of the most important features of ICNs. In these networks, all of the routers are capable of caching content to satisfy users demand using local cache, instead of fetching data from the main producer. Storing contents in limited sized caches in different nodes efficiently to improved performance is the main goal of research in this field.

Routing is another aspect which is different in Information Centric Networks compared to

IP world. In ICNs, routing is done based on Data object Names, not addresses and packets are sent on a hop-by-hop basis rather than end-to-end conversation. Populating routing tables to address replicas of content with different names and maintaining a scalable routing table is the main issue in this context. Furthermore, hop-by-hop forwarding of data packets using routing tables and making decision to choose between different paths to address looping problem, load balancing, efficiency, and etc is one of the major research agendas. Congestion Avoidance and Control problem in Information Centric Networks is very different compared to IP networks. In ICNs, congestion control, load balancing, rate control, and etc can be done in a hop-by-hop basis rather than end-to-end which makes it more efficient compared to end-to-end congestion control.

In this thesis we evaluate different aspects of current ICN architectures along with protocols defined to perform on them. We show that, current architectures and protocols, face different problems in specific situations. To address such problems, we introduce new architecture and protocols that perform much better in different aspects, compared to current methods.

In section 2, we present different architectural proposals made in literature for information Centric Networks. Problems with current forwarding strategies are discussed in chapter 3 and our proposal, SIFAH for correct forwarded strategy is presented later. In chapter 4 we evaluate advantages and disadvantages of Pending Interest Tables using simulation, and propose a different architecture to perform the same tasks with much higher efficiency. In chapter 5 we propose CCN-GRAM, an interest based forwarding strategy which uses anonymous datagrams for interest and data messages.

In chapter 6 we present an efficient interest based multicasting strategy based on Datagram Content Centric Networking approach, which doesn't maintain per interest forwarding state in forwarding tables. In chapter 7 we propose CCN-RAMP, an architecture based on CCN-GRAM which enables higher performance at Internet Scale by integrating name resolution and routing. An On-demand routing protocol for Mobile Ad hoc networks in proposed in chapter 8, which improves performance of routing protocols by reduces signaling overhead mainly by aggregating the route requests .

Chapter 2

Related Works

In recent years many architectures have been proposed for Information Centric Networks which were pioneered by Translating Relaying Internet Architecture integrating Active Directories (TRAID) project [129]. Named Data Networking [17] and Content Centric Networking [14] are two of the most popular ones. In next section we are going to briefly discuss some of the architectures and their forwarding strategies.

2.1 Directed Diffusion for Sensor Networks

Directed diffusion is probably the first data centric architecture, proposed by C. Intanagowiwat and et. al. for wireless sensor networks [13]. As shown in figure 2.1 [13], in this architecture, a sink node sends a request for a task, called interest packet toward the source node. The interest packet is leaves breadcrumbs in the cache of every hop that it traverse while being propagated to the source node. When node receives an interest packet, it adds an entry in the cache or updates an existing entry. An existing entry indicates that a similar interest has already traversed this node and probably has been propagated. Current node may decide to propagate such interest to all or subset or none of the neighbors based on cache entry. By receiving the interest packet, source node will perform the task and send the data packets toward the sink node, using the breadcrumbs. A node that receives the data, will check the cache for entry of the interest. If no such entry exists, it will drop the data packet. If such entry exists, it will check the data cache. Existence of same data in the data cache, can be an indication of loop in data path.

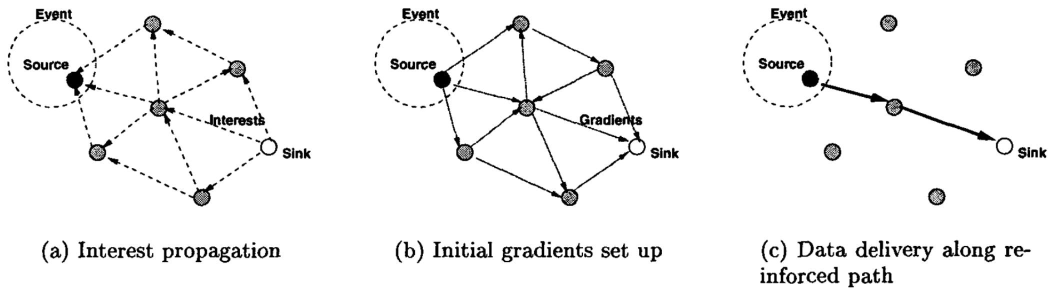


Figure 2.1: Data Propagation in Directed Diffusion [13]

2.2 Data Oriented Network Architecture

The Data Oriented Network Architecture (DONA) proposed by Koponen, et al. [8], is a clean-slate redesign of Internet naming and name resolution. As shown in figure 2.2 [8], DONA includes a flat naming system with a hierarchical name resolution: Content names in DONA are in the form of P:L in which P is the globally unique principal field, which is the hash of publisher's public key. Publisher and cache nodes, register to the resolution infrastructure which consists of Resolution Handlers (RHs). Each RH table includes entries with three fields of name, next hop and hop count. Each entry points to the next hop toward the publisher of the requested name. These tables are filled by REGISTER messages that are generated by publisher nodes. When a RH receives a REGISTER message, it either creates new entry in the RH or updates the existing entry in case the REGISTER message introduces a shorter distance to the content. After updating the table the RH will forward the REGISTER message to the parent RH, until the content is registered in the root RH. Request for content are sent by consumers in the form of FIND packets toward the RH that has the path to the content. When a RH receive a FIND message, it will lookup the table, in case there is a route toward that name, the FIND message is sent to the next hop RH. Otherwise it will be forwarded to the parent RH.

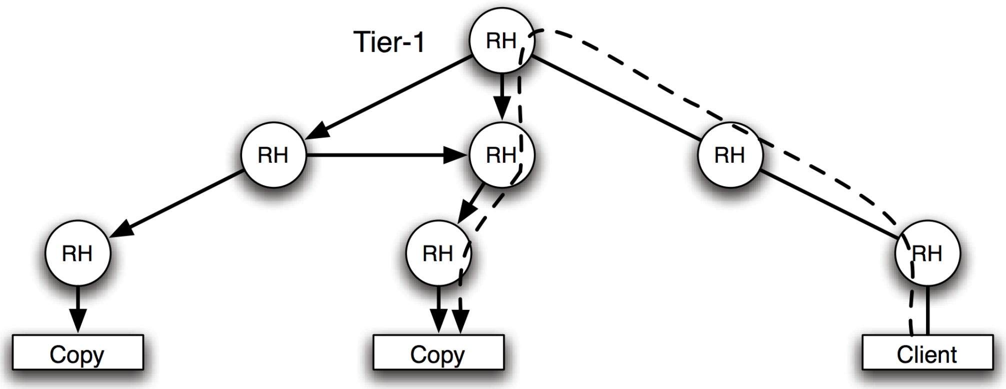


Figure 2.2: DONA Architecture [8]

2.3 Publish Subscribe Internet Routing Paradigm

Publish Subscribe Internet Routing Paradigm (PSIRP) is a publish/subscribe model for ICN architecture [28]. In PSIRP, every content has a flat name in the form of P:L as in DONA, which is called Resource Identifier. In addition to that, for every content a Scope Identifier is also assigned. SId, describes the different information about data including access rights, availability, reachability, replication and etc. All data request packets and data packets include the pair of RId and SId. In PSIRP the whole network is divided into domains. As shown in figure 2.3 [28] nodes in each domain will be one of the these types: RendezVous Nodes (RN), Topology Node (TN), Branching Node (BN), Forwarding Node (FN). Each domain has one RN, one TN, one BN, and many FNs.

- RN nodes are responsible for matching the publishers and subscribers, and resolution of the names, which can be different for RNs of different domains. But all of RN nodes are connected with Distributed Hash Tables (DHTs), which makes their scope global.

- TN nodes are responsible for keeping and exchanging the topology information with other TN nodes in other domains. Furthermore routing information is also handled by TNs.

- BN nodes are responsible for routing protocol implementation. These nodes basically build up the routing map.

- FN nodes, are responsible for forwarding the content objects from publishers to the subscriber nodes. These nodes keep track of the request for content packets in a bloom filter based structure, which is then used to forward the data packets to the subscribers.

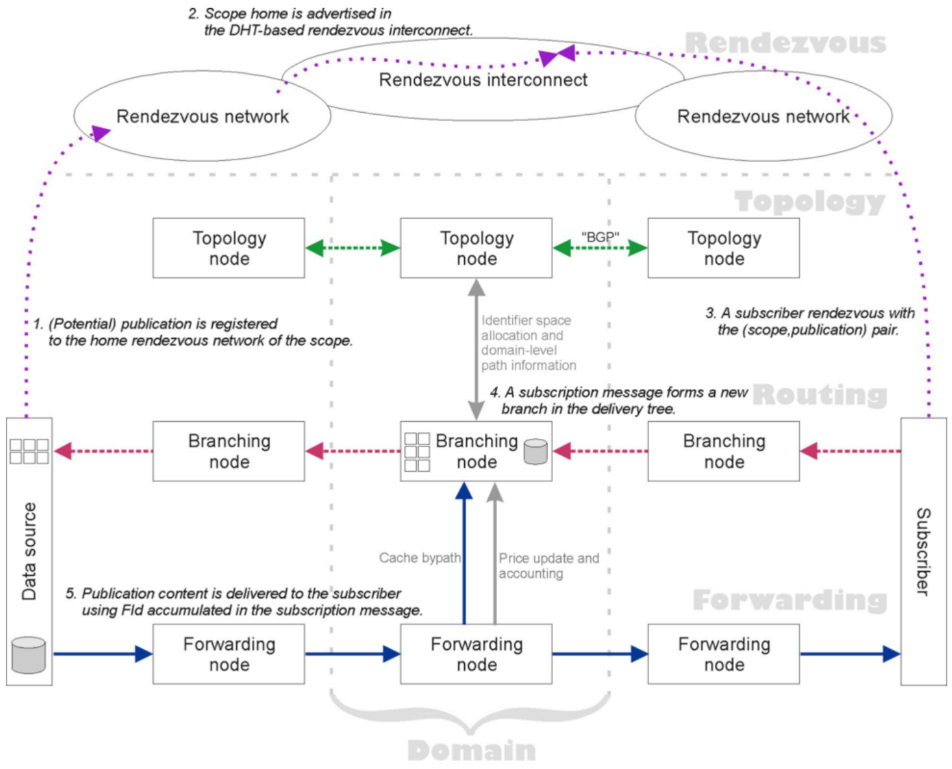

2. Scope home is advertised in the DHT-based rendezvous interconnect.
Rendezvous
Rendezvous interconnect
Rendezvous network
Rendezvous network
Topology
Topology node
Topology node
"BGP"
Topology node
1. (Potential) publication is registered to the home rendezvous network of the scope.
Identifier space allocation and domain-level path information
3. A subscriber rendezvous with the (scope,publication) pair.
4. A subscription message forms a new branch in the delivery tree.
Routing
Data source
Branching node
Branching node
Branching node
Subscriber
Cache bypath
Price update and accounting
5. Publication content is delivered to the subscriber using FId accumulated in the subscription message.
Forwarding
Forwarding node
Forwarding node
Forwarding node
Domain

Figure 2.3: PSRIP Architecture [28]

2.4 Named Data Networking

Named Data Networking is a project supported by NSF which aims at defining architecture for Information Centric Networks. In this architecture, information fetching is done in a publish/subscribe model.

2.4.1 Main Components of NDN Architecture

Main components of Named Data Networking architecture includes:

- Consumer: An application over node which sends requests for content
- Producer: An application which is the publisher of content

- Interest Message: Type of message generated by consumer which includes information about requested content
- Data Object: Type of message which includes content requested by consumer
- Routers: Intermediate nodes which propagate Interest Messages and Data Objects.

Further more each router include different components:

- Forwarding Information Base (FIB): This table is used by router to forward interest packets toward the producer of data. Each entry of the FIB table includes a name prefix and a set of faces which show the next hop toward producer.
- Pending Interest Table (PIT): Pending Interest Table is used to keep track of path taken by interest message toward the producer. Each entry of the PIT table includes the name of the interest, set of incoming faces which show the previous hops, and set of outgoing faces which show the next hops taken by interests of that name. PIT table can also be used for other purposes such as aggregation of Interest Messages, Congestion Control, and etc.
- Content Store (CS): Content Store is used to in-network caching of Data Objects. Content Stores can satisfy Interest Messages on behalf of producers in case they have cached Data Objects previously.

Consumer, which is an application over a node in network, will send request for a Data Object using Interest Messages. This Interest Message will be forwarded by routers toward the publisher of the content which is called Producer. Producer will reply to the interest message by sending the Data Object related to the requested name and routers will forward the Data Object toward the consumer. As shown in figure 2.4 [18] Each Interest Messages consists of different fields:

- Name: Refers to the requested Data Object name.
- Nounce: A random number generated by consumer to make unique Interest Packets with equal names in network.
- MetaInfo: Includes different information about requested object and Interest itself, like interest life time which shows the validity period of the interest.

Also data packet has different fields, as shown in figure 2.4 which includes:

- Name: Unique Name of Data Object
- Content: Payload of the Data Object
- Signature: Used to verify Data Objects by routers and consumers
- MetaInfo: Includes different information about Data Object including Freshness, Chunk Number, Content Type, ...

2.4.2 Forwarding Strategy of NDN Architecture

As shown in figure 2.5 [18] and algorithms 1 and refalgo-ndn-forward, each router consists of three main components: Forwarding Information Base (FIB), Content Store (CS), and Pending Interest Table (PIT). The Upper figure shows the functions performed when an Interest Message is received by a router: It will first check the Content Store. If the requested content is available in CS, it will return the Data Object to the previous node.

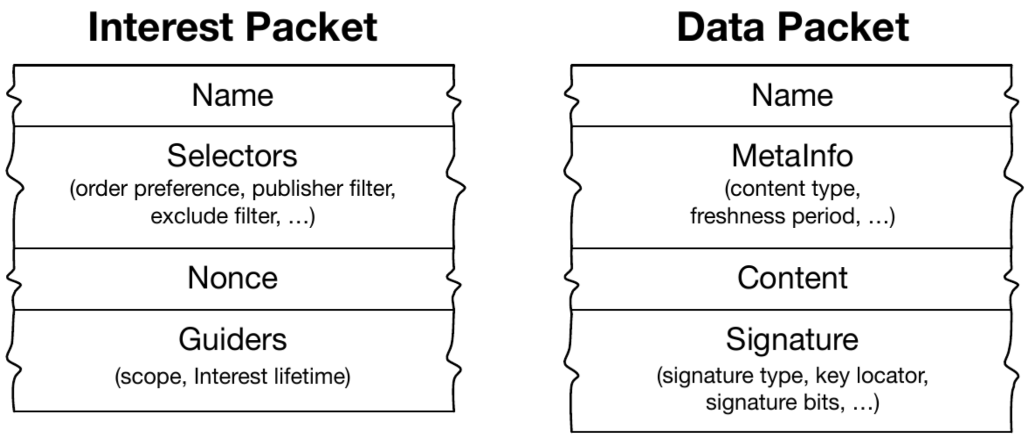

Interest Packet
Name
Selectors
(order preference, publisher filter, exclude filter, ...)
Nonce
Guiders
(scope, Interest lifetime)
Data Packet
Name
MetaInfo
(content type, freshness period, ...)
Content
Signature
(signature type, key locator, signature bits, ...)

Figure 2.4: Interest Packet and Data Packet format [18]

If CS has not cached the Data Objects, router will check the PIT table for the requested Name. If no match found, it will create a PIT entry which includes the incoming interface, Nounce, and Name of the interest. But if an entry found with the same Name, it will check the related nonces. If nonces match too, router will detect a loop and discard the Interest Message. If nonces do not match, router will aggregate the Interest Message and add the incoming face and nonce to the PIT entry.

In case, no PIT entry found for that name prefix, as mentioned previously, router will create a PIT entry for the Interest and then will check the FIB table by performing a Longest Prefix Match and find the next hop for the interest message. Interest message will be forwarded toward the producer using Interest Forwarding Strategy.

The second part of figure 2.5 shows the propagation of data back to the consumer node. When a Data Object is received by a router, it will first check the Pending Interest Table and satisfy the pending interest entry. If there is none, it will discard the data object. Then it will add the Data Object to the Content Store to be used by future requests.

In the rest of the thesis, we use the term name data object (NDO) or content object interchangeably, and use the term neighbor instead of interface or face. We denote the name of NDO j by $n(j)$, and the name prefix that includes that NDO name by $n(j)^*$. We denote the existence of an entry for a prefix $n(j)^*$ or NDO with name $n(j)$ in the FIB, PIT or CS of router i by $n(j)^* \in FIB^i$, $n(j) \in PIT^i$, and $n(j) \in CS^i$, respectively.

We use $I[n(j), id_j(s)]$ to denote an Interest that requests NDO with name $n(j)$ and that is originated by consumer s, who assigns nonce $id_j(s)$ to the Interest. A content-object message (or NDO message) sent in response to an Interest $I[n(j), id_j(s)]$, denoted $D[n(j), id_j(s), sig(j)]$, states the name and nonce of the Interest, a signature payload $sig(j)$ used to validate the content object, and the object itself.

The entry in FIB^i for name prefix $n(j)^*$ is denoted by $FIB^i_{n(j)^*}$ and consists of $n(j)^*$ and the list of neighbors that can be used to reach the NDO. If neighbor k is listed in $FIB^i_{n(j)^*}$, then we state $k \in FIB^i_{n(j)^*}$. In NDN [24], the FIB entry for an NDO also contains a stale time after which the entry could be deleted; the round-trip time through the neighbor; a rate limit; and status information stating whether it is known or unknown that the neighbor can bring data back, or is known that the neighbor cannot bring data back.

The entry in PIT^i for NDO with name $n(j)$ is denoted by $PI^i_{n(j)}$ and consists of a vector of one or multiple tuples, one for each nonce processed for the same NDO name. The tuple for a given NDO states the nonce used, the incoming and the outgoing neighbor(s). The tuple created as a result of processing Interest $I[n(j), id_j(s)]$ received from k and forwarded to a set of neighbors $OUTSET$ is denoted by $PI^i_{n(j)}[id_j(s), in : k, out :$

$OUTSET]$, and the set of outgoing neighbors in $PI^i_{n(j)}$ is denoted by $OUTSET(PI^i_{n(j)})$.

Each PIT entry $PI^i_{n(j)}[id_j(s), in : k, out : OUTSET]$ has a lifetime, which should be larger than the estimated round-trip time to a site where the requested NDO can be found.

We denote by $NI[n(j), id_j(s), \mathsf{CODE}]$ the NACK sent in response to $I[n(j), id_j(s)]$, where CODE states the reason why the NACK is sent.

Algorithms 1 and 2 illustrate the NDN Interest processing approach [23, 24] using the notation we have introduced, and correspond to Interest-processing and forwarding-strategy algorithms in [24]. Algorithm 2 does not include the probing of neighbors proposed in NDN, given that this aspect of NDN is still being defined [24]. Routers forward NACKs received from those neighbors to whom they sent Interests, unless the PIT entries have expired or do not match the information provided in the NACKs. The NDN forwarding strategy augments the original CCN strategy by introducing negative acknowledgements (NACK) sent in response to Interests for a number of reasons, including: routers identifying congestion, routers not having routes in their FIBs to the requested content, or Interest loops being detected. Algorithms 1 and 2 indicate the use of NACKs that is not part of the original CCN design by "**[NDN]**."

A number of ICN (information-centric networking) architectures that are not based on Interests adopt a push-based approach to multicasting using mechanisms that are much the same as those introduced for PIM-SM [106] for the IP Internet. A good example of this case is COPSS [105]. Users subscribe to content on a content descriptor (CD), which can be any legal content name, and each CD is associated with a Rendezvous Point (RP).

Algorithm 1 NDN Processing of Interest at router i

```
1: function Process Interest
2: INPUT: $PIT^i$, $CS^i$, $FIB^i$;
3: INPUT: $I[n(j), id_j(s)]$ received from $k$;
4: if $n(j) \in CS^i$ then
5:     send $D[n(j), id_j(s), sig(j)]$ to $k$
6: else
7:     if $n(j) \notin PIT^i$ then
8:         create $PI^i_{n(j)}[id_j(s), in : k, out : \emptyset]$;
           call Forwarding Strategy($PI^i_{n(j)}$)
9:     else
10:        % There is a PIT entry for $n(j)$
11:        if $\exists\ PI^i_{n(j)}[id_j(x)]$ with $id_j(x) = id_j(s)$ then
12:            % A duplicate Interest is detected
               [NDN] send $NI[n(j), id_j(s), \mathsf{duplicate}]$ to $k$;
               drop $I[n(j), id_j(s)]$
13:        else
14:            % Interest can be aggregated
               create $PI^i_{n(j)}[id_j(s), in : k, out : \emptyset]$;
15:            if $RT_i(I[n(j), id_j(s)])$ is exprired then
16:                call Forwarding Strategy($PI^i_{n(j)}$);
17:            end if
18:        end if
19:    end if
20: end if
```

The number of RPs may be as large as the number of ICN nodes. Routers maintain CD-based subscription tables to provide the same functionality as IP multicast, and COPSS supports sparse-mode multicasting at the content layer. The RP's receive content from one or more publishers and send it over the multicast trees established by routers for the multicast groups.

A major selling point for maintaining per-Interest forwarding state using PITs in CCNx and NDN [4, 17] has been that it enables "native" support for multicasting in the

Algorithm 2 NDN forwarding of Interest at router i

```
 1: function Forwarding Strategy
 2: INPUT: $PIT^i$, $CS^i$, $FIB^i$;
 3: INPUT: $PI^i_{n(j)}[id_j(s), in : k, out : OUTSET]$
 4: if $n(j)^* \in FIB^i$ then
 5:     for each neighbor $m$ in $FIB^i_{n(j)^*}$ by rank do
 6:         if $m \neq in : k$ for all $in : k \in PI^i_{n(j)} \wedge$
              $m \notin SET$ for all $out : SET \in PI^i_{n(j)}$ then
 7:             if $m$ is available then
 8:                 $OUTSET(PI^i_{n(j)}) = OUTSET(PI^i_{n(j)}) \cup m$;
                    start $RT_i(I[n(j), id_j(s)])$;
                    forward $I[n(j), id_j(s)]$ to neighbor $m$;
                    return
 9:             end if
10:         end if
11:     end for
12:     [NDN] send $NI[n(j), id_j(s), \text{congestion}]$ to $k$;
        drop $I[n(j), id_j(s)]$; delete $PI^i_{n(j)}$
13: else
14:     send $NI[n(j), id_j(s), \text{no data}]$ to $k$;
        drop $I[n(j), id_j(s)]$; delete $PI^i_{n(j)}$
15: end if
```

data plane with no additional signaling required in the control plane. In short, multicast receivers simply send Interests towards the multicast source. As Interests from receivers and previous-hop routers are aggregated in the PITs on their way to the multicast source, a multicast forwarding tree (MFT) is formed and maintained in the data plane. Multicast Interest are forwarded using the same forwarding information base (FIB) entries used for unicast traffic, and multicast data packets are sent using reverse path forwarding (RPF) over the paths traversed by aggregated Interests. If Interests can be forwarded without incurring loops, then MFTs can be created and multicast data can be disseminated without requiring the use of complex multicast routing protocols operating in the control plane (e.g.,

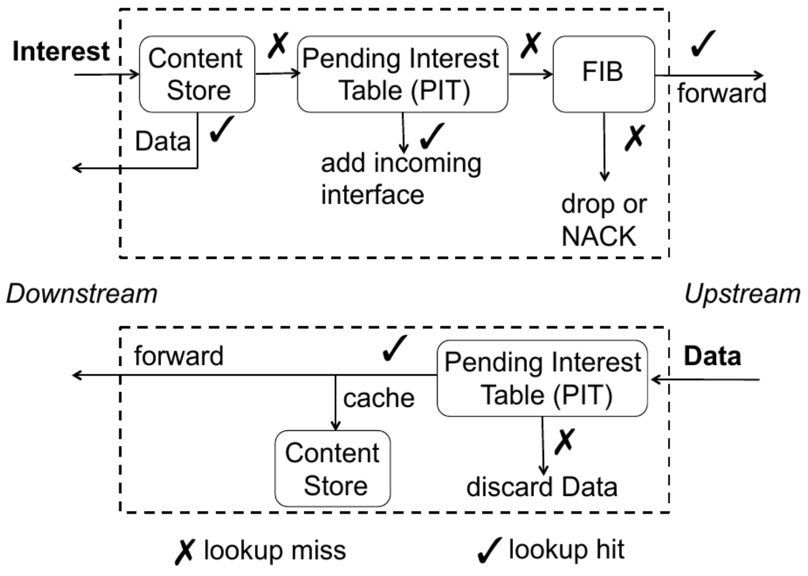


Figure 2.5: Forwarding Strategy in Named Data Networking [18]

[106, 110]. Using PITs is appealing in this context; however, we show that native support of multicasting in the data plane can be easily done using anonymous datagrams.

2.5 Mobile Ad hoc Routing Protocols

Many Mobile Ad hoc Networks (MANET) routing protocols have been proposed since the introduction of the routing protocol for the DARPA packet-radio network [88] and excellent surveys and comparative studies of this prior work have been presented over the years [79, 80, 87, 90, 94, 96, 99, 103].

Optimized Link State Routing (OLSR) is the best-known example of proactive routing for MANETs [82]. It uses HELLO messages to maintain neighbor connectivity, and Topology Control (TC) messages to disseminate link-state information throughout the network. To reduce signaling overhead, OLSR takes advantage of connected dominating sets. Some nodes are elected as multipoint relays (MPRs) and only MPRs forward TC messages, and only link-state information needed to connect MPRs is advertised in the network.

Ad hoc On-Demand Distance Vector (AODV) [94] is the most popular example of the on-demand routing approach. To find a route to an intended destination, a source broadcasts a Route Request (RREQ) stating the source and destination nodes, the most recent sequence number known for each, a a broadcast ID, and a hop count to the source. A router that forwards a RREQ for the first time creates a record for the RREQ stating the source and broadcast-ID pair of the RREQ; and a a reverse route to the source of the RREQ stating the next hop and hop count to the source, and the sequence number of the source. It maintains any RREQ record and reverse route for a finite time. A router discards any received RREQ that states a source and broadcast-ID pair for which it has a RREQ record.

The intended destination or a router with a valid route to the destination responds to the RREQ by sending a Route Reply (RREP) over the reverse route from which the RREQ was received. The RREP states the destination and the source of the RREQ, the destination sequence number, and the hop count to the destination. A router receiving a RREP establishes a route record to the destination stating the destination sequence number, the next hop to it, and the neighbors using the route (precursors). A router forwards only the first copy of a RREP (based on the destination sequence number) and increments by one the hop count to the destination when it forwards a RREP.

Link failures can be recognized in AODV by the absence of HELLO messages sent periodically between neighbors. When a node detects a link failure, it sends a Route Error (RERR) to all neighbor nodes that are precursors of a route that is broken because of the link failure. Nodes receiving a RERR message invalidate all routes that were using the failed link and propagate the RERR message to their precursor nodes.

Hybrid routing protocols attempt to reduce the signaling overhead of proactive and on-demand schemes by combining the two. This has been done by either using clusters within which routes to destinations are maintained proactively and using on-demand routing across clusters (e.g., ZRP [94]), or by maintaining routes to certain destinations proactively and using on-demand routing for the rest of destinations [98].

Interestingly, all prior approaches proposed for on-demand and hybrid routing have assumed that a router that receives route-requests (RREQ) regarding destinations for which it does not have valid routes forwards *each* new RREQ it receives, and replicas of the same RREQ are silently dropped. This constitutes a major performance limitation for all on-demand and hybrid routing schemes proposed to date. Intuitively, as the number of destinations increase, the failure of just a few links may cause many sources to engage in the discovery of new routes to those destinations, with each source flooding RREQs. Because a router forwards each RREQ it receives as long as it does not state the same source and request ID pair, the flooding of RREQs grows linearly with the number of sources, even if the sources are seeking the same few destinations.

The following section describes our approach to address this problem crated by too many RREQs. We use a specific protocol as an example of the basic approach.

Chapter 3

Enabling Correct Forwarding and Retransmission in ICNs

3.1 Undetected Interest Loops in CCN an NDN

The use of nonces in NDN and the original CCN approach can be extrapolated to include the case in which an Interest states a nonce and the path traversed by the Interest by assuming that $id_j(s)$ equals the tuple $(id_j(s)[nonce], id_j(s)[path])$. If a nonce and path traversed by the Interest are used, deciding whether an Interest has not traversed a loop can be based on whether $id_j(x)[nonce] \neq id_j(s)[nonce] \vee i \notin id_j(s)[path]$. However, including path information in Interests reveals the identity of originators of Interests.

The key aspect of the forwarding strategies that have been proposed for NDN and CCN is that a router determines whether or not an Interest is a duplicate Interest based solely on the content name and Interest-identification data for the Interest (a nonce in NDN's case). To discuss the correctness of the forwarding strategy and other strategies, we define an Interest loop as follows.

Interest Loop: An Interest loop of h hops for NDO with name $n(j)$ occurs when one or more Interests asking for $n(j)$ are forwarded and aggregated by routers along a cycle $L = \{v_1, v_2, ..., v_h, v_1\}$ such that router v_k receives an Interest for NDO $n(j)$ from v_{k-1} while waiting for a response to the Interest it has forwarded to v_{k+1} for the same NDO, with $1 \leq k \leq h$, $v_{h+1} = v_1$, and $v_0 = v_h$.

According to the NDN forwarding strategy, a router can select a neighbor to forward an Interest if it is known that it can bring content and its performance is ranked higher than other neighbors that can also bring content. The ranking of neighbors is done by a router independently of other routers, which can result in long-term routing loops implied by the FIBs if the routing protocol used in the control plane does not guarantee instantaneous loop freedom (e.g., NLSR [125]).

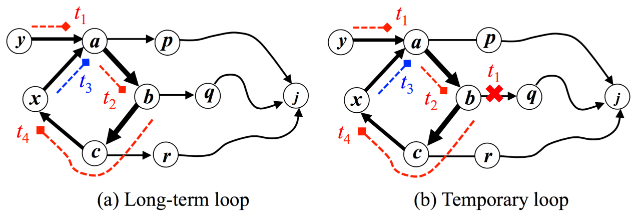


Figure 3.1: Undetected Interest loops in NDN and CCN

Figure 3.1 illustrates Interest looping in NDN. Arrowheads in the figure indicate the next hops to content advertised by router j according to the FIB entries stored in routers. Thick lines indicate that the perceived performance of a neighbor is better than neighbors shown with thinner lines. Dashed lines indicate the traversal of Interests over links and paths. The time when an event is processed at a router is indicated by t_i. Figure 3.1(a) shows the case of a long-term Interest loop formed because the multi-paths implied in FIBs are not loop-free, even though all routing tables are consistent. Figure 3.1(b) shows the case of a temporary Interest loop when single-path routing is used and FIBs are inconsistent due to a topology change at time t_1 (link (b, q) fails). In both cases, router a aggregates the Interest from x at time t_3, router x aggregates the Interest from c at time t_4, and the combined steps preclude the detection of Interest looping. This results in x and y having to wait for their Interests to time out, before they can retransmit. Furthermore, there is no guarantee that their retransmissions will elicit a response (content or NACK).

As Theorem 7 proves, the CCN and NDN forwarding strategies specified in [14, 24, 27] cannot ensure that Interest loops are detected when Interests are aggregated, even if nonces were to denote Interests uniquely. The theorem assumes that all messages are sent correctly and that no routing-table changes occur to show that the NDN forwarding strategy can fail to return any content or NACK in response to Interests independently of network dynamics. Furthermore, Theorem 2 shows that no forwarding strategy can be correct if it allows Interest aggregation and attempts Interest-loop detection by the matching of Interest-identification data.

Theorem 1. *Interest loops can go undetected in a stable, error-free network in which NDN or CCN is used, even if nonces were to denote Interests uniquely.*

Proof. Consider the NDN or CCN forwarding strategy running in a network in which no two nonces created by different nodes for the same content are equal, all transmissions are received correctly, and no topology or routing-table changes occur after time t_0. Let

$LT^{v_k}(I[n(j), id_j(s)])$ denote the lifetime of $I[n(j), id_j(s)]$ at router v_k.

Assume that Interests may traverse loops when they are forwarded according to the forwarding strategy, and let a loop $L = \{v_1, v_2, ..., v_h, v_1\}$ exist for NDO j, and let Interest $I[n(j), id_j(x)]$ start traversing the chain of nodes $\{v_1,\ v_2, ...,\ v_k\} \in L$ (with $1 < k < h$) at time $t_1 > t_0$.

Assume that $I[n(j), id_j(x)]$ reaches router v_k at time $t_3 > t_1$ and that router v_k forwards Interest $I[n(j), id_j(y)]$ to its next hop $v_{k+1} \in L$ at time t_2, where $t_1 \leq t_2 < t_3$, $id_j(x) \neq id_j(y)$, and v_{k+1} may be v_1.

According to the Interest processing strategy in NDN and CCN, router v_k creates an entry in its PIT for $I[n(j), id_j(y)]$ at time t_2, and perceives any Interest for name $n(j)$ and a nonce different than $id_j(y)$ received after time t_2, and before its PIT entry for $I[n(j), id_j(y)]$ is erased, as a subsequent Interest.

Let $|t_2 - t_3| < LT^{v_k}(I[n(j), id_j(y)])$ when router v_k receives $I[n(j), id_j(x)]$ from router $v_{k-1} \in L$ at time t_3, where $1 < k - 1$. According to the Interest processing strategy in NDN and CCN, router v_k must treat $I[n(j), id_j(x)]$ as a subsequent Interest for content $n(j)$ that is aggregated, because v_k is waiting for $D[n(j), id_j(y)]$ at time t_3.

Because of the existence of L, Interest $I[n(j), id_j(y)]$ must be forwarded from v_k to v_1. Let t_4 denote the time when $I[n(j), id_j(y)]$ reaches v_1, where $t_4 > t_2 \geq t_1$, and assume that $|t_1 - t_4| < LT^{v_1}(I[n(j), id_j(x)])$. According to NDN's Interest processing strategy, v_1 must treat $I[n(j), id_j(y)]$ as a subsequent Interest, because it is waiting for $D[n(j), id_j(x)]$ at time t_4.

Given the Interest aggregation carried out by nodes v_k and v_1, nodes in the chain

$\{v_1, v_2, ..., v_{k-1}\} \in L$ process only $I[n(j), id_j(x)]$, nodes in the chain $\{v_{k+1}, v_{k+2}, ..., v_h\} \in L$ process only $I[n(j), id_j(y)]$, and no Interest loop detection can take place. Therefore, no content can be submitted in response to $I[n(j), id_j(x)]$ and $I[n(j), id_j(y)]$. □

Similar results to Theorem 1 can be proven for NDN and the original CCN operating in a network in which routing tables are inconsistent as a result of network or content dynamics. In this case, Interest loops can go undetected even if the control plane supports only single-path forwarding of Interests.

Theorem 2. *No correct forwarding strategy exists with Interest aggregation and Interest loop detection based on the matching of Interest-identification data.*

Proof. Assume any forwarding strategy in which a router remembers an Interest it has forwarded as long as necessary to detect Interest loops, and detects the occurrence of an Interest loop by matching the Interest-identification data carried in an Interest it receives with the Interest-identification data used in the Interest it forwarded previously asking for the same content. Let $I[n(j), id_j(s)]$ denote the Interest asking for $n(j)$ with Interest-identification data $id_j(s)$ created by router s.

Assume that an Interest loop $L = \{v_1, v_2, ..., v_h, v_1\}$ for NDO with name $n(j)$ exists in a network using the forwarding strategy. Let Interest $I[n(j), id_j(x)]$ start traversing the chain of nodes $\{v_1, v_2, ..., v_k\} \in L$ (with $1 < k < h$) at time t_1.

Assume that $I[n(j), id_j(x)]$ reaches router v_k at time $t_3 > t_1$ and that router v_k forwards Interest $I[n(j), id_j(y)]$ to its next hop $v_{k+1} \in L$ at time t_2, where $t_1 \leq t_2 < t_3$, $id_j(x) \neq id_j(y)$. Let $I[n(j), id_j(y)]$ traverse the chain of nodes $\{v_k, v_{k+1}, ..., v_1\} \in L$, reaching v_1 at time t_4, where $t_4 > t_2 \geq t_1$.

By assumption, Interest aggregation occurs, and hence v_k aggregates $I[n(j), id_j(x)]$ at time t_3, and v_1 aggregates $I[n(j), id_j(y)]$ at time t_4. Therefore, independently of the amount of information contained in $id_j(x)$ and $id_j(y)$, v_1 cannot receive $I[n(j), id_j(x)]$ from v_h and v_k cannot receive $I[n(j), id_j(y)]$ from v_{k-1}. It thus follows that no node in L can successfully use the matching of Interest-identification data to detect that Interests for $n(j)$ are being sent and aggregated along L and the theorem is true. $\square$

The results in Theorems 1 and 2 can also be proven by mapping the Interest processing strategy of NDN, and any forwarding strategy that attempts to detect Interest loops by matching Interest-identification data, to the problem of distributed termination detection over a cycle, where Interests serve as the tokens of the algorithm [7, 16]. Because Interest aggregation erases a token traversing the ring (Interest loop) when any node in the ring has previously created a different token, correct termination detection over the ring (i.e., Interest loop detection) cannot be guaranteed in the presence of Interest aggregation.

Obviously, a loop traversed by an Interest can be detected easily if each Interest is identified with the route it should traverse. This is easy to implement but requires routers in the network to have complete topology information (e.g., [125, 19, 21]) or at least path information or partial topology information (e.g., [3, 19]). Similarly, carrying the path traversed by an Interest in its header also ensures that an Interest loop is detected if it occurs. In these two cases, however, there is no need for using nonces to detect Interest loops. More importantly, path information reveals the identity of the source router requesting content and hence defeats one of the key objectives of the NDN and CCN forwarding strategies.

Another view of the problem would be to say that Interest aggregation is not common and hence undetected Interest loops should be too rare to cause major performance problems. However, if Interests need not be aggregated, then very different architectures could be designed for content-centric networking that do not require using PITs.

3.2 SIFAH

3.2.1 Design Rationale

It is clear from the results in the previous section that using nonces or identifying Interests uniquely is useless for Interest-loop detection when Interests are aggregated, and that source routing of Interests or including the path traversed by an Interest are not desirable. Accordingly, for an Interest forwarding strategy to be correct in the presence of Interest aggregation, it must be the case that, independently of the identity of an Interest or how Interests for the same content are aggregated, at least one router detects that it is traversing a path that is not getting the Interest closer to a node that has advertised the requested content.

Ensuring that at least one router in an Interest loop detects the incorrect forwarding of the Interest can be attained if Interests were to carry any type of ordering information that cannot be erased by the use of Interest aggregation. Fortunately, distance information for advertised name prefixes is exactly this type of ordering information.

Given that forwarding information bases (FIB) are populated from the routing tables maintained in the control plane of a network, they constitute a readily-available tool to establish the proper interaction between the forwarding strategy operating in the data plane and the distances to advertised content prefixes maintained by the routing protocol operating in the control plane. This is the basis of the *Strategy for Interest Forwarding and Aggregation with Hop-Counts* (SIFAH).

3.2.2 Information Stored and Exchanged

A router maintains a FIB, a PIT, and an optional content store. FIB^i is indexed using content name prefixes. The FIB entry for prefix $n(j)^*$ is denoted by $FIB^i_{n(j)^*}$, and consists of a list of one or more tuples. Each tuple states a next hop to $n(j)^*$ and a hop count to the prefix. The set of next hops to $n(j)^*$ listed in $FIB^i_{n(j)^*}$ is denoted by $S^i_{n(j)^*}$. The hop count to $n(j)^*$ through neighbor $q \in S^i_{n(j)^*}$ is denoted by $h(i, n(j)^*, q)$.

An Interest sent by node k requesting NDO $n(j)$ is denoted by $I[n(j), h^I(k)]$, and states the name $n(j)$, and the hop count $(h^I(k))$ from node k to the name prefix $n(j)^*$ that is the best match for NDO name $n(j)$ when k forwards the Interest.

An NDO message sent in response to the Interest $I[n(j),\ h^I(k)]$ is denoted by $D[n(j), sig(j)]$, and states the name of the Interest, a signature payload $sig(j)$ used to validate the content object, and the object itself.

The NACK sent by router i in response to an Interest is denoted by $NI[n(j), \mathsf{CODE}]$ where CODE states the reason why the NACK is sent. Possible reasons for sending a NACK include: (a) an Interest loop is detected, (b) a route failed towards the requested content, (c) no content is found, and (d) the PIT entry expired.

PIT^i is indexed using NDO names. $PI^i_{n(j)}$ denotes the entry created in PIT^i for NDO with name $n(j)$, and specifies: the name of the NDO; the hop count $h^I(i)$ assumed by router i when it forwards Interest $I[n(j), h^I(i)]$; the set of incoming neighbors from which Interests for $n(j)$ are received $(INSET(PI^i_{n(j)}))$; the set of outgoing neighbor(s)

$(OUTSET(PI^i_{n(j)}))$ to whom router i forwards its Interest; and the remaining lifetime for the Interest $(RT(PI^i_{n(j)}))$.

3.2.3 Interest Loop Detection

To define a correct forwarding strategy, special attention must be paid to the fact that updates made to the FIBs stored at routers occur independently of and concurrently with the updates made to their PITs. For example, once a router has forwarded an Interest that assumed a given distance to content prefix $n(i)^*$ and waits for its Interest to return a data object, its distance to the same content may change based on updated to its FIB. Hence, simply comparing the minimum distance from a router to content against a distance to content stated in an Interest is not enough to ensure that Interests are not incorrectly forwarded to routers that are farther away form the requested content.

SIFAH takes into account the fact that FIBs and PITs are updated independently by requiring that a router that forwards an Interest for a given piece of content remembers in its PIT entry the value of the distance to content assumed when it issues its Interest. The following rule is then used for a given router to determine whether an Interest may be propagating over an Interest loop.

The number of hops to requested content is used as the metric for the invariant condition. This is done for two reasons, storing hop-count distances in the FIB incurs less storage overhead than storing complex distance values, and the next hops to a prefix stored in the FIB can be ranked based on the actual distances to content.

HFAR–Hop-Count Forwarding with Aggregation Rule: Router i can accept $I[n(j), h^I(k)]$ from router k if one of the following two conditions is satisfied:

1. $n(j) \notin PIT^i \wedge \exists\, v(\ v \in S^i_{n(j)^*} \wedge h^I(k) > h(i, n(j)^*, v)\ )$

2. $n(j) \in PIT^i \wedge h^I(k) > h^I(i)$

The first condition ensures that router i accepts an Interest from neighbor k only if i determines that is closer to $n(j)^*$ through at least one neighbor than k was when it sent its Interest. The second condition ensures that router i accepts an Interest from neighbor k only if i was closer to $n(j)^*$ than k when i and k sent their Interests.

Section 7.1 proves that using HFAR is *sufficient* to ensure that an Interest loop cannot occur without a router in the loop detecting that the Interest has been forwarded incorrectly. This result is independent of whether Interests are aggregated or sent over one or multiple paths, or how Interests are retransmitted.

Similar forwarding rules based on more sophisticated lexicographic orderings could be defined based on the same general approach stated in HFAR. The requirement for such forwarding rules is that more information needs to be maintained in the FIBs, such as distance values to name prefixes that take into account such factors as end-to-end delay, reliability, cost, or bandwidth available.

HFAR is very similar to sufficient conditions for loop-free routing introduced in the past, in particular sufficient conditions for loop-free routing based on diffusing computations [9, 21, 26]. Indeed, the approach we introduce for Interest-loop detection in SIFAH can be viewed as a case of termination detection based on diffusing computations [6].

It should be pointed out that, because HFAR is not *necessary* to detect loops, there are cases in which HFAR is not satisfied even though no Interest loops exist. However, prior results on multi-path routing based on diffusing computations [25] indicate that this does not constitute a performance problem. Given that FIBs are updated to reflect correct hop counts, or correct complex distance values in general, a sufficient condition for loop detection operating with multi-path routing is a good baseline for an Interest-based forwarding strategy.

3.2.4 SIFAH Operation

Algorithms 3 to 8 specify the steps taken by routers to process Interests, forward Interests, return NDOs, process perceived link failures, handle Interest-lifetime expirations, and send NACKs according to SIFAH. Optional steps and data in algorithms are indicated by "[**o**]".

The algorithms used to describe SIFAH were not designed to take into account such issues as load balancing of available paths, congestion-control, or the forwarding of an Interest over multiple concurrent paths. For simplicity, it is assumed that all Interest retransmissions are carried out on an end-to-end basis (i.e., by the consumers of content) rather than routers. Hence, routers do not attempt to provide any "local repair" when a neighbor fails or a NACK to an Interest is received; the origin of an Interest is in charge of retransmitting it after receiving a NACK for any reason. Interest retransmissions could also be done by routers. The design and analysis of Interest retransmission strategies implemented by routers or by content consumers is a topic deserving further study.

Algorithm 3 implements HFAR. Router i determines that an Interest can be forwarded because Condition 1 in HFAR is satisfied (Line 9 of Algorithm 3), or an Interest can be aggregated because Condition 2 of HFAR is satisfied (Line 17 of Algorithm 3).

Content requests from local content consumers are sent to the router in the form of Interests stating infinite hop counts to content, and each router knows which neighbors are remote and which are local.

The Maximum Interest Life-time (MIL) assumed by a router before it deletes an Interest from its PIT should be large enough to preclude an excessive number of retransmissions. On the other hand, MIL should not be too large to cause the PITs to store too many Interests for which no NDO messages or NACKs will be sent due to failures or transmission errors. A few seconds would be a viable value for MIL. In practice, however, the consumer submitting an Interest to its local router could provide an initial value for the Interest lifetime estimated over a number of Interests submitted for NDOs in the same NDO group corresponding to a large piece of content (e.g., a movie). This is specially the case given our assumption that Interest retransmissions are carried out by content consumers, rather than by routers.

Algorithm 4 describes a simple forwarding strategy in which router i simply selects the first neighbor v in the ranked list of neighbors stored in the FIB for prefix $n(j)^*$ that satisfies the first condition in HFAR (Line 4 of the algorithm). More sophisticated strategies can be devised that attain load balancing among multiple available routes towards content and can be close to optimum (e.g., [21]). In addition, the same Interest could be forwarded over multiple paths concurrently, in which case content could be sent back over some or all the paths that the Interest traversed successfully. To be effective, however, these approaches should require the adoption of a loop-free multi-path routing protocol in the control plane (e.g., [10, 12]). In this context, the control plane establishes valid multi-paths to content prefixes using long-term performance measures, and the data plane exploits those paths using HFAR and short-term performance measurements, without risking the long delays associated with backtracking due to looping.

Algorithm 5 outlines the processing of NDO messages received in response to Interests. A router accepts an NDO received from a neighbor if it has a PIT entry waiting for the content and the NDO message came from one of the neighbors over which the Interest was sent (Line 5 of the algorithm). The router forwards the valid NDO to any neighbor that requested it and deletes the corresponding PIT entry. A router stores an NDO it receives optionally (Step 7 of Algorithm 5). The caching of NDOs is done according to the caching strategy used in the network, which can be path-based or edge-based [5], for example. However, SIFAH works independently of the caching strategy adopted in the network.

Algorithm 6 shows a simple approach to handle the case when a PIT entry expires with no NDO or NACK being received. Given that routers do not initiate Interest retransmissions, router i simply sends NACKs to all neighbors from which it received Interests for $n(j)$. A more sophisticated approach would be needed for the case in which routers must provide Interest retransmissions in a way similar to on-demand routing protocols that support local repair of route requests.

Algorithm 7 states the steps taken to handle NACKs. Router i forwards the NACK it receives for $n(j)$ to all those neighbors from whom it received Interests for $n(j)$ and deletes the Interest entry after that. Supporting Interest retransmissions by routers would require a more complex approach for the handling of NACKs.

Algorithm 8 lists the steps taken by a router in response to the failure of connectivity with a neighbor. Reacting to the failure of perceived connectivity with a neighbor over which Interests have been forwarded could be simply to wait for the life-times of those Interests to expire. However, such an approach can be very slow reacting to link failures compared to using Algorithm 8. The algorithm assumes that the control plane updates FIB^i to reflect any changes in hop counts to name prefixes resulting from the loss of connectivity to one or more neighbors. For each Interest that was forwarded over the failed link, router i sends a NACK to all neighbors whose Interests were aggregated.

3.2.5 Examples of SIFAH Operation

Figures 4.5(a) to (d) illustrate how SIFAH operates using the same example used in Figure 3.1. Figures 4.5(a) and (b) address the case in which the control plane establishes multiple paths to each name prefix but does not guarantee loop-free routing tables. Figures 4.5(c) and (d) illustrate how SIFAH operates when single-path routing is used.

The pair of numbers next to each link outgoing from a node in Figure 4.5(a) indicates the hop count to $n(j)$ through a neighbor and the ranking of the neighbor in the FIB. The example assumes that: (a) routers execute a routing protocol that does not enforce loop-free FIBs; and (b) the ranking of neighbors is determined independently at each router using some data-plane strategy based on the perceived performance of each path and interface. It should be noted that the distance value of a path need not be directly proportional to the hop-count value of the path shown in the figure.

Let the tuple (v: h, r) indicate a neighbor, its hop count and its ranking. In Figure 4.5(a), FIB^a lists (b: 7, 1), (p: 7, 2), and (x: 9, 3), which is shown in green font. Similarly, FIB^y states (a: 8, 1); FIB^b states (c: 10, 2), (a: 8, 1), and (q: 6, 3); FIB^c

states (b: 7, 1), (x: 9, 2), and (r: 9, 3); and FIB^x states (a: 8, 1) and (c: 8, 2). Some of the FIB entries for p, q and r are shown in black font.

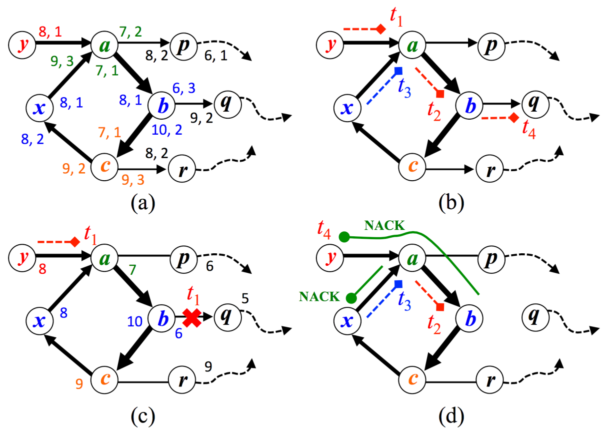


Figure 3.2: Interest looping is avoided or detected with SIFAH

In Figure 4.5(b), router y originates an Interest for $n(j)$ and sends $I[n(j), h^I(y) = 8]$ to a. Router a receives the Interest from router y at time t_1 and, given that $8 = h^I(y) > h(a, n(j)^*, b) = 7$, it accepts the Interest because it has at least one neighbor that satisfies HFAR. Router a sends $I[n(j), h^I(a) = 7]$ to b because it is the highest-ranked neighbor satisfying HFAR. Router a aggregates $I[n(j),\ h^I(x) = 8]$ at time $t_3 > t_1$, because it sent $I[n(j),\ h^I(a) = 7]$ at time t_1 and $8 = h^I(x) > h^I(a) = 7$. Router b receives the Interest from a at time $t_2 > t_1$; accepts it because it has at least one neighbor that satisfies HFAR ($7 = h^I(a) > h(b, n(j)^*, q) = 6$); and sends $I[n(j), h^I(b) = 6]$ to q because q is the highest-ranked neighbor of b that satisfies HFAR. This is an example that Interests are forwarded along loop-free paths if SIFAH is used and the FIBs maintained by routers have consistent information, even if some of the multi-paths implied in the FIBs involve loops. The next section proves this result in the general case.

Figure 4.5(c) shows the hop count values stored in the FIBs for name prefix $n(j)$

when single-path routing is used. Each router has a single next hop and one hop count for each prefix listed in its FIB. Router b updates its FIB to reflect the failure of link (b, q) at time t_1, while router y sends an Interest to router a requesting $n(j)$. Routers have inconsistent FIB states for $n(j)$ while routing updates propagate and Interests are being forwarded.

As shown in Figure 4.5(d), router b *must* send $NI[n(j), \mathsf{loop}]$ to a, because $7 = h^I(a) \not> h(b, n(j)^*, c) = 10$ and HFAR is not satisfied. In turn, when a receives the NACK from b, it must forward $NI[n(j), \mathsf{loop}]$ to y and to x. Eventually, the routing protocol running in the control plane makes routers a and y change the hop count to $n(j)^*$ in their FIBs to reflect the failure of link (b, q). At that point, a retransmission of the Interest from y would state $h^I(y) = 9$ and would make a forward $I[n(j), h^I(a) = 8]$ to p.

3.3 Correctness of SIFAH

The following theorems show that SIFAH enforces correct Interest forwarding and aggregation, and constitutes a safe Interest forwarding strategy. The results are independent of whether the network is static or dynamic, the specific caching strategy used in the network (e.g., at the edge or along paths traversed by NDO messages [5]), or the retransmission strategy used by content consumers after experiencing g a timeout or receiving a NACK from attached routers. SIFAH ensures that Interests cannot be incorrectly propagated and aggregated along loops without meeting routers that detect the incorrect forwarding and hence send NACKs in return.

Theorem 3. *Interest loops cannot occur and be undetected in a network in which SIFAH* *is used.*

Proof. Consider a network in which SIFAH is used. Assume for the sake of contradiction that nodes in a loop L of h hops $\{v_1, v_2, ..., v_h, v_1\}$ send and possibly aggregate Interests for $n(j)$ along L, with no node in L detecting the incorrect forwarding of any of the Interests sent over the loop.

Given that L exists by assumption, $v_k \in L$ must send $I[n(j), h^I(v_k)]$ to node $v_{k+1} \in L$ for $1 \leq k \leq h-1$, and $v_h \in L$ must send $I[n(j), h^I(v_h)]$ to node $v_1 \in L$. For $1 \leq k \leq h-1$, let $h(v_k, n(j)^*)^L$ denote the value of $h^I(v_k)$ when node v_k sends $I[n(j), h^I(v_k)]$ to node v_{k+1}, with $h(v_k, n(j)^*)^L = h(v_k, n(j)^*, v_{k+1})$. Let $h(v_h, n(j)^*)^L$ denote the value of $h^I(v_h)$ when when node v_h sends $I[n(j), h^I(v_h)]$ to node $v_1 \in L$, with $h(v_h, n(j)^*)^L = h(v_h, n(j)^*, v_1)$.

Because no node in L detects the incorrect forwarding of an Interest, each node in L must aggregate the Interest it receives from the previous hop in L or it must send its own Interest as a result of the Interest it receives from the previous hop in L. This implies that $v_k \in L$ must accept $I[n(j), h^I(v_{k-1})]$ before $RT(PI^{v_k}_{n(j)})$ expires for $1 \leq k < h$, and $v_1 \in L$ must accept $I[n(j), h^I(v_h)]$ before $RT(PI^{v_1}_{n(j)})$ expires.

According to SIFAH, if v_k aggregates $I[n(j),\ h^I(v_{k-1})]$, then it must be true that $h^I(v_{k-1}) > h^I(v_k)$. Similarly, if v_1 aggregates $I[n(j), h^I(v_h)]$, then it must be the case that $h^I(v_h) > h^I(v_1)$.

On the other hand, if v_k sends $I[n(j), h^I(v_k)]$ to v_{k+1} as a result of receiving $I[n(j), h^I(v_{k-1})]$ from v_{k-1}, then it must be true that $h^I(v_{k-1}) > h(v_k, n(j)^*)^L = h^I(v_k)$ for $1 < k \leq h$. Similarly, if v_1 sends $I[n(j), h^I(v_1)]$ to v_2 as a result of receiving $I[n(j), h^I(v_h)]$

from v_h, then $h^I(v_h) > h(v_1, n(j)^*)^L = h^I(v_1)$.

It follows from the above argument that, for L to exist when each node in the loop follows SIFAH to send Interests asking for $n(j)$, it must be true that $h^I(v_h) > h^I(v_1)$ and $h^I(v_{k-1}) > h^I(v_k)$ for $1 < k \leq h$. However, this is a contradiction, because it implies that $h^I(v_k) > h^I(v_k)$ for $1 \leq k \leq h$. Therefore, the theorem is true. $\square$

The proof of Theorem 3 can be augmented to account for Interest forwarding strategies based on complex distance values rather than hop counts.

To be safe, an Interest forwarding strategy must ensure that either an NDO message with the requested content or a NACK is received within a finite time by the consumer who issues an Interest. The following theorem shows that this is the case for SIFAH, independently of the state of the topology or the fate of messages.

Theorem 4. *SIFAH ensures that an NDO message for name $n(j)$ or a NACK is received within a finite time by any consumer who issues an Interest for NDO with name $n(j)$.*

Proof. Consider $I[n(j), h^I(s)]$ being issued by consumer s at time t_1. The forwarding of Interests assumed in SIFAH is based on the best match of the requested NDO name with the prefixes advertised in the network. Furthermore, according to Algorithm 3, a router sends back an NDO message to a neighbor that sent an Interest for NDO $n(j)$ only if has an exact match of the name $n(j)$ in its content store. According to Algorithm 5, a router that receives an NDO message in response to an Interest it forwarded must forward the same NDO message. Hence, the wrong NDO message cannot be sent in response to an Interest. There are three cases to consider next: (a) there are no routes to the name prefix $n(j)^*$ of the requested NDO, (b) the Interest traverses an Interest loop, or (c) the Interest traverses a simple path towards a router d that can reply to the Interest.

Case 1: If there is no route to $n(j)^*$, then it follows from the operation of SIFAH (Algorithm 4) that a router issues a NACK stating that there is no route. That NACK is either forwarded successfully back to s or is lost due to errors or faults. In the latter case, it follows from Algorithms 6 and 8 that a router must send a NACK back towards s stating that the Interest expired or the route failed.

Case 2: If $I[n(j), h^I(s)]$ is forwarded along an Interest loop and does not reach any node with a copy of $n(j)$, then it follows from Theorem 5 that the Interest must either reach some router k that detects the incorrect forwarding of the Interest and must issue a NACK $NI[n(j), \mathsf{loop}]$ in response, or the Interest is dropped due to faults or transmission errors before reaching such router k.

If $NI[n(j), \mathsf{loop}]$ reaches a router k that detects the loop and issues $NI[n(j), \mathsf{loop}]$, then according to SIFAH (Algorithm 7), every router receiving the NACK $NI[n(j), \mathsf{loop}]$ originated by router k from the neighbor to whom the Interest was sent must relay the NACK towards s. Hence, if no errors or faults prevent the NACK from reaching s, the consumer receives a NACK stating that an Interest loop was found.

On the other hand, if either the Interest traversing an Interest loop or the NACK it induces at some router k is lost, it follows from Algorithms 6 and 8 that a router between s and router k must send a NACK towards s indicating that the Interest expired or that the route failed. Accordingly, consumer s must receive a NACK within a finite time after issuing its Interest in this case.

Case 3: If the Interest traverses a simple path towards a router d that advertises

$n(j)^*$ or has a content store containing $n(j)$, then the Interest must either reach d or not.

If the Interest is lost and does not reach d, then it follows from Algorithms 6 and 8 that a router between s and router d must send a NACK towards s indicating that the Interest expired or that the route failed. As a result, s must receive a NACK originated by some router between s and d.

If the Interest reaches d, then that router must either send the requested NDO back, or (in the case that d advertises $n(j)^*$ and $n(j)$ does not exist) issue a NACK stating that $n(j)$ does not exist. According to Algorithms 5 and 7, the NDO message or NACK originated by d is forwarded back towards s along the reversed simple path traversed by the Interest. If no fault or errors occur between d and s, it follows that the theorem is true for this case. Alternatively, if the NDO or NACK originated by d is lost due to faults or errors, it follows from Algorithms 6 and 8 that a router between s and router d must send a NACK towards s indicating that the Interest expired or that the route failed. $\square$

3.4 Performance Comparison

We compare SIFAH with NDN and the original CCN forwarding strategy in terms of the storage complexity of the approaches; the average time that a PIT entry remains in the PIT waiting for an NDO message or a NACK to be received in response, which we call PIT entry pending time; the end-to-end delay experienced by content consumers in receiving either the content they request or negative feedback; and the number of entries in the PITs maintained by content routers.

The storage complexity of each approach provides an indication of the storage overhead induced by the type of information required for routers to detect Interests loops. The simulation results we present on PIT entry pending times, end-to-end delays, and PIT sizes should be viewed simply as indications of the negative effects that undetected Interest loops have on the performance of NDN and CCN, and the fact that they can be completely avoided using SIFAH.

3.4.1 Storage Complexity

There is a large difference in the storage overhead incurred with the NDN forwarding strategy compared to SIFAH.

In SIFAH, router i uses only the value of $h^I(i)$ to determine whether the Interest it receives from k may be traversing an Interest loop, and does not store $h^I(k)$. Hence, the PIT storage size for SIFAH is

$$SS_{SIFAH} = O((INT + |mh|)|PIT^i|_{SIFAH})$$

where $|PIT^i|_{SIFAH}$ is the number of pending Interests in PIT^i when SIFAH is used, $|mh|$ is the number of bits used to store $h^I(i)$, and INT is the average storage required to maintain information about the incoming and outgoing neighbors for a given Interest. For a given NDO with name $n(j)$, the amount of storage needed to maintain the incoming and outgoing neighbors is

$$INSET(PI^i_{n(j)}) + OUTSET(PI^i_{n(j)}).$$

The NDN forwarding strategy requires each router to store the list of different nonces used to denote valid Interests for a given NDO name $n(j)$. With each nonce being of size $|id|$ and router i having up to I neighbors that send valid Interests for an NDO, the

PIT storage size for NDN is

$$SS_{NDN} = O((INT + |id|I)\ |PIT^i|_{NDN})$$

where $|PIT^i|_{NDN}$ is the number of pending Interests in PIT^i when NDN is used. Hence, even if $|PIT^i|_{NDN}$ is the same as $|PIT^i|_{SIFAH}$, the amount of additional PIT storage needed in NDN over SIFAH is

$$SS_{NDN} - SS_{SIFAH} \geq$$
$$(|id|I)(|PIT^i|_{NDN}) - (|mh|)(|PIT^i|_{NDN}).$$

A maximum hop count of 255 for an Interest is more than enough. Hence, with the size of a nonce in NDN of four bytes, the savings in PIT storage obtained with SIFAH compared to NDN is $(32I - 8)\ |PIT^i|_{NDN}$. This represents enormous savings of RAM in large networks. Furthermore, because the NDN forwarding strategy may not detect loops when Interests are aggregated, many Interest entries in PITs may have to be stored until their lifetimes expire. Accordingly, $|PIT^i|_{SIFAH}$ can be much smaller than $|PIT^i|_{NDN}$. This is confirmed by the simulation results presented subsequently.

The additional FIB storage overhead in SIFAH compared to the NDN forwarding strategy consists of storing the hop count information for each prefix $n(j)^*$ from each neighbor. This amounts to $(|mh|)(|FIB^i|)D^i$ at router i, where D^i is the number of neighbors of router i and $|FIB^i|$ is the number of entries in FIB^i. Given that D^i and I are of the same order and $O(|FIB^i|) < O(|PIT^i|)$, this is far smaller than the additional PIT storage needed by the NDN forwarding strategy compared to SIFAH.

3.4.2 Performance Impact of Undetected Interest Loops

Implementation of Forwarding Strategies in ndnSIM We implemented SIFAH in ndnSIM, an open-source NS-3 based simulator for Named Data Networks and Information Centric Networks [1]. Following the NDN architecture, ndnSIM is implemented as a new network-layer protocol model, which can run on top of any available link-layer protocol model, as well as on top of network-layer and transport-layer protocols.

We used the NDN implementation of its data plane from ndnSIM without any modifications. The ndnSIM NDN implementation is capable of detecting simple loops by matching nonces. The PIT entry expiration time for NDN is set to the default of one second. It should be pointed out that, in the default NDN implementation, a router that receives a duplicate Interest simply drops the Interest without sending a NACK back. This corresponds to the original CCN forwarding strategy. The ndnSIM NDN implementation also allows the use of NACKs after Interest loop detection. The results presented in this section for "CCN" correspond to the ndnSIM implementation of NDN without NACKs, and the results presented for "NDN" correspond to the ndnSIM implementation of NDN with NACKs enabled.

To implement Algorithms 3 to 8 defining SIFAH in ndnSIM, we had to make some modifications on the basic structures of ndnSIM, namely: the FIBs, Interest packets, NACKs, and the forwarding strategy. A new field "rank" is added to every entry of the FIB. Unlike ndnSIM in which the next hop selection for requested prefixes is based on hop count, in SIFAH next hops are sorted based on rank of each FIB entry. The field $h(k)$ was added to each Interest message, which determines the hop count from forwarding node k

to the prefix requested by the Interest. A new type of NACK for loop detection is added and the behavior of forwarding strategy for NACKs is modified based on SIFAH definitions. Furthermore, a new class of forwarding strategy is added to ndnSIM that implements SIFAH functions.

Simulation Scenarios To isolate the operation of the data plane from the performance of different routing protocols operating in the control plane, we used static routes and manually configured routing loops for specific prefixes.

Given the use of static routes and configured loops, we used a simple grid topology of sixteen nodes with two consumers producing Interests with different prefixes and one producer announcing the content requested in the Interests. Interest traffic is generated at a constant bit rate with a frequency of 2000 Interests per second. The delay over each link of the topology is set t to 10 msec and PIT entry expiration time is set to only 1000 msec, which is too short for real networks but is large enough to illustrate the consequences of undetected Interest loops.

Five different scenarios, each lasting 90 seconds of simulation time, were used to compare SIFAH with NDN and CCN. Each scenario is defined by the percentage of Interests traversing loops, which was set to equal 0%, 10%, 20%, 50%, and 100% of the Interests generated by consumers. In practice, it should be the case that only a small fraction of Interests traverse loops, assuming a correct routing protocol is used in the control plane and sensible policies are used to rank the available routes in the FIBs. The scenarios we present illustrate that just a few Interests traversing undetected loops cause performance degradation, and that network performance is determined by the PIT entry expiration times as the fraction of Interests traveling loops increases.

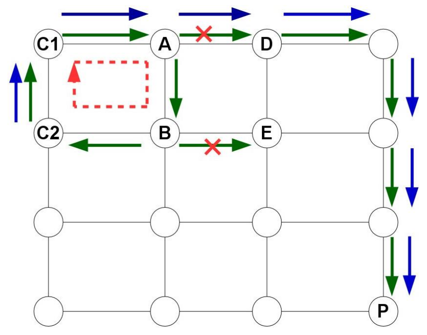


Figure 3.3: Initial Routes and Custom Loop Scenario

Figure 3.3 shows the topology and scenario we used in our simulations. Consumer C1, produces Interests for $prefix_1$ and $prefix_2$, and consumer C2 produces Interests only for $prefix_2$. Blue arrows and green arrows shows initial routes for $prefix_1$ and $prefix_2$, respectively. We assume that the route between nodes A and D, and the route between nodes B and E for $prefix_2$ are disconnected. Therefore, Interests requesting $prefix_2$ use alternate paths from node A to node B and from node B to node C2, which causes the looping of such Interests.

Interests for $prefix_2$ generated by C1 and C2, request the same content at approximately the same time, so that aggregation can take place at routers along the paths traversed by Interests. This results in the aggregation of Interests at node C1 for Interests generated by C2, and the aggregation of Interests at node C2 for Interests generated by C1. Our simple scenarios provide enough insight on the negative impact of undetected Interest loops in the presence of Interest aggregation using NDN and the original CCN design.

Simulation results are shown for three different forwarding strategies: The original CCN, NDN, and SIFAH. The difference between CCN and NDN is that CCN does not send

NACKs when duplicate Interests are detected. On the other hand, NDN sends NACK when simple loops are detected by receiving duplicate Interests.

Impact on PIT Entry Duration Figure 3.4 shows the average value of the PIT entry pending time for all PIT entries. When no Interest loops are present, NDN, CCN and SIFAH exhibit the same performance, with each having an average PIT entry pending time of 60 msec. This should be expected, given that Interests and NDO messages traverse shortest paths between consumers and producers or caches.

The average PIT entry pending time in SIFAH does not increase as he percentage of Interests that encounter Interest loops increases. The reason for this is that SIFAH ensures that an Interest must elicit either an NDO message or a NACK to be sent back from some router along the route it traverses back to the consumer that originates the Interest. Hence, the average amount of time an Interest entry spends in the PIT is a function of the round-trip time it takes for either an NDO message or a NACK to evict it from the PIT. This is proportional to a round-trip time between a consumer and a router with the content or a router at which HFAR is not satisfied, which is a few milliseconds in the grid topology.

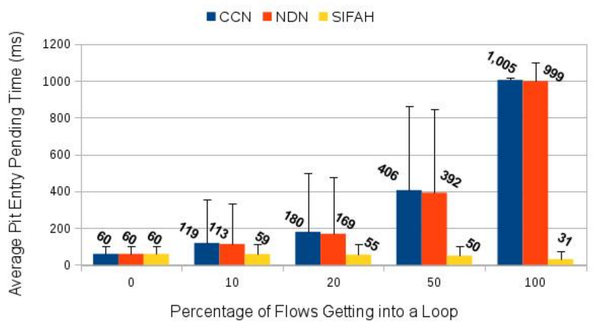


Figure 3.4: Average PIT entry pending time for CCN, NDN, and SIFAH

By contrast, the average PIT entry pending time in CCN and NDN increases dramatically with the percentage of Interests that encounter Interest loops. The percentage of Interests that traverse loops need not be large to have negative performance consequences. For the scenario in which 10% of the Interests encounter Interest loops, we observe that the average PIT entry pending time increases dramatically in NDN and CCN, with the average PIT entry pending time being 113 msec, which is about twice the average PIT entry pending time in SIFAH.

The results for NDN and CCN can be easily explained. CCN simply deletes and drops duplicate Interests, each Interest that encounters an Interest loop is discarded by the router that detects a duplicate Interest, and this action forces the corresponding PIT entries in the routers traversed by the Interest to remain in those PITs, until their PIT entry expiration timers expire. In NDN, Interest loops can go undetected with aggregation and therefore no NACKs are sent in those cases. As a result, the time an Interest entry spends in the PIT equals the PIT entry expiration time if the Interest traverses an undetected loop. The results are almost the same for CCN and NDN. The reason for observing slightly lower values for NDN compared to CCN, is that some of the Interests for content in $prefix_2$ are not generated by C1 and C2 with sufficient time correlation to enable Interest aggregation, which results in detection of Interest loops in NDN and Interests being discarded in CCN.

The PIT entry pending times in NDN and CCN are many orders of magnitude larger for Interests that traverse undetected Interest loops. This is unavoidable, given that the PIT entry pending time is proportional to a PIT entry expiration time, which by design must be set conservatively to values that are far longer than average round-trip times between consumers and producers. In the simulations, the PIT entry expiration time is just one second.

Impact on PIT Size Figure 3.5 shows the average size of PIT tables in terms of number of entries for a router included in Interest flows for five different scenarios comparing CCN, NDN, and SIFAH. CCN, NDN and SIFAH have exactly the same PIT size in the absence of Interest loops, which is expected. As the percentage of Interests that encounter loops increases, the average number of entries in the PITs increases dramatically for CCN and NDN. For the case in which only 10% of Interests encounter loops, the number of entries doubles in NDN and CCN compared to SIFAH. For the case in which 100% of Interests encounter loops, the average number of PIT entries in CCN and NDN is 1889 and 1884, respectively, while the number of PIT entries in SIFAH actually decreases.

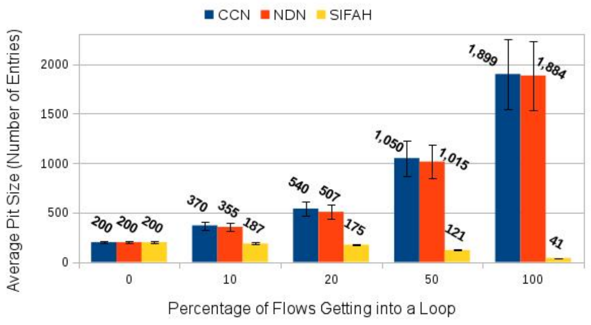


Figure 3.5: Average PIT table size for CCN, NDN, and SIFAH

The reason for the decrease in average number of PIT entries for SIFAH as the percentage of Interests that encounter loops increases is a consequence of the shorter round-trip times between the consumers submitting Interests and the routers sending NACKs compared to the round-trip times of paths to the producers of requested content.

Impact on Round-Trip Times Figure 3.6 shows the average round-trip time (RTT) for all five scenarios for CCN, NDN and SIFAH. In the simulation experiments, the round-trip time is considered to be the time elapsed from the instant when an Interest is first sent to the instant when an NDO message or a NACK is received by the consumer who created the Interest.

For the case of no loops, CCN, NDN, and SIFAH have the same average RTT. When the percentage of Interests traversing loops is 10%, the average RTT in CCN and NDN increases to almost two times the average RTT in SIFAH, and some Interests have much larger RTTs than the average. As the percentage of Interests that loop increases, the average RTT becomes proportional to the PIT entry expiration time, which is to be expected. The average RTT in SIFAH decreases as more Interests traverse loops, which is a result of the shorter RTTs between consumers and routers sending the NACKs.

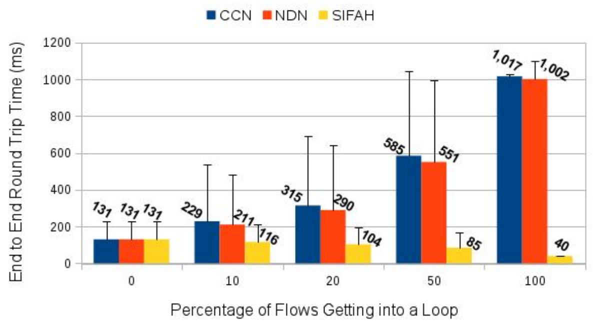


Figure 3.6: Average round trip time (RTT) for CCN, NDN, and SIFAH

3.4.3 Design Implications

The simulation experiments we have presented are meant only to help illustrate the negative impact of undetected Interest loops when they occur, rather than to provide representative scenarios of the performance of Interest-based forwarding strategies in large networks. Our results illustrate that loops in FIBs need not be long lasting or impact a large percentage of Interest to cause the number of stored PIT entries and end-to-end delays to increase quickly.

As we have shown, the PIT storage requirements for SIFAH are smaller than those for the original CCN and NDN forwarding strategies. Thus, SIFAH is more efficient than CCN and NDN even in the absence of Interest loops. Given that SIFAH is so easy to implement in the context of CCN and NDN, it makes practical sense to eliminate the current practice in NDN and CCN of attempting to detect Interest loops by the matching of nonces and Interest names, which does not work.

3.5 Conclusions

We showed that the forwarding strategies in NDN and the original CCN architectures may fail to detect Interest loops when they occur, and that a correct forwarding strategy that supports Interest aggregation cannot be designed simply by identifying each Interest uniquely and deciding that there is an Interest loop based on the matching of Interest names and nonces.

We introduced the Strategy for Interest Forwarding and Aggregation with Hop-counts (SIFAH). It is the first Interest-based forwarding strategy shown to be correct in the presence of Interest loops, Interest aggregation, faults, and the forwarding of Interests over multiple paths. SIFAH operates by requiring that FIBs store the next hops and the hop count through such hops to named content, and by having each Interest state the name of the content requested and the hop count from the relaying router to the content.

We showed that SIFAH incurs less storage overhead than using nonces to identify Interests. We also showed that, if NDN or the original CCN design is used in a network, the number of PIT entries and end-to-end delays perceived by consumers can increase substantially with just a fraction of Interests traversing undetected loops. Although our simulation experiments assumed a very small network, our results provide sufficient insight on the negative effects of undetected Interest loops in NDN and the original CCN design.

This work is just a first step in the definition of correct Interest-based forwarding strategies, and it is applicable to any Interest retransmission approach. For simplicity, we assumed that content consumers are in charge of Interest retransmissions and that routers do not provide local repair of Interests after receiving NACKs or detecting link failures. The design of an efficient Interest retransmission strategy and determining whether Interest retransmissions by routers improves performance are arguably the most important next steps. However, SIFAH provides the necessary foundation to define a correct retransmission strategy, because it guarantees that each Interest results in an NDO message or a NACK being sent to the consumer who originated the Interest.

More work is also needed to understand the performance of SIFAH in large networks, the effect of PIT entry expiration timers on performance, the effect of load balancing of Interests over multiple available routes to content, the impact of local repairs in Interest forwarding, and the performance implications of the interaction between SIFAH and a routing protocol that guarantees loop-free routing tables (and hence FIBs) at all times [10, 11, 12] compared to one that does not [125].

Algorithm 3 SIFAH Processing of Interest at router i

```
1: function Process Interest
2: INPUT: $PIT^i$, $CS^i$, $FIB^i$, $I[n(j), h^I(k)]$;
3: if $n(j) \in CS^i$ then send $D[n(j), sig(j)]$ to $k$
4: if $n(j) \notin CS^i$ then
5:     if $n(j) \notin PIT^i$ then
6:         if $n(j)^* \notin FIB^i$ then
7:             % Route failed for $n(j)^*$:
               send $NI[n(j), \textsf{no route}]$ to $k$; drop $I[n(j), h^I(k)]$
8:         else
9:             if $\exists\ v \in S^i_{n(j)^*}(\ h^I(k) > h(i, n(j)^*, v)\ )$ then
10:                % Interest can be forwarded:
                   call Forwarding Strategy($PI^i_{n(j)}$)
11:            else
12:                % Interest may be traversing a loop:
                   send $NI[n(j), \textsf{loop}]$ to $k$;   drop $I[n(j), h^I(k)]$
13:            end if
14:        end if
15:    else
16:        % There is a PIT entry for $n(j)$:
17:        if $h^I(k) > h^I(i)$ then
18:            % Interest can be aggregated:
               $INSET(PI^i_{n(j)}) = INSET(PI^i_{n(j)}) \cup k$
19:        else
20:            % Interest may be traversing a loop:
               send $NI[n(j), \textsf{loop}]$ to $k$;  drop $I[n(j), h^I(k)]$
21:        end if
22:    end if
23: end if
24: end function
```

Algorithm 4 SIFAH Interest forwarding at router i

```
1: function Forwarding Strategy
2: INPUT: $PIT^i$, $FIB^i$, $MIL$, $I[n(j), h^I(k)]$;
3: for each $v \in S^i_{n(j)^*}$ by rank do
4:     if $h^I(k) > h(i, n(j)^*, v)$ then
5:         create $PI^i_{n(j)}$;
           $INSET(PI^i_{n(j)}) = \{k\}$; $OUTSET(PI^i_{n(j)}) = \{v\}$;
           $RT(PI^i_{n(j)}) = MIL$; $h^I(i) = h(i, n(j)^*, v)$;
           forward $I[n(j), h^I(i)]$ to $v$; return
6:     end if
7: end for
8: % No neighbor can be used in $S^i_{n(j)^*}$:
   for each $k \in INSET(PI^i_{n(j)})$ send $NI[n(j), \text{no route}]$ to $k$
9: end function
```

Algorithm 5 Process NDO message from q at router i

```
1: function Process NDO message
2: INPUT: $PIT^i$, $CS^i$, $FIB^i$, $D[n(j), sig(j)]$ received from $q$;
3: [o] verify $sig(j)$;
4: [o] if verification fails then drop $D[n(j), sig(j)]$
5: if $n(j) \in PIT^i \wedge q \in OUTSET(PI^i_{n(j)})$ then
6:     for each $p \in INSET(PI^i_{n(j)})$ do
       send $D[n(j), sig(j)]$ to $p$;
7:     [o] store the content with name $n(j)$ in $CS^i$;
8:     delete $PI^i_{n(j)}$
9: else
10:     drop $D[n(j), sig(j)]$
11: end if
12: end function
```

Algorithm 6 Process Interest life-time expiration

```
1: function Process Interest Life-time Expiration
2: INPUT: $PIT^i$, $RT(P^i_{n(j)}) = 0$;
3: for each $p \in INSET(PI^i_{n(j)})$ do
   send $NI[n(j), \text{Interest expired}]$
4: delete $PI^i_{n(j)}$
5: end function
```

Algorithm 7 Process NACK at router i

```
1: function Process NACK
2: INPUT: $PIT^i$, $NI[n(j), \mathsf{CODE}]$;
3: if $n(j) \notin PIT^i$ then
4:     drop $NI[n(j), \mathsf{CODE}]$
5: else
6:     if $k \notin OUTSET(PI^i_{n(j)})$ then drop $NI[n(j), \mathsf{CODE}]$;
7:     if $k \in OUTSET(PI^i_{n(j)})$ then
8:         for each $p \in INSET(PI^i_{n(j)})$ do
           send $NI[n(j), \mathsf{CODE}]$;
9:         delete $PI^i_{n(j)}$
10:     end if
11: end if
12: end function
```

Algorithm 8 Process failure of link (i, k) at router i

```
1: function Process Link Failure
2: INPUT: $PIT^i$;
3: for each $n(j) \in PIT(i)$ do
4:     if $k \in INSET(PI^i_{n(j)})$ then
5:         $INSET(PI^i_{n(j)}) = INSET(PI^i_{n(j)}) - \{k\}$;
           if $INSET(PI^i_{n(j)}) = \emptyset$ then delete $PI^i_{n(j)}$;
6:     end if
7:     if $k \in OUTSET(PI^i_{n(j)})$ then
8:         $OUTSET(PI^i_{n(j)}) = OUTSET(PI^i_{n(j)}) - \{k\}$;
9:         if $OUTSET(PI^i_{n(j)}) = \emptyset$ then
10:             for each $p \in INSET(PI^i_{n(j)})$ do
11:                 send $NI[n(j), \mathsf{route\ failed}]$
12:             end for
13:             delete $PI^i_{n(j)}$
14:         end if
15:     end if
16: end for
17: end function
```

Chapter 4

Content Centric Networks without Pending Interest Tables

The leading approach in information-centric networking (ICN) architectures can be characterized as *Interest-based*, and consists of: populating forwarding information bases (FIB) maintained by routers with routes to name prefixes denoting content, sending content requests (called Interests) for specific content objects (CO) over paths implied by the FIBs, and delivering Data packets with content objects along the reverse paths traversed by Interests. Today, named data networking (NDN) [17] and CCNx [4] are the most prominent Interest-based ICN approaches.

Since the introduction of CCN [14], it has been assumed [4, 17, 24] that: (a) Interest loops can be detected by stating a content name or a content name and a nonce; (b) per-Interest forwarding state maintained in Pending Interest Tables (PIT) is required for Interests and responses to them to be forwarded without revealing the identities of the sources of Interests; and (c) Interests requesting the same content need to be aggregated in the PITs to attain efficiency. Several results have been reported regarding the PIT sizes required for NDN or CCN to operate at Internet scale, and several approaches have been proposed to reduce the large storage requirements of PITs [33, 41, 42, 43]. However, no prior work explores alternatives that eliminate PITs and per-Interest forwarding state.

The main contributions of this chapter consist of showing by example that the premises stated above are not true; and introducing a far more efficient alternative for content-centric networking than NDN and the original CCN proposal.

Section 4.1.1 addresses the impact of Interest aggregation in NDN by means of simulation experiments ran using the implementation of NDN in ndnSIM [1] without modifications. The experiments assume networks with on-path caching and average round trip times (RTT) that are representative of recent IP latency statistics. The results show that the percentage of Interests that are aggregated and the performance benefits derived from Interest aggregation become negligible as the storage capacity per cache increases.

Section 7.1.1 introduces **CCN-DART**, which replaces PITs with data answer routing tables (DART). A DART stores forwarding state regarding the routes traversing the router, rather than the Interests forwarded by the router. An Interest in CCN-DART states the name of the requested content, a hop count, and a destination-and-return token (*dart*). The hop count is used to avoid forwarding loops. The dart leaves a trace of the path traversed by the Interest using local identifiers of the previous hop and the current hop, without revealing the identity of the source of the Interest.

Section 7.1 proves that no forwarding loops can occur in CCN-DART and that responses to Interests are forwarded correctly. Section 7.2 compares the performance of CCN-DART with NDN when routes to name prefixes are loop-free and static, and either on-path caching or edge caching is used in a 200-router network. CCN-DART attains similar or better end-to-end latencies and incurs very similar Interest traffic than NDN to retrieve content, even though Interests are not aggregated in CCN-DART. However, at high Interest rates, NDN requires 10 to 20 times the number of forwarding entries needed in CCN-DART.

4.1 Interest Aggregation

4.1.1 Impact of Interest Aggregation in NDN

We analyze the impact of interest aggregation on the performance of NDN using simulations carried out with the ndnSIM simulation tool [1]. We used the implementation of NDN in ndnSIM without modifications. Our study is independent of the Interest retransmission strategy. For simplicity, we assume that routers use exact Interest matching to decide whether an Interest can be answered. We consider the percentage of aggregated Interests in the network and the average number of PIT entries created per second per router as the performance metrics.

Scenario Parameters and Scenarios The simulation parameters we consider include the average latencies between routers, the storage capacity of caches, the Interest request rates from routers, the popularity of content, and the temporal correlation of content requests.

The scenarios we use consist of random networks with 200 nodes corresponding to routers distributed uniformly in a 100m$\times$100m area. Routers with 12m or shorter distance are connected to each other with a point-to-point link, which results in a topology with 1786 edges. Each router acts as a producer of content and also has local consumers generating Interests.

Producers are assumed to publish 1,000,000 different content objects that are uniformly distributed among routers. For simplicity, we assume that *all* routers have the same storage capacity in their caches, which depending on the experiment ranges from 0 to up to 100,000 cache entries per router, or 10% of the published objects.

The distribution of object requests determines how many Interests from different users request the same content.

It has been argued [34, 35] that Internet traffic follows a Zipf distribution with a parameter (α) value close to 1. A smaller Zipf parameter value results in a lower Interest aggregation amount. Accordingly, we model object popularity using a Zipf distribution with values of α equal to 0.7 and 1.

We considered different values of the *total Interest rate per router*, corresponding to the sum of Interests from all local users. Increasing values of Interest rates can be viewed as higher request rates from a constant user population of local active users per router, or an increasing population of active users per router. For example, 50 to 500 Interests per second per router can be just 10 Interests per second per active user for a local population of 5 to 50 concurrently active users per router. The Interest rates we assume per router are not large compared to recent results on the size that PITs would have in realistic settings [33, 42, 43, 41].

The percentage of Interests that benefit from Interest aggregation is a function of the time elapsed from the time when a router receives an Interest, until the time the requested CO or a NACK is received by that router. In turn, this time is a function of the RTT between the router originating the Interest and the site with the requested content, as well as the PIT entry expiration time when the Interest is not answered with a Data packet or a NACK.

Recent Internet latency statistics [123, 32] show that Internet traffic latency varies from 11ms for regional European traffic to 160ms for long-distance traffic. Accordingly, we consider point-to-point delays of 10ms between neighbor routers in many of our simulations, which leads to RTTs of about 200ms. We also carried our experiments varying the RTT of the network below and above 200ms.

Simulation Results The following simulation results can be viewed as applicable to the steady-state behavior of a network using NDN.

Figure 4.1 shows the percentage of aggregated Interests for four different total request rates per router from 50 to 500 Interests per second, and seven different storage capacities of the caches, ranging from 100 to 10,000 objects (i.e., up to 1% of the published objects). The Zipf parameter value assumed is $\alpha = 1$, which is the best case for Interest aggregation.

The impact of in-network caching on Interest aggregation is very clear. Interest aggregation is useful when caches have insignificant capacities and request rates are high. However, as the capacity of a cache in each router increases, the percentage of Interests that are aggregated drops quickly. The percentage of aggregated Interests drops to less than 5%

for a request rate per router of 500 requests per second when a cache can store only 0.1% of the published objects.

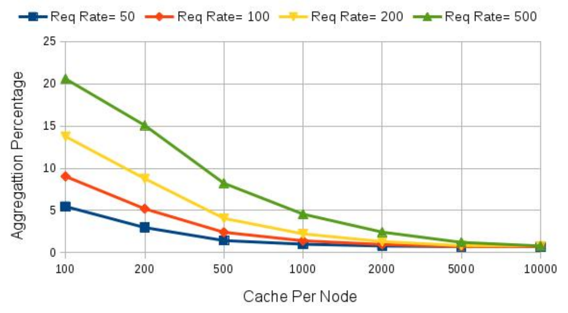


Figure 4.1: Interest aggregation as a function of the storage capacity per content store ($\alpha = 1$).

Figure 4.2 shows the effect of the α parameter and RTTs when the total request rate per router is only 50 Interests per second. The latencies between neighbor routers are set to 5 and 15 ms, which produce RTTs of 66 to 70 ms and 193 to 200 ms, respectively. It is clear that Interest aggregation is far less important when consumers are less likely to request similar content ($\alpha = 0.7$). It is also clear that the benefits of Interest aggregation vanish as caches are allowed to cache more content. When caches can store up to 1% of the total number of objects published, the percentage of Interests that are aggregated is less than 2% for $\alpha = 1$ and less than 0.8% for $\alpha = 0.7$.

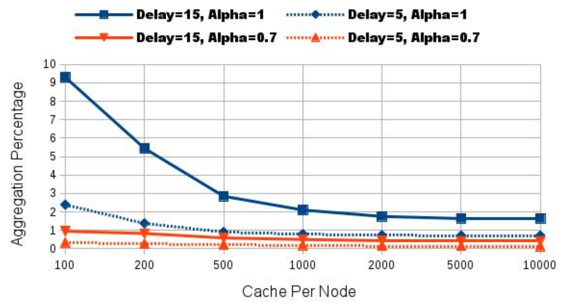


Figure 4.2: Interest aggregation as a function of the values of Zipf parameter, storage capacity, and RTTs

Figure 4.3 shows the average number and variance of PIT entries created per second per router with total requests rates per router varying from 50 Interests per second to 500 Interests per second. Results are presented for six different values of cache capacity ranging from no caching to 10% of the published objects. In all cases, the average number of PIT entries created grows proportionally with the request rate per router. The large variance indicates that only some routers benefit from Interest aggregation. Furthermore, orders of magnitude increases in the storage capacity of content stores do not produce a similar reduction in the number of PIT entries created per second per router, which is a function of the total request rates after caches are large enough.

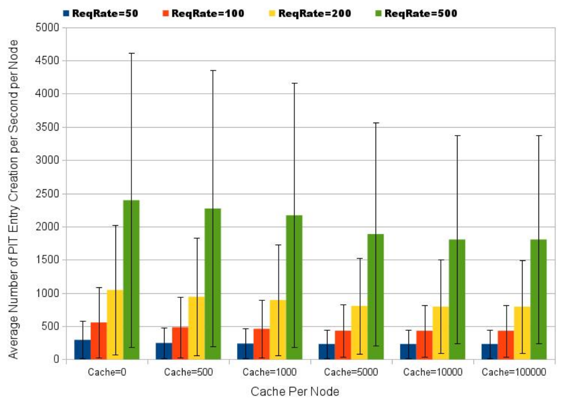


Figure 4.3: Average number of PIT entries created per second at each router

In theory, Interest aggregation is most useful when Interests exhibit temporal correlation, such as when popular live events take place. Figure 4.4 shows the impact of caching on Interest aggregation when Interests have temporal correlation and either no caching is used or caches with capacity for 1000 objects are used (only 0.1% of total objects published in the network). Localized Interests are generated using the model proposed by Dabirmoghaddam et al [5] with a Zipf parameter value of $\alpha = 0.7$ and results for three total Interest rates per router and four temporal localization factors for Interests are shown. A higher temporal locality factor indicates a higher degree of popularity of objects in the same time period. The results in Figure 4.4 show that, without caching, Interest aggregation *is* very important for all values of temporal locality of Interest popularity, and is more important when Interest locality is high (large localization factor). However, once caching is allowed and even if caches can store only up to 0.1% of the published objects, the percentage of aggregated Interests is minuscule and actually decreases with the temporal correlation of Interests. This experiment further illustrates the overlapping nature of PITs and caches in NDN.

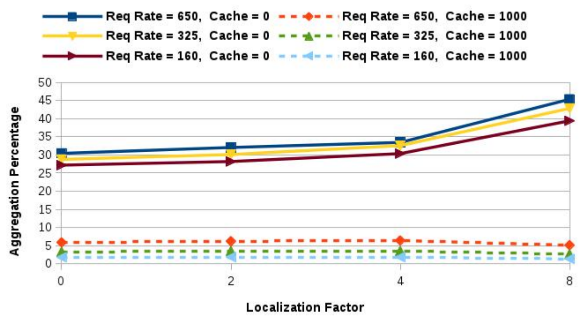


Figure 4.4: Impact of caching on the aggregation of Interests with temporal locality

4.2 CCN-DART

4.2.1 Design Rationale and Assumptions

The design of CCN-DART is based on three observations. First, the results in Section 4.1.1 show that content caching—which is essential for content-centric networking—makes the occurrence of Interest aggregation extremely rare an obviates the need for PITs as a means of reducing the number of Interests being forwarded. In-network caching tends to make popular content available locally before subsequent requests for the same content arrive. This is important, because using PITs comes at a big price. PITs enable users to mount Interest flooding attacks aimed at overflowing PITs [114, 43], and the storage overhead they incur is significant [33, 41, 42].

Second, the number of routers in a network is orders of magnitude smaller than the number of COs accessed through them. Hence, maintaining forwarding state based on the *routes* going through a router–each used by many Interests–is by nature orders of magnitude smaller than forwarding state based on the *Interests* traversing a router.

Third, preventing Interests from traversing loops when Interest aggregation is allowed cannot be attained by identifying Interests uniquely using names of content objects and nonces [36, 37]. However, it can be done based on an ordering of the routers that forward the Interests, and content routing protocols [10, 125] can provide all the needed information to attain proper ordering in the forwarding plane.

We make the following assumptions in the description of CCN-DART, none of which should be considered design requirements for the basic approach we introduce.

Interests are retransmitted only by the users that originated them, rather than routers that relay Interests. Of course, "local repair" mechanisms can be used in CCN-DART to react more quickly to congestion or topology changes.

We assume that routers use exact Interest matching. Given the name of a CO, a router can determine whether or not the exact same CO is stored locally.

The COs corresponding to a name prefix could be stored in subsets of the prefix at multiple sites, with each site announcing that the entire prefix is local. However, resolving an Interest for a given CO in this case would require contacting all the sites where the prefix is local. For simplicity, in our description of CCN-DART we assume that a router that announces being an origin of a name prefix stores all the COs in that prefix locally, and call such a router an *anchor* of the prefix. If all the COs of a prefix are mirrored at multiple sites, each router connected to the site storing the COs is an anchor of the prefix.

Routers know which interfaces are neighbor routers and which are local consumers, and forward Interests on a best-effort basis. For convenience, a request for content from a local user is sent to its router in the form of an Interest stating an empty hop count to content and an empty dart.

4.2.2 Information Exchanged and Stored

CCN-DART uses Interests, NACKs and Data packets to support content exchange among routers. Our description of these messages addresses only that information needed to attain correct forwarding, which consists of the names of COs, the hop counts to prefixes, and *destination-and-return tokens* (**darts**). The terms neighbor and interface are used interchangeably. The name of CO j is denoted by $n(j)$, the name prefix that is the best match for $n(j)$ in a FIB is denoted by $n(j)^*$, and $S^i_{n(j)^*}$ denotes the set of neighbors of router i considered to be next hops to prefix $n(j)^*$. Darts are local identifiers used to uniquely denote routes established between source and destination routers over which Interests, Data packets, and NACKs are sent. Accordingly, darts can be very small (e.g., 32 bits).

An Interest forwarded by router k requesting CO j is denoted by $I[n(j), h^I(k)$ $, dart^I(k)]$. It states the name of the requested CO ($n(j)$), the hop count ($h^I(k)$) from k

to prefix $n(j)^*$, and the dart $(dart^I(k))$ used to establish an anonymous route back to the router that originates the Interest.

A Data packet sent in response to an Interest is denoted by $DP[n(j), sp(j), dart^I(i)]$, and states the name of the CO requested in the Interest being answered $(n(j))$, a security payload $(sp(j))$ used optionally to validate the content object, the dart $(dart^I(i))$ from the Interest being answered, and the CO itself.

A NACK to an Interest is denoted by $NA[n(j), \mathsf{CODE}, dart^I(i)]$ and states the name of the CO $(n(j))$, a code (CODE) indicating the reason why the NACK is sent, and the dart $(dart^I(i))$ from the Interest being answered. Reasons for sending a NACK include: an Interest loop is detected, no route is found towards requested content, and no content is found.

Router i maintains three tables: a forwarding information base (FIB^i), a data-answer routing table $(DART^i)$, and an *optional* requested-content table (RCT^i). All routers must maintain FIBs and DARTs, and only those routers with local users and routers supporting content caching need to maintain an RCT.

A *predecessor* of router i for Interests related to name prefix $n(j)^*$ is a router that forwards Interest for COs with names that are best matched by $n(j)^*$. Similarly, a *successor* of router i for Interests related to $n(j)^*$ is a router to whom router i forwards Interest regarding COs with names that are best matched by name prefix $n(j)^*$.

FIB^i is indexed using name prefixes. The entry for prefix $n(j)^*$ consists of a set of tuples, one for each next hop to prefix $n(j)^*$. The tuples for prefix $n(j)^*$ are ranked based on their utility for forwarding. As a minimum, the tuple for next hop $q \in S^i_{n(j)^*}$ states:

1. $h(i, n(j)^*, q)$: The distance to $n(j)^*$ through q.

2. $a(i, n(j)^*, q)$: The nearest anchor of $n(j)^*$ through q.

$DART^i$ stores the mappings of predecessors to successors along loop-free paths to name prefixes. The entry created for Interests received from router p (predecessor) and forwarded to router s (successor) towards a given anchor a of a name prefix is denoted by $DART^i(a, p)$ and specifies:

1. $a^i(a, p)$: The anchor a for which the entry is set.
2. $p^i(a, p)$: The predecessor p of the path to $a^i(a, p)$.
3. $pd^i(a, p)$: The *predecessor dart*, which equals the dart received in Interests from p forwarded towards $a^i(a, p)$.
4. $s^i(a, p)$: The successor s selected by router i to forward Interests received from p towards $a^i(a, p)$.
5. $sd^i(a, p)$: The *successor dart* included in Interests sent by router i towards $a^i(a, p)$ through the successor.
6. $h^i(a, p)$: The hop-count distance to prefix a through successor s when i establishes the DART entry.

DART entries can be removed using a least-recently used policy or a maximum lifetime, for example. An entry in a DART can remain in storage for long periods of time in the absence of topology changes. The removal of a DART entry simply causes a router to compute a new entry for Interests flowing towards an anchor of prefixes.

RCT^i serves as an index of local content as well as local requests for remote content. It is indexed by the CO names that have been requested by the router. The entry for CO name $n(j)$ states the name of the CO ($n(j)$), a pointer to the local storage where the CO ($p[n(j)]$) is stored, and a list of zero or more identifiers of local consumers ($lc[n(j)]$)

that have requested the CO. The RCT could be implemented as two separate indexes, one for local content and one for requests for remote content.

If router i is an anchor of name prefix $n(j)^*$ then it stores all the COs with names that match the name prefix. This is denoted by $n(j)^* \in RCT^i$. If CO $n(j)$ has been requested by one ore more local consumers and no copy of the CO is yet available, then $n(j) \in RCT^i$, $p[n(j)] = nil$, and $lc[n(j)] \neq \phi$. On the other hand, if router i caches CO $n(j)$, then $n(j) \in RCT^i$, $p[n(j)] \neq nil$, and $lc[n(j)] = \phi$.

4.2.3 Preventing Forwarding Loops

We have shown [36, 37] that undetected Interest loops can occur in NDN and CCNx when Interests are aggregated while traversing routing loops resulting from inconsistencies in FIB entries or inconsistent rankings of routes at different routers. CCN-DART uses the same approach we proposed for the elimination of undetected Interest loops in the context of NDN and CCNx to prevent forwarding loops when DART entries are created.

Dart Entry Addition Rule (DEAR):

Router i accepts $I[n(j), h^I(k), dart^I(k)]$ from router k and creates a DART entry for prefix $n(j)^*$ with k as its predecessor and a router $v \neq k$ as its successor if:

$$\exists\ v \in S^i_{n(j)^*}(\ h^I(k) > h(i, n(j)^*, v)\ )$$

The distance information that must be stored in the FIBs to implement DEAR can be obtained easily from the control plane. Such content routing protocols as DCR [10] and NLSR [125] are able to compute the required minimum-hop distances, which can then be copied into the FIBs.

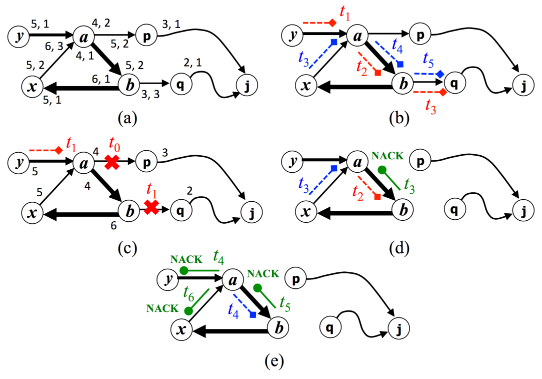


Figure 4.5: CCN-DART avoids forwarding loops

Figures 4.5(a) and (b) illustrate how using DEAR prevents Interests from traversing loops when a multi-path routing protocol is used and FIB entries are consistent but local rankings of multiple routes available at each router (e.g., NLSR [125]) cause routing loops. The pair of numbers next to a node in Figure 4.5(a) indicate the hop count from that router to $n(j)^*$ over an interface and the ranking of the interface according to the FIB of the router.

Let the triplet (v, h, r) denote an interface, its hop count and its ranking. In Figure 4.5(a), FIB^a states $(b, 4, 1)$, $(p, 4, 2)$, $(x, 6, 3)$, and $(y, 6, 4)$; FIB^b states $(x, 6, 1)$, $(a, 5, 2)$, and $(q, 3, 3)$; and FIB^x states $(a, 5, 2)$ and $(b, 5, 1)$. As Figure 4.5(b) shows, router a receives $I[n(j), h^I(y) = 5, dart^I(y)]$ from router y at time t_1 and forwards $I[n(j), h^I(a) = 4, dart^I(a)]$ to b because $5 = h^I(y) > h(a, n(j)^*, b) = 4$ and b is ranked above p. Similarly, router b receives the Interest at time t_2 and accepts it because $4 = h^I(a) > h(b, n(j)^*, q) = 3$. Router b uses router q as the next hop for the Interest, because q is the highest ranked neighbor satisfying DEAR. This example illustrates that, independently of local rankings of multiple routes to prefixes, Interests traverse simple paths by requiring each relaying router to satisfy DEAR.

Figures 4.5(c) to (e) illustrate how DEAR operates when FIBs are inconsistent due to topology changes. Routers a and b update their FIBs at time times t_0 and t_1, respectively. We assume that the routing updates have not been processed at routers y and a when they forward Interests at time t_1 and t_2, respectively. As shown in Figure 4.5(d), router b *must* send a NACK to router a because it does not have a neighbor with a shorter hop count to prefix $n(j)^*$ than $h^I(a) = 4$. In turn, router a forwards a NACK to router y, and the Interest from x also prompts a NACK from b because DEAR is not satisfied. Within a finite time after t_1, the FIBs of routers are updated to show that prefix $n(j)^*$ cannot be reached and Interests from local users for COs in that prefix cannot forwarded by routers a, b, x and y.

By contrast, assuming NDN or CCNx in the same example results in the Interests sent by y and x to be forwarded along the forwarding loop involving a, b and x. Router a aggregates the Interest from x, and router x aggregates the Interest from y. Those Interests must then "wait to infinity" in the PITs until their lifetimes expire or they are otherwise evicted from the PITs. Using nonces in Interests incurs considerable PIT storage overhead. However, denoting Interests using only CO names as in CCNx can result in even more Interest-looping problems. Given the speed with which FIBs are updated to reflect correct distances computed in the control plane, false loop detection using DEAR should be rare, and it is preferable by far than storing PIT entries that expire only after many seconds without receiving responses.

4.2.4 Maintaining Forwarding State

Algorithms 9 to 4 specify the steps taken by routers to process and forward Interests, Data packets, and NACKs. The algorithms we present assume that the control plane updates FIB^i to reflect any changes in hop counts to name prefixes and anchors resulting from the loss of connectivity to one or more neighbors or link-cost changes. In addition, a DART entry is silently deleted when connectivity with the successor or predecessor of the entry is lost, or it is not used for a prolonged period of time.

Algorithm 9 shows the steps taken by router i to process Interests received from local consumers. For convenience, content requests from local consumers are assumed to be Interests stating the name of a CO, together with an empty hop count to content and an empty dart.

Router i first looks up its RCT to determine if the requested CO is stored locally or a request for the CO is pending. If the CO is stored locally, a Data packet is sent back to the user requesting it. If a request for the same content is pending, the name of the user is added to the list of customers that have requested the CO. Router i sends back a NACK if it is an anchor of name prefix $n(j)^*$ and the specific CO is not found locally, or the CO is remote and no FIB entry exists for a name prefix that can match $n(j)$.

If possible, router i forwards the Interest through the highest ranked neighbor in its FIB for the name prefix matching $n(j)$. How such a ranking is done is left unspecified, and can be based on a distributed or local algorithm.

If a DART entry exists for the selected successor that should receive the Interest, the existing route is used; otherwise, a new DART entry is created before the Interest is sent. The successor dart assigned to the new DART entry is a locally unique identifier that must be different from all other successor darts being used by router i.

Algorithm 9 Processing Interest from user c at router i

```
function Interest_Source
INPUT: $RCT^i$, $FIB^i$, $DART^i$, $I[n(j), nil, nil]$;
if $n(j) \in RCT^i$ then
    if $p[n(j)] \neq nil$ then
        retrieve CO $n(j)$; send $DP[n(j), sp(j), nil]$ to $c$
    else
        $lc[n(j)] = lc[n(j)] \cup c$; $p[n(j)] = nil$ (% Interest is aggregated)
    end if
else
    if $n(j)^* \in RCT^i$ then
        send $NA[n(j), \text{no content}, nil]$ to $c$ (% $n(j)$ does not exist)
    else
        if $n(j)^* \notin FIB^i$ then
            send $NA[n(j), \text{no route}, nil]$ to $c$ (% No route to $n(j)^*$ exists)
        else
            create entry for $n(j)$ in $RCT^i$: (% Interest from $c$ is recorded)
            $lc[n(j)] = lc[n(j)] \cup c$; $p[n(j)] = nil$;
            for each $v \in S^i_{n(j)^*}$ by rank in $FIB^i$ do
                $a = a(i, n(j)^*, v)$;
                if $\exists DART^i(a, i)$ ( $s^i(a, i) = v$ ) then
                    $h^I(i) = h^i(a, i)$; $dart^I(i) = sd^i(a, i)$;
                    send $I[n(j), h^I(i), dart^I(i)]$ to $v$; return
                else
                    create entry $DART^i(a, i)$:
                    compute $SD \neq sd^i(p, q)\ \forall\ DART^i(p, q)$;
                    $pd^i(a, i) = SD$; $sd^i(a, i) = SD$;
                    $p^i(a, i) = i$; $s^i(a, i) = v$; $h^i(a, i) = h(i, n(j)^*, v)$;
                    $h^I(i) = h^i(a, i)$; $dart^I(i) = sd^i(a, i)$;
                    send $I[n(j), h^I(i), dart^I(i)]$ to $v$; return
                end if
            end for
        end if
    end if
end if
```

Algorithm 10 outlines the processing of Data packets. If the router has local consumers that requested the content, the Data packet is sent to those consumers based on the information stored in RCT^i. If the Data packet is received in response to an Interest that was forwarded from router k, router i forwards the Data packet after swapping the successor dart received in the Data packet for the predecessor dart stored in $DART^i$. Router i stores the data object if caching is supported.

Algorithm 10 Processing Data packet at router i

```
function Data Packet_Handling
INPUT: $DART^i$, $RCT^i$, $DP[n(j), sp(j), dart^I(q)]$;
[o] verify $sp(j)$;
[o] if verification fails then discard $DP[n(j), sp(j), dart^I(q)]$
if $\exists DART^i(a,k)$ ( $dart^I(q) = sd^i(a,k) \wedge p^i(a,k) = i$ )
  (% router $i$ was the origin of the Interest) then
    for each $c \in lc[n(j)]$ do
        send $DP[n(j), sp(j), nil]$ to $c$; $lc[n(j)] = lc[n(j)] - \{c\}$
    end for
end if
if $\exists DART^i(a,k)$ ( $dart^I(q) = sd^i(a,k) \wedge p^i(a,k) = k \in N^i$ ) then
    (% Data packet can be forwarded to $k$:)
    $dart^I(i) = pd^i(a,k)$; send $DP[n(j), sp(j), dart^I(i)]$ to $k$
end if
[o] if no entry for $n(j)$ exists in $RCT^i$ then
        create $RCT^i$ entry for $n(j)$: $lc[n(j)] = \emptyset$
    end if
[o] store CO in local storage; $p[n(j)]$ = address of CO in local storage
```

Algorithm 11 states the steps taken to handle NACKs, which are similar to the forwarding steps taken after receiving a Data packet. Router i forwards the NACK to local consumers if it was the origin of the Interest, or to a neighbor router k if it has a DART entry with a successor dart matching the dart stated in the NACK.

Algorithm 11 Process NACK at router i

```
function NACK_Handling
INPUT: $DART^i$, $RCT^i$, $NA[n(j), \mathsf{CODE}, dart^I(q)]$;
if $\exists DART^i(a,k)$ ( $dart^I(q) = sd^i(a,k) \wedge p^i(a,k) = i$ )
  (% router $i$ was the origin of the Interest) then
    for each $c \in lc[n(j)]$ do
        send $NA[n(j), \mathsf{CODE}, nil]$ to $c$; $lc[n(j)] = lc[n(j)] - \{c\}$
    end for
    delete entry for $n(j)$ in $RCT^i$
end if
if $\exists DART^i(a,k)$ ( $dart^I(q) = sd^i(a,k) \wedge p^i(a,k) = k \in N^i$ ) then
    (% NACK can be forwarded to router $k$:)
    $dart^I(i) = pd^i(a,k)$; send $NA[n(j), \mathsf{CODE}, dart^I(i)]$ to $k$
end if
```

Algorithm 12 shows the steps taken by router i to process an Interest received from a neighbor router k. If the requested content is cached locally, a Data packet is sent back. As in Algorithm 9, router i sends back a NACK if it is an anchor of $n(j)^*$ and the CO with name $n(j)$ is not found locally, or the CO is remote and no FIB entry exists for $n(j)^*$. In contrast to Algorithm 9, Interests received from other routers are not aggregated.

If the Interest must be forwarded and an entry exists in $DART^i$ with router k as the predecessor and $dart^I(k)$ as the predecessor dart, then DEAR has been satisfied by a previous Interest from k over the existing path that k is requesting to use. Accordingly, router i simply swaps $dart^I(k)$ for the successor dart stated in the entry in $DART^i$ and forwards the Interest.

Alternatively, if no DART entry exists with k as a predecessor and $dart^I(k)$ as the predecessor dart, router i tries to find a neighbor that satisfies DEAR. The highest-ranked router v satisfying DEAR is selected as the successor for the Interest and router i creates a successor dart different from any other successor darts in $DART^i$, stores an entry with v and the new successor dart, and forwards the Interest to v. If DEAR is not satisfied, then router i sends a NACK back to router k.

4.2.5 Supporting Multipoint Communication

NDN and CCNx support multicasting by the reverse path forwarding (RPF) of Data packets over paths traversed by aggregated Interests. Interests serve the dual purpose of maintaining multicast forwarding trees (MFT) and pacing multicast sources. CCN-DART also supports multipoint communication using the RPF approach, but separates the establishment of an MFT from the mechanisms used to pace a source or disseminate multicast data over the tree.

CCN-DART uses RCTs and multicast data answer routing tables (MDART) to maintain MFTs. A single dart is used to denote all the predecessors and successors in the MFT of a group at each router . This means that a single dart must be used to label all the branches of the MFT of a multicast group. The dart used for multicast group name $g(j)$ is denoted by $d[g(j)]$, and can be made part of the group name to simplify its dissemination.

A router with local receivers of a multicast group maintains the mapping of the names of local receivers to the name of the multicast group in its RCT. The MDART at router i is denoted by $MDART^i$ and is indexed by the names of the multicast groups for which the router forwards traffic. The entry for multicast group with name $g(j)$ in $MDART^i$ states: the dart of the group ($d[g(j)]$), the successor selected by router i to join the group, the set of routers (predecessors) that requested to join $g(j)$ through router i, and the hop-count distance to the anchor of $g(j)$ when router i established the MDART entry for the group ($h^i(g(j))$).

If router i has local receivers for group $g(j)$, then it sends a join request (JR), denoted by $JR[g(j), h^J(i), dart^J(i)]$, stating the name of the group, $h^J(i) = h^i(g(j))$, and $dart^J(i) = d[g(j)]$. The forwarding of JRs is based on FIB entries and is similar to the forwarding of Interests. A relay router can forward a JR towards the anchor of $g(j)$ in two cases. If no MDART entry exists and DEAR is satisfied, an MDART entry is created for the group. If an MDART entry exists, then the router simply adds a new predecessor for the group in the existing MDART entry. Negative acknowledgments may be sent if no routes to $g(j)$ are found, DEAR is not satisfied, or MDART entries become invalid due to topology changes.

The design of algorithms for multicast data dissemination or pacing of multicast sources is outside the scope of this dissertation. However, similar approaches to those designed to address the ACK implosion problem in reliable multicasting [38] can be used to pace sources and pull or push data over multicast trees. Multicast data packets have the same format of Data packets and are forwarded from sources towards receivers based on the darts of the multicast groups.

4.2.6 CCN-DART Forwarding Example

Figure 4.6 illustrates how darts are used to label Interests and associate Interests with Data packets and NACKs. In the example, routers a , b, and x have local consumers originating Interests, and those Interests are assumed to request COs with names that are best matched with name prefixes for which router d is an anchor.

The arrowheads in the links of the figure denote the next hops stored in the FIBs of routers, and $y(i)$ denotes the ith dart in $DART^y$. The figure shows the DART entries maintained at all routers for name prefixes for which router d is an anchor, and the RCT entries stored at routers a, b, and x. Consumers *A, C, N,* and P request the same CO with name $n(j)$, and router a aggregates their requests and needs to send only one Interest for $n(j)$ towards d. Similarly, it aggregates the Interests from consumers *A, C,* and Q. Similar Interest aggregation of local requests occur at routers b and x.

Router a forwards Interests intended for anchor d to neighbor r, and routers b and x forwards their Interests to neighbors s and c, respectively. Routers a, r, and s establish the following mappings in their DARTs: $[a; a(k)] \leftrightarrow [r; a(k)]$ at a, $[a; a(k)] \leftrightarrow [s; r(m)]$ at

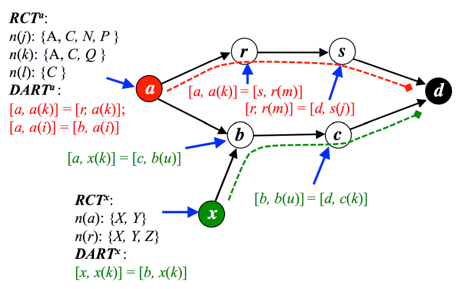


Figure 4.6: Interest forwarding in CCN-DART.

r, and $[r; r(m)] \leftrightarrow [d; s(j)]$ at s. These mappings denote the route (a, r, s, d) uniquely. Similarly, routers establish the DART mappings shown in the figure that denote the routes (x, b, c, d) and (b, c, d).

All the Interests from consumers local to routers a, b, and x regarding COs with names in prefixes for which d is an anchor can be routed towards d using the same few darts shown. Given that a Data packet or NACK specifies the successor dart stated the Interest it answers, Data packets and NACKs can be forwarded correctly from d (or a router caching the requested CO along the way to d) to routers a, b, or x unambiguously. In turn, routers a, b, and x can determine how to deliver the responses to local consumers based on the the RCT entries mapping each CO requested with the names of the customers that requested them.

4.3 Correctness of CCN-DART

The following two theorems show that CCN-DART operates correctly. Theorem 5 shows that CCN-DART prevents Interests from being propagated along loops, independently of whether the topology is static or dynamic or the FIBs are consistent or not.

To discuss the correctness of Interest forwarding in CCN-DART, we say that a forwarding loop of h hops for a CO with name $n(j)$ occurs when Interests requesting the CO are forwarded by routers along a cycle $L = \{v_1, v_2, ..., v_h, v_1\}$, such that router v_k receives an Interest for CO $n(j)$ from v_{k-1} and forwards the Interest to v_{k+1}, with $1 \leq k \leq h$, $v_{h+1} = v_1$, and $v_0 = v_h$.

Theorem 5. *Interests cannot traverse forwarding loops in a network in which CCN-DART is used.*

Proof. Consider a network in which CCN-DART is used. Assume for the sake of contradiction that routers in a forwarding loop L of h hops $\{v_1, v_2, ..., v_h, v_1\}$ send Interests for $n(j)$ along L, with no router in L detecting the incorrect forwarding of any of the Interests sent over the loop.

Given that L exists by assumption, $v_k \in L$ must send $I[n(j), h^I(v_k), dart^I(v_k)]$ to router $v_{k+1} \in L$ for $1 \leq k \leq h-1$, and $v_h \in L$ must send $I[n(j),\ h^I(v_h),\ dart^I(v_h)]$ to router $v_1 \in L$.

For $1 \leq k \leq h-1$, let $h(v_k, n(j)^*)^L$ denote the value of $h^I(v_k)$ when router v_k sends $I[n(j),\ h^I(v_k),\ dart^I(v_k)]$ to router v_{k+1}, with $h(v_k,\ n(j)^*)^L = h(v_k, n(j)^*, v_{k+1})$. Let $h(v_h, n(j)^*)^L$ denote the value of $h^I(v_h)$ when router v_h sends $I[n(j),\ h^I(v_h),\ dart^I(v_h)]$ to router $v_1 \in L$, with $h(v_h,\ n(j)^*)^L = h(v_h, n(j)^*,\ v_1)$.

Because no router in L detects the incorrect forwarding of an Interest and forwarded Interests are not aggregated in CCN-DART, each router in L must send its own Interest as a result of the Interest it receives from the previous hop in L. This implies that router $v_k \in L$ must accept $I[n(j), h^I(v_{k-1}),\ dart^I(v_{k-1})]$ for $1 \leq k < h$, and router $v_1 \in L$

must accept $I[n(j), h^I(v_h),\ dart^I(v_h)]$.

According to DEAR, if router v_k sends Interest $I[n(j),\ h^I(v_k),\ dart^I(v_k)]$ to router v_{k+1} as a result of receiving $I[n(j), h^I(v_{k-1}),\ dart^I(v_{k-1})]$ from router v_{k-1}, then it must be true that $h^I(v_{k-1}) > h(v_k, n(j)^*)^L = h^I(v_k)$ for $1 < k \leq h$. Similarly, if router v_1 sends $I[n(j), h^I(v_1),\ dart^I(v_1)]$ to router v_2 as a result of receiving $I[n(j),\ h^I(v_h),\ dart^I(v_h)]$ from router v_h, then $h^I(v_h) > h(v_1,\ n(j)^*)^L = h^I(v_1)$.

It follows from the above argument that, for L to exist and be undetected when each router in the loop uses DEAR to create DART entries, it must be true that $h^I(v_h) > h^I(v_1)$ and $h^I(v_{k-1}) > h^I(v_k)$ for $1 < k \leq h$. However, this is a contradiction, because it implies that $h^I(v_k) > h^I(v_k)$ for $1 \leq k \leq h$. Therefore, the theorem is true. □

Theorem 2 addresses the ability for routers to send Data packets and NACKs correctly to the consumers who issued the corresponding Interests using only the information stated in messages and stored in DARTs and RCTs. The theorem assumes that transmissions are sent correctly.

Theorem 6. *CCN-DART ensures that, in the absence of failures, Data packets and NACKs are sent correctly to the consumers that submitted the corresponding Interests.*

Proof. If a router can resolve an Interest from a local consumer, it follows from Algorithm 1 that the result is true. Let router s be the origin of an Interest and let router $d \neq s$ be the router that replies to the Interest from s with a Data packet or a NACK.

As Theorem 1 shows, Interests cannot traverse forwarding loops. Accordingly, if router d receives the Interest originated by s, then router d and all routers in the simple path from s to d must have established forwarding state according to Algorithm 4. Each router uses a different successor dart for each path traversing the router and defined by a predecessor name and a predecessor dart; therefore, each DART entry at a router uniquely denotes a different route traversing the router. Accordingly, given that router d can respond to an Interest only it it receives the Interest correctly, the proof can assume that the routers along the path from the source s to router d have established correct forwarding state in their DARTs. The rest of the proof must show that each router from d to s is able to demultiplex correctly the Data packets and NACKs that traverse the reverse path established by Interests delivered from s to d.

Let h be the number of hops in the path traversed by a Data packet or a NACK in response to an Interest originated at router s and answered by router d, and let f_i denote the router at the ith hop in such a path.

Basis Case: Let $h = 1$, then $s = f_1$. Router s labels its Interest with a dart assigned uniquely for its one-hop route to d, and remembers the user(s) that requested any CO in its RCT. According to Algorithm 4, router d responds to the Interest from s directly and includes the dart from the Interest in its response. According to Algorithms 2 and 3, router s associates the CO name in the response with the local consumer(s) that submitted requests for that CO. It follows that the basis case is true.

Inductive Step: Assume that each router up to $k-1$ hops away from router d in the path from router d to router s receives and forwards a Data packet or NACK from router d correctly over the path to s. We need to show hat the result is true for the router that is k hops away from router d in the path to router s, with $1 \leq k \leq h$.

When router f_k receives a Data packet or NACK from router f_{k-1} containing

$dart^I(f_{k-1})$, it uses Algorithm 2 or 3, respectively. Given that routers have established correct forwarding state in their DARTs, f_{k-1} and $dart^I(f_{k-1})$) must be the successor and successor dart in an entry in $DART^{f_k}$, respectively. Furthermore, the predecessor of the entry in $DART^{f_k}$ must equal either f_k or f_{k+1}. In the first case ($k = h$ and $s = f_k$), router f_k was the origin of the Interest being answered. In the second case ($k < h$), router f_k forwards the response to router f_{k+1} swapping $dart^I(f_{k-1})$ for the predecessor dart listed in the DART entry, which is the dart that router f_{k+1} included in the Interest it sent to router f_k. Hence, router f_k must forward its response correctly to either the previous hop in the path from d to s or the origin of the Interest.

It follows by induction that a Data packet or NACK traverses correctly the path of length h hops from router d to router s. Furthermore, the name of the CO in the Data packet or NACK is the one stated in the Interest originated by router s on behalf of one or multiple local consumers. Because s uses its RCT to associate each CO name stated in a Data packet or NACK with the correct set of local consumers that requested the CO, router s can forward the response to the correct local consumers. Therefore, the theorem is true. □

4.4 Performance Comparison

We compare the performance of CCN-DART and NDN using simulation experiments based on implementations of NDN and CCN-DART in the ndnSIM simulation tool [1].

The NDN implementation was used without modifications, and CCN-DART was implemented based on Algorithms 1 to 4. The performance metrics used to compare NDN with CCN-DART are the number of entries needed in PITs and DARTs, the number of Interests handled by routers, and the delay incurred in obtaining a Data packet or a NACK in response to an Interest.

The simulation parameters used for this study are the same as those presented in Section 4.1.1. We set the data rates of the links to 1Gbps to minimize the effects that link congestion or a sub-optimal implementation of CCN-DART or NDN may have on the results, especially for the case of end-to-end delays. The assumption that each router is locally attached to a content producer and local users requesting content constitutes the worst-case scenario for CCN-DART compared to NDN, because it results in many more DART entries. In a realistic deployment, only a small subset of the total number of routers in the network are attached to local producers of content. We consider on-path caching and edge caching. For the case of on-path caching, every router on the path traversed by a Data packet from the producer to the consumer caches the CO. On the other hand, with edge caching, only the router directly connected to the requesting consumer caches the resulting CO.

4.4.1 Size of PITs and DARTs

Figure 4.7 shows the average size and standard deviation of the number of entries in PITs used in NDN and the number of entries in DARTs and RCTs used in CCN-DART as a function of the total content-request rates per router. The vales shown for RCTs represent only the number of local pending Interests; the number of COs cached locally is not shown, given that the number of such entries would be very large and would be the same for NDN and CCN-DART.

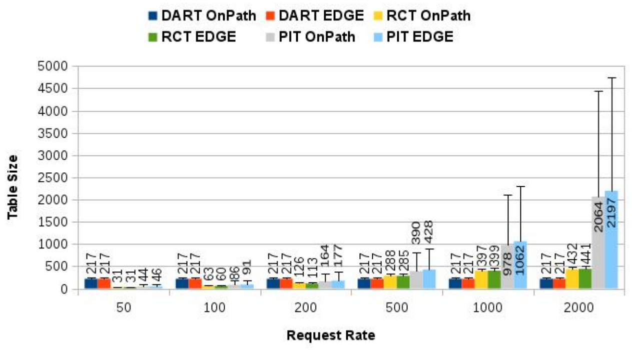


Figure 4.7: Size of PITs, DARTs and RCTs

As the figure shows, the size of PITs grows dramatically as the rate of content requests increases, which is expected given that PITs maintain per-Interest forwarding state. By contrast, the size of DARTs remains constant with respect to the content-request rates. The small average size of RCTs compared to the average size of PITs indicates that the average size of a PIT is dominated by the number of Interests a router forwards from other routers.

For low request rates, the average number of entries in a DART is actually larger than in a PIT. This is a direct consequence of the fact that a PIT entry is deleted immediately after an Interest is satisfied, while a DART entry is kept for long periods of time (seconds) in our implementation, independently of whether or not it is used to forward Interests. Given the small sizes of DARTs, the cost of maintaining DART entries that may not be used at light loads is more than compensated by the significant reduction in forwarding state derived from many Interests being forwarded using existing DART entries at higher request rates. This should be the case in real deployments, where the number of routers that are origins of routes to prefixes is much smaller than the total number of routers. However, optimizing the length of time that a DART entry lasts as a function of its perceived utility for content forwarding is an area that deserves further study.

As the total content-request rate per router increases, the size of a PIT can be more than 10 to 20 times the size of a DART, because a given DART entry is used for many Interests in CCN-DART, while NDN requires a different PIT entry for each Interest. It is also interesting to see the effect of on-path caching compared to edge-caching. The average size of DARTs is independent of where content is being cached, and on-path caching in NDN does not make a significant difference in the size of a PIT compared to edge-caching.

4.4.2 Interest Traffic and End-to-End Delays

Figure 4.8 shows the average number of interests received by each router in NDN and CCN-DART as a function of the content request rates for on-path caching and edge caching. The number of Interests received in CCN-DART is larger than the corresponding number for NDN. However, it is clear from the figure that the average numbers of Interests received by each router in NDN and CCN-DART are almost the same for all request rates.

The benefit of on-path caching is apparent for both NDN and CCN-DART, but appears slightly more pronounced for the case of CCN-DART. This should be expected, because CCN-DART does not aggregate Interests and on-path caching results in fewer Interests being forwarded.

Figure 4.9 shows the average end-to-end delay for NDN and CCN-DART as a function of content-request rates for on-path caching and edge caching. As the figure shows, the average delays for NDN and CCN-DART are essentially the same for all content-request

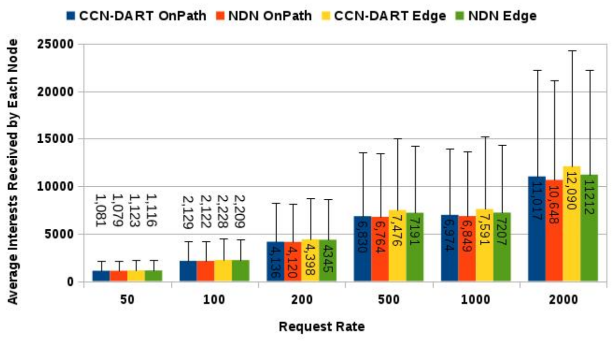


Figure 4.8: Number of Interests received by routers

rates. This should be expected, given that in the experiments the routes in the FIBs are static and loop-free, and the number of Interests processed by routers is similar.

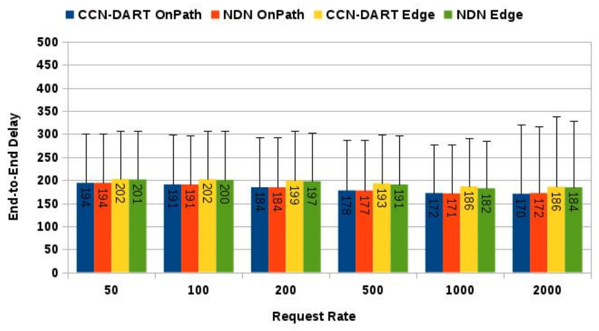


Figure 4.9: Average end-to-end delays

4.4.3 Impact of Interest Flooding Attacks

Information Centric Networks can be a target for attacks known as Interest Flooding which is a type of Distributed Denial of Service attacks. In this kind of attack, attackers generate interests messages for valid prefixes in a high rate. The goal of such attacks is to overwhelm PIT of routers, preventing them from handle legitimate interests. The other goal would be to swamp content producers [78]. Generated interests can be for existing or non-existent data objects. For the first case, where interests can be satisfied, PIT entries will not remain for long time, and they will be satisfied by data objects. For the requetes generated repeatedly for same objects, attack will not be effective since requests will be satisfied by the content stores. For the case of non-existent data objects, generated interests are routed toward content producers, resulting in creation of PIT entries, which will remain in PIT tables till they get expired, which can affect all routers from acting properly since loose state of traffic and can not serve legitimate traffic due to occupancy of PIT by non-legitimate entries. Looseing state will result defeat of other protocols such as forwarding, congestion control, and etc which perform based on state of routers and traffic.

We ran simulations for three cases. The first case serves as the baseline; all routers receive 200 valid Interests per second from local users. In the second case, each router with local attackers receives 2000 invalid Interests per second. The resulting average size of the PITs almost doubles with respect to the case of no attacks, and the average size of DARTs does not change. In the third case, each router with local attackers receives 4000 invalid Interests per second, the average size of the PIT table increases dramatically, and the size of the DART still remains the same.

Interest flooding attacks in NDN translate into PITs that can easily be overwhelmed and much more traffic (Interests and NACKs). It is important to note that, by the very nature of Interest flooding attacks, Interest aggregation is not useful. Given the results from the other experiments and Section 4.1.1, it is clear that using PITs to maintain forwarding state is to the detriment of the system.

It is clear from the results of this experiment that the size of DARTs is not affected by the presence of Interest flooding attacks. The size of RCTs grows for those routers with local attackers in the presence of Interest flooding attacks, because RCTs maintain per-Interest state to enable the aggregation of local requests. However, limiting the size of RCTs can be done by limiting the rate at which any one local consumer is allowed to submit content requests, which impacts only that specific local user.

CCN-DART eliminates a major vulnerability of NDN, because forwarding tables cannot be attacked. However, an approach is still needed to address Interest flooding attacks in CCN-DART, given that invalid Interests still consume valuable bandwidth and can overwhelm content providers. Any viable solution to Interest flooding attacks requires RCTs (or Content Stores in the NDN case) to act as filters of valid requests.

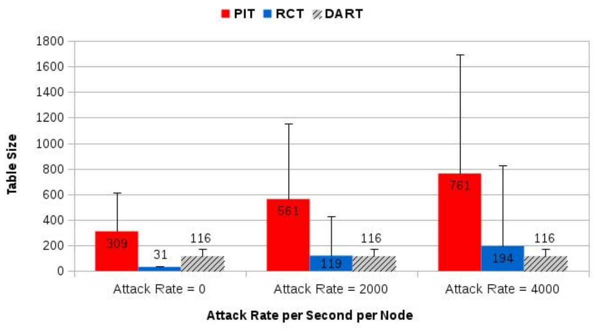


Figure 4.10: Table Sizes under DDoS Attack

4.4.4 Implementation and Deployment

The mappings stored in DARTs are equivalent to the label mappings first introduced for packet switching based on virtual circuits [39] and used today in high-performance routers running multiprotocol label switching (MPLS). The small fixed-size darts and the relatively small number of DART entries needed for CCN-DART to operate at Internet scale are preferable by far to the long variable-length names (plus large nonces for the case of NDN) and the large number of PIT entries needed to maintain forwarding state in NDN and CCNx [33, 42].

Both DARTs and PITs must be updated when the paths traversed by Interests and their responses must change due to congestion, topology changes, or mobility of consumers and providers. Yi et al [24] argue that a stateful forwarding plane enables a fast response to topology changes and congestion. However, the existence of MPLS fast rerouting mechanisms demonstrate that maintaining per-Interest forwarding state is not necessary to enable fast restoration of paths in the data plane. Similar mechanisms can be adopted in CCN-DART with much less signaling overhead than attempting to update forwarding tables with forwarding state for each Interest. Furthermore, approaches similar to those introduced in the past for congestion-oriented multipath routing and dynamic load balancing [21, 44] can be used in the context of CCN-DART, taking advantage of the fact that Interests cannot traverse loops.

CCN-DART provides native support for single-source multicasting. However, it separates the maintenance of multicast forwarding trees from the mechanisms used for source pacing and data dissemination over such trees. The benefit of this separation is that both pull and push-based mechanisms for multipoint communication can be used, and both are important for content-centric networking [31].

End users cannot mount Interest flooding attacks [114, 43] to overflow DARTs in the same way that PITs can be attacked. A DART entry can be added only for valid anchors of name prefixes and for routes that satisfy the ordering constraint imposed with

DEAR. Given that both conditions are managed in the control plane, mounting attacks on DARTs must be much more sophisticated than simply having users send Interests for COs corresponding to valid prefixes.

Countermeasures to DART attacks can be implemented based on configuration data or information protected in the control plane. For example, a neighbor router can be designated as an edge router and be limited to creating one dart per anchor. Similarly, Interests from neighbors must satisfy DEAR or be rejected, and an upper bound on the number of darts allowed from each neighbor for any prefix can be set based on the network size and the maximum number of routes to an anchor that can flow through that neighbor.

The performance results for edge and on-path caching we have presented have important consequences for CCN-DART deployments. An efficient deployment of CCN-DART could consist of using full content routers with FIBs, DARTs and RCTs only at the edge, and using "dart routers" elsewhere, which are are dedicated to content forwarding and maintain only FIBs and DARTs.

4.5 Conclusions

We introduced CCN-DART, the first approach to Interest-based content-centric networking that supports Interest forwarding without revealing the sources of Interest and with no need to maintain forwarding state on a per-Interest basis.

CCN-DART replaces PITs with Data Answer Routing Tables (DART) that establish forwarding state for each route traversing the router over which many Interests are multiplexed, rather than establishing state for each different Interest using routes traversing the router.

We proved that forwarding loops cannot occur in CCN-DART, even if routing loops exist in the FIBs maintained by routers, and that Data packets and NACKs are forwarded over the correct paths to consumers. The results of simulation experiments based on implementations of NDN and CCN-DART in ndnSIM show that CCN-DART is far more efficient than NDN. Compared to NDN, CCN-DART rendered the same end-to-end delays, incurred similar signaling overhead in the data plane, and resulted in a forwarding state with a number of entries smaller than one order of magnitude the number required in NDN.

Algorithm 12 Processing Interest from router k at router i

```
function Dart_Swapping
INPUT: $RCT^i$, $FIB^i$, $DART^i$, $I[n(j), h^I(k), dart^I(k)]$;
if $n(j) \in RCT^i \wedge p[n(j)] \neq nil$ then
    retrieve CO $n(j)$; send $DP[n(j), sp(j), dart^I(k)]$ to $k$
else
    (% $n(j) \notin RCT^i \vee p[n(j)] = nil$ )
    if $n(j)^* \in RCT^i$ then
        send $NA[n(j), \textsf{no content}, dart^I(k)]$ to $k$
    else
        if $n(j)^* \notin FIB^i$ then
            send $NA[n(j), \textsf{no route}, dart^I(k)]$ to $k$
        else
            if $\exists DART^i(a,k)$ ( $pd^i(a,k) = dart^I(k)$ ) then
                $h^I(i) = h^i(a,k)$; $dart^I(i) = sd^i(a,k)$;
                send $I[n(j), h^I(i), dart^I(i)]$ to $s^i(a,k)$
            else
                for each $v \in S^i_{n(j)^*}$ by rank in $FIB^i$ do
                    if $h^I(k) > h(i, n(j)^*, v)$ (% DEAR is satisfied) then
                        $a = a(i, n(j)^*, v)$;
                        create entry $DART^i(a,k)$:
                        compute $SD \neq sd^i(p,q)\ \forall\ DART^i(p,q)$;
                        $h^i(a,k) = h(i, n(j)^*, v)$;
                        $p^i(a,k) = k$; $pd^i(a,k) = dart^I(k)$;
                        $s^i(a,k) = v$; $sd^i(a,k) = SD$;
                        create Interest:
                        $h^I(i) = h^i(a,k)$; $dart^I(i) = sd^i(a,k)$;
                        send $I[n(j), h^I(i), dart^I(i)]$ to $v$;
                        return
                    end if
                end for (% Interest may be traversing a loop)
                send $NA[n(j), \textsf{loop}, dart^I(k)]$ to $k$
            end if
        end if
    end if
end if
```

Chapter 5

Content Centric Networking Using Anonymous Datagrams

The leading approach in content-centric networking consists of: populating forwarding information bases (FIB) maintained by routers with routes to name prefixes denoting content, sending content requests (called Interests) for specific content objects (CO) over paths implied by the FIBs, and delivering data packets with content objects along the reverse paths traversed by Interests.

The main advantages that such Interest-based content-centric networking approach offers compared to the IP Internet are that: (a) content providers and caching sites do not know the identity of the consumers requesting content; (b) content can be obtained by name from those sites that are closer to consumers; (c) data packets carrying content cannot traverse loops, because they are sent over the reverse paths traversed by Interests; and (d) content-oriented security mechanisms can be implemented as part of the content delivery mechanisms.

Named data networking (NDN) [17] and CCNx [4] are the two prominent Interest-based content-centric networking approaches. Routers in NDN and CCNx maintain a "stateful forwarding plane" [24] (i.e., per-Interest forwarding state) by means of Pending Interest Tables (PIT). The PIT of a router maintains information regarding the incoming interfaces from which Interests for a CO were received and the interfaces where the Interest for the same CO was forwarded.

Since the inception of CCNx and NDN, PITs have been viewed as necessary in order to maintain routes to the origins of Interests while preserving the anonymity of those sources, aggregate Interests requesting the same content in order to attain efficient Interest and content forwarding, and support multicasting without additional support in the control plane.

However, using PITs at Internet scale comes at a big price. PITs grow very large [33, 41, 42] as the number of Interests from users increases, which results from PITs having to store per-Interest forwarding state. Furthermore, PITs make routers vulnerable to Interest-flooding attacks [78, 43, 112, 113] in which adversaries send malicious Interests aimed at making the size of PITs explode. Known countermeasures to these attacks [114] attempt to reduce the rates at which suspected routers can forward Interests. However, these solutions cannot prevent all flooding attacks and can actually be used to mount other types of denial-of-service attacks.

Section 4.1.1 analyzes the effectiveness of Interest aggregation in NDN by means of simulations based on the implementation of NDN in ndnSIM [1] without modifications.

The results show that the percentage of Interests that are aggregated is negligible when in-network caching is enabled, even when Interests exhibit temporal or spatial correlation.

Given that in-network caching obviates the need for Interest aggregation, and given the vulnerability of NDN and CCNx to Interest-flooding attacks, it is clear that a new Interest forwarding approach is needed for content-centric networking.

We present **CCN-GRAM** (*Gathering of Routes for Anonymous Messengers*), which provides all the benefits of content-centric networking, including native support for multicasting in the data plane, and eliminates the need to maintain per-Interest forwarding state by forwarding Interests and responses to them using *anonymous* datagrams.

Section 7.1.1 describes the operation of CCN-GRAM. Like NDN and CCNx, CCN-GRAM uses Interests, data packets, and replies to Interests. Similar to IP datagrams, the messages sent in CCN-GRAM specify a source and a destination. For an Interest, the source of an Interest is an anonymous identifier with local context and the destination is the name of a content object. For data packets and replies to Interests, the source is the name of a content object and the destination is an anonymous identifier. A novel on-demand routing approach is used to maintain routes to the anonymous routers that originate Interests for specific content on behalf of local content consumers. Only the local router serving a user knows the identity of the user; no other router, content provider, or caching site can determine the consumer that originated an Interest, without routers collaborating along the path traversed by the Interests to establish the provenance of the Interest.

In contrast to NDN and CCNx in which Interests may traverse forwarding loops [36, 37, 120], forwarding loops cannot occur in CCN-GRAM for either Interests or responses sent to Interests, even if the FIBs maintained by routers contain inconsistent forwarding state involving routing-table loops. Furthermore, the anonymous datagram forwarding of CCN-GRAM is much simpler than the label-swapping approach we have advocated before [108].

Forwarding of Interests and responses to them in CCN-GRAM uses four tables: a LIGHT (Local Interests GatHered Table), a FIB, an ART (Anonymous Routing Table) and a LIST (Local Interval Set Table). The LIGHT of a router is an index listing content that is locally available and content that is remote and has been requested by *local* users. The FIB of each router states the next hops to each name prefix and the distance to the name prefix reported by each next hop. The ART is maintained using Interests and states the paths to destinations denoted with local identifiers from which routers cannot discern the origin of Interests. The LIST states the intervals of local identifiers that a router assigns to its neighbors and that each neighbor assigns to the router.

Section 7.2 compares the performance of CCN-GRAM with NDN when routes to name prefixes are static and loop-free, which is the best case for NDN. The network consists of 150 routers, with 10 being connected to content producers and 50 being connected to consumers. CCN-GRAM attains similar end-to-end latencies than NDN in retrieving content. However, depending on the rate at which Interests are submitted, CCN-GRAM requires an average number of forwarding entries per router that is 5 to more than150 times smaller than the number of PIT entries needed in NDN.

5.1 CCN-GRAM

We assume that Interests are retransmitted only by the consumers that originated them. We assume that routers use exact Interest matching, and that a router that advertises being an origin of a name prefix stores all the content objects associated with that prefix at a local content store. Routers know which interfaces are neighbor routers and which are local users, and forward Interests on a best-effort basis. For convenience, it is assumed that a request for content from a local user is sent to its local router in the form of an Interest.

5.1.1 Information Exchanged and Stored

The name of content object (CO) j is denoted by $n(j)$ and the name prefix that is the best match for name $n(j)$ is denoted by $n(j)^*$. The set of neighbors of router i is denoted by N^i.

An Interest forwarded by router k requesting CO $n(j)$ is denoted by $I[n(j), AID^I(k), D^I(k)]$, and states the name of the requested CO $(n(j))$, an anonymous identifier $(AID^I(k))$ used to denote the origin of the Interest, and the distance from k to the requested content.

A data packet sent by router i in response to an Interest is denoted by $DP[n(j), AID^R(i), sp(j)]$, and states the name of the CO being sent $(n(j))$, an anonymous identifier $(AID^R(i))$ that states the intended recipient of the data packet, and a security payload $(sp(j))$ used optionally to validate the CO.

A reply sent by router i in response to an Interest is denoted by $REP[n(j), AID^R(i), \mathsf{CODE}]$ and states the name of a CO $(n(j))$, an anonymous identifier $(AID^R(i))$ that states the intended recipient of the reply, and a code (CODE) indicating the reason why the reply is sent. Possible reasons for sending a reply include: an Interest loop is detected, no route is found towards requested content, and no content is found.

Router i maintains four tables for forwarding: an optional Local Interests Gathered Table ($LIGHT^i$), a forwarding information base (FIB^i), an anonymous routing table (ART^i), and a Local Interval Set Table ($LIST^i$).

$LIGHT^i$ lists the names of the COs requested by router i or already stored at router i. It is indexed by the CO names that have been requested by the router on behalf of local customers. The entry for CO name $n(j)$ states the name of the CO ($n(j)$), a pointer to the content of the CO ($p[n(j)]$), and a list of zero or more identifiers of local consumers ($lc[n(j)]$) that have requested the CO while the content is remote.

FIB^i is indexed using known content name prefixes. The entry for prefix $n(j)^*$ states the distance reported by each next-hop neighbor router for the prefix. The distance stored for neighbor q for prefix $n(j)^*$ in FIB^i is denoted by $D(i, n(j)^*, q)$. Each entry in FIB^i is stored for a maximum time determined by the lifetime of the corresponding entry in the routing table of the router.

$LIST^i$ maintains the intervals of anonymous identifiers used by router i. It states the local interval of identifiers accepted by router i (denoted by $LI^i(i)$), and the local interval of identifiers accepted by each neighbor router k (denoted by $LI^i(k)$) . Clearly, $LI^i(k) = LI^k(k)$. All local intervals have the same length $|LI|$.

ART^i is indexed using the anonymous identifiers taken from $LI^i(i)$. Each entry states an anonymous identifier of a destination ($AID(ART^i)$), a next hop to the destination, ($s(ART^i)$), and an identifier mapping used to handle identifier collisions ($map(ART^i)$).

$ART^i[AID, s, map]$ is used to denote a given entry in ART^i.

Routers can exchange local intervals with their neighbors in a number of ways. The exchange can be done in the data plane using Interests and data packets. An example would be having a router send an Interest stating a common name denoting that a local interval is requested, and an empty AID. Given the succinct way in which local intervals can be stated (an identifier denotes its interval), the exchange can also be easily done as part of the routing protocol running in the control plane. Routers could exchange interval identifiers in HELLO messages, link-state advertisements or distance updates. To simplify our description of CCN-GRAM, we assume that routers have exchanged their local intervals with one another and have populated their LISTs accordingly. We also assume that local intervals do not change for extended periods of time after they are assigned.

5.1.2 Eliminating Forwarding Loops

Let $S^i_{n(j)^*}$ denote the set of next-hop neighbors of router i for prefix $n(j)^*$. The following rule is used to ensure that Interests cannot traverse routing loops, even if the routing data stored in FIBs regarding name prefixes is inconsistent and leads to routing-table loops.

Loop-Free Forwarding Rule (LFR):

Router i accepts $I[n(j), AID^I(k), D^I(k)]$ from router k if:

$$\exists\ v \in S^i_{n(j)^*}(\ D^I(k) > D(i, n(j)^*, v)\ ) \tag{5.1}$$

LFR is based on the same invariants we have proposed previously to eliminate Interest looping in NDN and CCNx [36, 120] and avoid forwarding loops in more efficient forwarding planes for content-centric networks [108]. As we explain in [36, 108, 120], the approach is a simple application of diffusing computations that ensures loop-free forwarding of Interests with or without aggregation.

5.1.3 Forwarding to Anonymous Destinations

The header of a datagram needs to denote its origin and destination, so that the datagram can be forwarded to the intended destination and responses to the datagram can be forwarded back to the source. Since the introduction of datagram packet switching by Baran [117], the identifiers used to denote the sources and destinations of datagrams have had global scope, and routers maintain FIBs with entries towards those sources. However, this need not be the case!

It is trivial to add information in Interests about the paths they traverse to allow responses to be sent back without the need for FIBs maintaining routes to the sources of Interests. However, this would negate the anonymity of Interests advocated in NDN and CCNx.

CCN-GRAM uses local identifiers to denote the sources of Interests in a way that responses to Interests (data packets or replies) can be forwarded correctly to the sources of Interests, without their identity being revealed to relaying routers, caching sites, or content producers.

Given that all local intervals have the same length $|LI|$, the local interval $LI^i(i)$ is uniquely defined by the smallest identifier of the interval, which we denote by $LI^i(i)[s]$.

If router p sends Interest $I[n(j), AID^I(p), D^I(p)]$ to router i, $AID^I(p)$ must be in $LI^p(i) = LI^i(i)$. Similarly, if router i forwards Interest $I[n(j), AID^I(i), D^I(i)]$ to router n,

$AID^I(i)$ must be in $LI^i(n) = LI^n(n)$. Hence, to forward an Interest from p to n, router i must map the AID received in the Interest from p to an AID that belongs to the local interval accepted by its neighbor n. Router i can accomplish this mapping with the following bijection, where ϵ is a constant known only to router i:

$$AID^I(i) = \epsilon + AID^I(p) - LI^i(i)[s] + LI^i(n)[s] \ mod \ |LI| \tag{5.2}$$

We denote the mapping of identifiers from $LI^i(i)$ to $LI^i(n)$ by $f_i(n) : LI^i(i) \rightarrow LI^i(n)$. The image of identifier $a \in LI^i(i)$ under $f_i(n)$ is denoted by $f_i(n)[a]$ and $f_i(n)[a] \in LI^i(n)$. The reverse mapping from $LI^i(n)$ to $LI^i(i)$ is denoted by $f_i^{-1}(n)$ and of course $f_i^{-1}(n)[f_i(n)[a]] = a$.

Algorithms 17 to 16 specify the steps taken by routers to process and forward Interests, and return data packets or replies. We assume that each router is initialized properly, knows the identifiers used to denote local consumers, knows all its neighbors, and knows the local identifier intervals associated with each neighbor. We assume that a routing protocol (e.g., DCR [10], NLSR [125]) operating in the control plane updates the entries of routing tables listing one or multiple next hops towards name prefixes. Routers populate their FIBs with routes to name prefixes based on the data stored in their routing tables. How long FIB entries are maintained is determined by the operation of the routing protocol.

We assume that router i uses a single anonymous identifier in $LI^i(i)$ to denote itself in its ART^i, and denote it by AID^i.

Algorithm 17 shows the steps taken by router i to process Interests received from local consumers. For convenience, content requests from local consumers are assumed to be

Algorithm 13 Processing Interest from user c at router i

```
function Interest_Source
INPUT: $LIGHT^i$, $LIST^i$, $FIB^i$, $ART^i$, $AID^i$, $I[n(j), c, nil]$;
if $n(j) \in LIGHT^i$ then
    if $p[n(j)] \neq nil$ (% CO is local) then
        retrieve CO $n(j)$; send $DP[n(j), c, sp(j)]$ to consumer $c$
    else
        $p[n(j)] = nil$; $lc[n(j)] = lc[n(j)] \cup c$ (% Interest is aggregated)
    end if
else
    if $n(j)^* \in LIGHT^i$ (%All content in $n(j)^*$ is local and $n(j)$ is not) then
        send $REP[n(j), c, \text{no content}]$ (% $n(j)$ does not exist)
    else
        if $n(j)^* \notin FIB^i$ then
            send $REP[n(j), \text{no route}, c]$ to $c$ (% No route to $n(j)^*$ exists)
        else
            create entry for $n(j)$ in $LIGHT^i$: (% Interest from $c$ is recorded)
            $lc[n(j)] = lc[n(j)] \cup c$; $p[n(j)] = nil$;
            if $AID^i = nil$ then
                select identifier $a \in LI^i(i)$ that is not used in any entry in $ART^i$;
                $AID^i = a$; create entry $ART^i[AID^i, i, AID^i]$
            end if
            for each $v \in N^i$ by rank in $FIB^i$ do
                $AID^I(i) = f_i(v)[AID^i]$; $D^I(i) = D(i, n(j)^*, v)$;
                send $I[n(j), AID^I(i), D^I(i)]$ to $v$; return
            end for
        end if
    end if
end if
```

Interests stating the name of a CO, the name of the consumer, and an empty distance to the content assumed to denote infinite. Similarly the same format of data packets and replies used among routers is used to denote the responses a router sends to local consumers.

After receiving an Interest from a local consumer, router i first searches its LIGHT to determine if the content is stored locally or a request for the same content is pending. If the content is stored locally, a data packet is sent back to the user requesting the CO. If a request for the same content is pending, the name of the user is simply added to the list of users that have requested the CO.

In our description of CCN-GRAM, a router that advertises being an origin of a prefix must have all the COs associated with the prefix stored locally. If router i states that it is an origin of the name prefix $n(j)^*$ and a specific CO with a name that is in that prefix is not found locally, a reply must be sent back to the consumer stating that the content does not exist. Additional steps could be taken to address the case of Interests sent maliciously for content that does not exist.

If the CO is remote and no FIB entry exists for a name prefix that can match $n(j)$, a reply is sent back stating that no route to the CO could be found. Otherwise, router i forwards the Interest through the highest ranked neighbor v in its FIB for the name prefix matching $n(j)$, which is denoted by $n(j)^*$. How such a ranking is done is left unspecified, and can be based on a distributed or local algorithm [10, 125, 111].

When router i originates an Interest on behalf of a local consumer and forwards Interest $I[n(j), AID^I(i), D^I(i)]$ to neighbor router n towards name prefix $n(j)^*$, router i selects an identifier $a \in LI^i(i)$ that is not used to denote any other source of Interests in ART^i, sets $AID^I(i) = f_i(n)[a] \in LI^i(n)$, and stores the entry $ART^i[a, i, a]$. Router i can use the same anonymous identifier for all the Interests it originates on behalf of local consumers and forwards to neighbor n.

If no ART entry exists with router i as the origin of Interests ($AID^i = nil$), AID^i is selected from the set of AIDs in $LI^i(i)$ that are not being used for other Interest sources, and a new ART entry is created for AID^i. The Interest is forwarded to the selected next hop for the Interest by first mapping AID^i into an AID in $LI^i(v)$ using the bijection in Eq.

5.2.

Algorithm 18 shows the steps taken by router i to process an Interest received from a neighbor router p. The main differences in the steps taken by router i compared to Interests received from local users are that no Interest aggregation is done for Interests received from neighbor routers, and router i maps the AID it receives in the Interest from the previous hop to the AID it should use in the Interest it sends to the next hop using a simple mapping function.

When router i forwards Interest $I[n(j), AID^I(p), D^I(p)]$ from predecessor router p to successor router n towards name prefix $n(j)^*$, router i makes sure that $AID^I(p) \in LI^i(i)$ is not listed in an ART^i entry with a next hop other than p. If that is the case, router i stores $ART^i[AID^I(p), p, AID^I(p)]$, and sets $AID^I(i) = f_i(n)[AID^I(p)] \in LI^i(n)$. Otherwise, router selects an AID $b \in LI^i(n)$ that is not used to denote any other source of Interests in ART^i, stores $ART^i[b, p, AID^I(p)]$, and sets $AID^I(i) = f_i(n)[b] \in LI^i(n)$.

If the requested content is cached locally, a data packet is sent back. If router i is an origin of $n(j)^*$ and the CO with name $n(j)$ is not found locally, a reply is sent back stating that the content could not be found. Additional steps can be taken to address the case of malicious Interests requesting non-existing content. If the CO is remote and no FIB entry exists for $n(j)^*$, then router sends a reply stating that no route could be found for the CO.

Router i tries to forward the Interest to a next hop s for the best prefix match for $n(j)$ that satisfies LFR. The highest-ranked router satisfying LFR is selected as the successor for the Interest and router i. If no neighbor is found that satisfies LFR, a reply is sent stating that a loop was found.

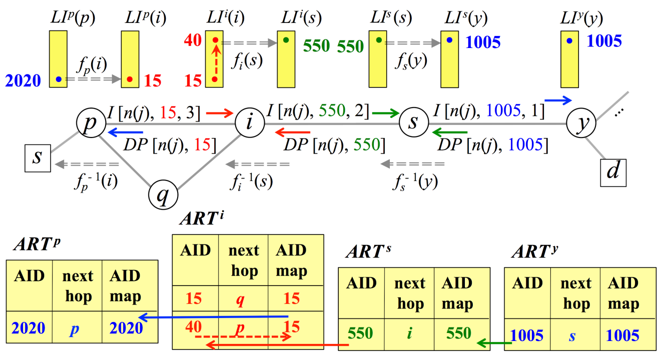
Algorithm 19 outlines the processing of data packets. If local consumers requested the content in the data packet, it is sent to those consumers based on the information stored in $LIGHT^i$. If the data packet is received in response to an Interest that was forwarded from router p, router i forwards the data packet doing the proper mapping of AIDs. Router i stores the data object if edge or on-path caching is supported.

When router i receives $DP[n(j), AID^R(n), sp(j)]$ from neighbor n, it obtains the AID of of the destination where the packet should be forwarded by computing $f_i^{-1}(n)[AID^R(n)]$. Router i uses entry $ART^i[f_i^{-1}(n)[AID^R(n)], p, m]$ to determine the next-hop neighbor p that should receive the data packet, and sets $AID^R(i) = m$.

Algorithm 16 states the steps taken to handle replies, which are similar to the forwarding steps taken after receiving a data packet. Router i forwards the reply to local consumers if it was the origin of the Interest, or to a neighbor router p if it has an ART entry with p as the next hop towards the destination denoted by the AID stated in the reply.

5.1.4 Example

Figure 5.1 illustrates the swapping of AIDs used by routers to forward Interests and responses to them. The local intervals used in the figure are small for simplicity, and the figure focuses on the forwarding state needed to forward Interests from p to name prefixes announced by router y, as well as the responses to such Interests. Interests are forwarded based on FIB entries, and responses to Interests (data packets or replies) are forwarded based on ART entries.



AID	next hop	AID map
2020	*p*	2020

AID	next hop	AID map
15	*q*	15
40	*p*	15

AID	next hop	AID map
550	*i*	550

AID	next hop	AID map
1005	*s*	1005

Figure 5.1: Forwarding of Interests and responses to them in CCN-GRAM

As illustrated in Figure 5.1, router i takes into account the possibility of collisions in the AIDs stated in Interests received from different neighbors by means of the identifier-mapping filed of ART entries. The bijection in Eq. 5.2 is used to map either the AID specified in the Interest received from neighbor p or the AID created by router i to handle collisions to the AID stated by router i in the Interest it forwards to a next-hop router s. In the example, router i has an exiting entry $ART^i[15, q, 15]$ when it receives Interest $I[n(j), 15, 3]$ from router $p \neq q$. Accordingly, router i selects $AID = 40$, creates entry $ART^i[40, p, 15]$, and sets $AID^I(i) = f_i(s)[40] = 550$ before forwarding Interest $I[n(j), 550, 2]$ to router s. When router i receives data packet $DP[n(j), 550, sp(j)]$ from router s, it computes $f_i^{-1}(s)[550] = 40$. Using $AID = 40$ as the key in ART^i, router i obtains the next hop p, sets $AID^R(i) = 15$, and forwards $DP[n(j), 15, sp(j)]$ to router p.

It is clear from the example that a router sending an Interest is unaware of collisions of AIDs at the next hop. The identifier mapping field of ARTs allows routers to multiplex Interests from different neighbors stating the same AID values.

Even when a very small number of routers is involved, only the router that originates an Interest is able to determine that fact, because the identifiers used for Interest forwarding are assigned by the next hops.

5.1.5 Native Support for Multicasting

Support of multicast communication in the data plane with no additional signaling required in the control plane is viewed as an important benefit derived from maintaining per-Interest forwarding state using PITs. In short, multicast receivers send Interests towards the multicast source. As Interests from receivers are aggregated in the PITs on their way to the multicast source, a multicast forwarding tree (MFT) is formed and maintained in the data plane. Multicast Interest are forwarded using the same FIB entries used for unicast traffic, and multicast data packets are sent using reverse path forwarding (RPF) over the paths traversed by aggregated Interests. Using PITs is appealing in this context; however, as we show below, native support of multicasting in the data plane can be easily done with no need for per-Interest forwarding state!

Information Stored and Exchanged We assume that the name stated in an Interest created to request content from a multicast source denotes a multicast source uniquely, and call such an Interest a *multicast Interest*. We also assume that consumers and routers differentiate between a multicast Interest and an Interest originated from a single consumer (unicast Interest).

A multicast Interest $MI[g(j), D^I(i), mc^I(i)]$ sent by router i to router n states: the name of a multicast group $g(j)$, the distance from router i to the source of the multicast group $D^I(i)$, and a multicast counter ($mc^I(i)$) used for pacing.

A multicast data packet $MP[g(j), sp(j), mc^R(i)]$ states the name of the multicast group $g(j)$, a security payload $sp(j)$, a multicast counter $mc^R(i)$, plus the content payload. A multicast reply $MR[g(j), CODE, mc^R(i)]$ states the reason for the reply and the current value of the multicast counter.

Router i maintains a multicast anonymous routing table ($MART^i$) that contains the forwarding state to the receivers of multicast groups. Each entry in $MART^i$ specifies a multicast group name, the value of the multicast counter (mc), and a list of next hops to the group of receivers who have sent Interests for the group. If router i has local receivers for group $g(j)$, the entry for the group in $MART^i$ includes router i as a next hop to the receivers of the group.

Router i also maintains a group membership table (GMT^i) that lists the mappings of multicast group names to the lists of local receivers that requested to join the groups. The GMT entries allow the router to deliver multicast content to local receivers of specific groups.

Multicast Content Dissemination The key difference of the way in which CCN-GRAM forwards multicast traffic compared to NDN or CCNx is that a MART maintains per-group forwarding state, while a PIT maintains per-Interest forwarding state. Figure 5.2 illustrates the forwarding of multicast Interests and multicast content in CCN-GRAM. There is no need for anonymous identifiers for multicast content forwarding, because all consumers of a group must receive the same multicast COs, which are forwarded using multicast group names.

A content consumer c requests to join a multicast group $g(j)$ as a receiver by sending a multicast Interest $MI[g(j), D^I(c), mc^I(c)]$ with $D^I(c) = nil$.

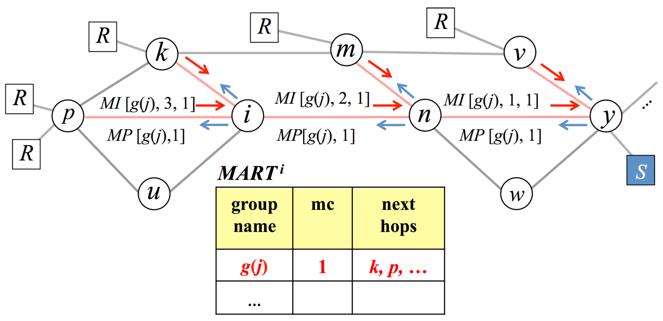


Figure 5.2: Native multicast support in CCN-GRAM

If router i has multiple local receivers or neighbor routers requesting to join the same multicast group $g(j)$, router i forwards multicast Interest $MI[g(j), D^I(i), mc^I(i)]$ only once towards the source of the multicast group $g(j)$ based on the information in its FIB. Router i simply adds new local consumers to the entry for $g(j)$ in GMT^i or new next hops to multicast receivers in $MART^i$.

Router i forwards multicast data packets based on the group names stated in the packets and the next hop stored in its MART entries, and discards the data packet if no MART entry exists for the multicast group. A similar approach is used for replies to Interests regarding multicast groups.

The dissemination of multicast data packets over the MFT of a multicast group can be of two types. A multicast source can push multicast data towards the receivers, or the receivers can pull data from the source by submitting Interests.

Push-based dissemination: The only forwarding state needed in CCN-GRAM for push-based multicast dissemination consists of the name of a multicast group and the names of the next hops towards the group receivers. In this mode, the *mc* value of an entry in a MART is updated with each multicast data packet forwarded by the router towards the receivers.

Pull-based dissemination: CCN-GRAM can also support pull-based multicast dissemination with no need for per-Interest forwarding state. An exemplary approach consists of a source-pacing algorithm based on the mc values carried in Interests and data packets. Each receiver increments the mc value of Interests it sends for the group asking for the next piece of multicast content from the source. When router i receives multicast Interest $MI[g(j), D^I(p), mc^I(p)]$ from a neighbor router or a local content consumer p, it forwards the Interest only if $mc = 1 + v$, where v is the current mc value stored in $MART^i$ for the multicast group. Router i updates the mc value in $MART^i$ as it forwards the Interest, and subsequent Interests with the same mc value of $1 + v$ are simply dropped. As a result, each router in an MFT forwards a single copy of any Interest asking for the next multicast content object towards the source. This is like aggregating Interests for a multicast group over the MFT of the group, but with no need to store per-Interest forwarding state.

5.2 Performance Comparison

We compare the forwarding entries needed to forward Interests and responses in NDN and CCN-GRAM, as well as the end-to-end delays incurred, using simulation experiments based on implementations of NDN and CCN-GRAM in the ndnSIM simulation tool [1]. The NDN implementation was used without modifications, and CCN-GRAM was implemented in the ndnSIM tool following Algorithms 1 to 4.

The network topology consists of 150 routers distributed uniformly in a 100m $\times$

100m area and routers with distance of 15m or less are connected with point-to-point links of delay 15ms. The data rates of the links are set to 1Gbps to eliminate the effects that a sub-optimal implementation of CCN-GRAM or NDN may have on the results. Only 10 routers chosen randomly are connected to local content producers of multiple name prefixes, 50 other routers are connected to local content consumers, and all routers act as relays. This choice was made to illustrate the existence of a "network edge" and the fact that only a relatively small number of sites host content producers. Interests are generated with a Zipf distribution with parameter $\alpha = 0.7$ and producers are assumed to publish 1,000,000 different COs. Each cache can store up to 1000 objects, or 0.1% of the content published in the network. This caching capacity was selected to compare on-path caching with edge caching when Interests must be forwarded in the network, rather than being answered with locally cached content.

We considered *total Interest rates per router* of 50, 100, 500, and 2000 objects per second corresponding to the sum of Interests from the local consumers connected to a router. The increasing values of total request rates can be viewed as higher request rates from a constant user population of local active users per router, or an increasing population of active users per router. The Interest rates we assume are actually very low according to recent results addressing the size that PITs would have under realistic Internet settings [33, 42, 43, 41].

We considered on-path caching and edge caching. For the case of on-path caching, every router on the path traversed by a data packet from the producer to the consumer caches the CO in its local cache. On the other hand, with edge caching, only the router directly connected to the requesting consumer caches the resulting CO. All caches are LRU.

5.2.1 Size of Forwarding Tables

Figure 5.3 shows the average size and standard deviation of the sizes of PITs, ARTs and LIGHTs on a logarithmic scale as functions of Interest rates. The size of LIGHTs corresponds only to the number of local Interests pending responses. The number of entries corresponding to content cached locally can be up to 1000 for both NDN and CCN-GRAM.

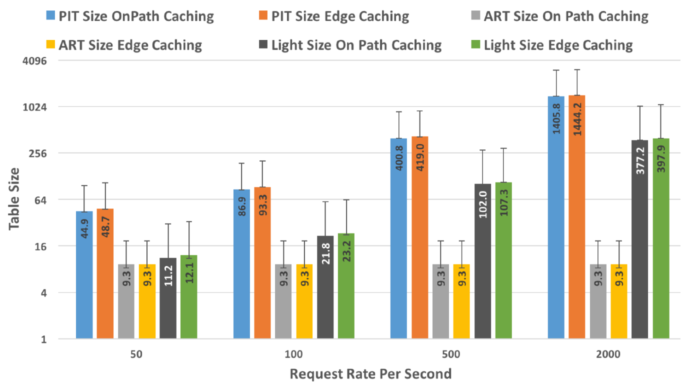


Figure 5.3: Average size of forwarding tables

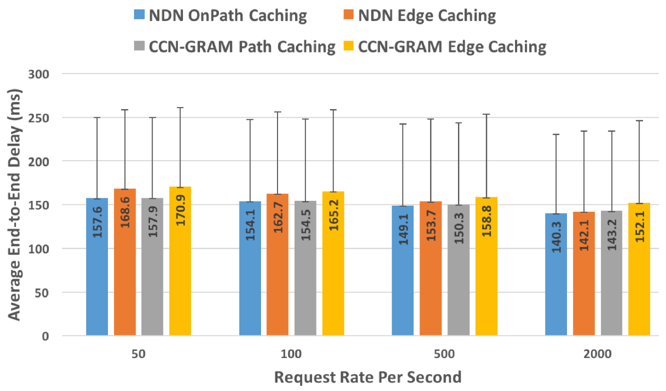


Figure 5.4: Average end-to-end delays

As the figure shows, the size of PITs grows dramatically as the rate of content requests increases, which is expected given that PITs maintain per-Interest forwarding state.

By contrast, the size of ARTs, which is the only forwarding state stored by relay routers, is only a small fraction of the total number of routers and remains fairly constant with respect to the content request rates, which is always one or multiple orders of magnitude smaller than the average PIT size. The size of LIGHTs is a function of the number of COs requested locally or cached on path, but the average size of a LIGHT is an order of magnitude smaller than the average size of a PIT. The size of a ART is independent of where content is being cached, given that an ART entry is stored independently of how many Interests traverse the route. Interestingly, edge-caching renders only slightly larger PIT sizes than on-path caching in NDN.

5.2.2 Average Delays

Figure 5.4 shows the average end-to-end delay for NDN and CCN-GRAM as a function of content request rates for on-path caching and edge caching. As the figure shows, the average delays for NDN and CCN-GRAM are comparable for all values of the content request rates. This should be expected, given that the static, loop-free routes in the FIBs prevent Interests to "wait to infinity" in PITs, the signaling overhead incurred by NDN and CCN-GRAM is similar, and in-network caching obviates the need for Interest aggregation.

5.3 Conclusions

We presented simulation results showing that Interest aggregation rarely occurs when in-network caching is used. Our analysis is limited; however, our detailed characterization of Interest aggregation via analytical modeling and simulation analysis renders the same conclusion.

We introduced CCN-GRAM to eliminate the performance limitations associated with PITs. CCN-GRAM is the first approach to Interest-based content-centric networking that supports the forwarding of Interests and responses to them using datagrams that do not reveal the identity of their origins to forwarding routers, caching sites, or content providers.

Simulation experiments were used to show that end-to-end delays incurred in CCN-GRAM and NDN are similar when either edge caching or on-path caching is used, but the storage requirements for CCN-GRAM are orders of magnitude smaller than for NDN. The results for CCN-GRAM indicate that it could be deployed with only routers at the edge maintaining LIGHTs and caches. Additional work is needed to make the forwarding of Interests in CCN-GRAM as efficient as the forwarding of responses to Interests using ARTs. The goal is to enable Interest forwarding at Internet scale that does not require routers to look up FIBs with billions of name-prefix entries as is the case in NDN and CCNx.

Both ARTs and PITs must be updated when the paths traversed by Interests and their responses must change due to congestion, topology changes, or mobility of consumers and providers. Yi et al [24] argue that per-Interest forwarding state enables faster response to topology changes and congestion, because local repair mechanisms can be used. However, multipath routing, and dynamic load balancing schemes based on datagram forwarding have been shown to attain results very close to optimal routing [21] and can be easily applied to CCN-GRAM in the future.

CCN-GRAM can use the same content security features adopted in CCNx and

NDN to limit or eliminate cache poisoning attacks, because it makes no modifications to the way in which content is protected in data packets or how a name can be securely linked to the payload of a CO. However, CCN-GRAM enjoys an enormous advantage over CCNx and NDN in that it eliminates the ability for malicious users to mount Interest-flooding attacks aimed at overwhelming the forwarding tables of routers [78, 43]. An ART entry can be added only for valid local identifiers at each router and for routes that satisfy the ordering constraint imposed with LFR. Given that both conditions are managed in the control plane, mounting attacks on ARTs is much more difficult than simply having users send Interests for COs corresponding to valid name prefixes.

Algorithm 14 Processing Interest from router p at router i

```
function Interest_Forwarding
INPUT: $LIGHT^i$, $LIST^i$, $FIB^i$, $ART^i$, $I[n(j), AID^I(p), D^I(p)]$;
$AID^R(i) = AID^I(p)$;
if $n(j) \in LIGHT^i$ then
    if $p[n(j)] \neq nil$ then
        retrieve CO $n(j)$; send $DP[n(j), AID^R(i), sp(j)]$ to $p$
    end if
else
    if $n(j)^* \in LIGHT^i$ then
        send $REP[n(j), AID^R(i), \textsf{no content}]$ to $p$    (% $n(j)$ does not exist)
    else
        if $n(j)^* \notin FIB^i$ then
            send $REP[n(j), AID^R(i), \textsf{no route}]$ to $p$  (% No route to $n(j)^*$ exists)
        else
            for each $s \in N^i$ by rank in $FIB^i$ do
                if $D^I(p) > D(i, n(j)^*, s)$  (% LFR is satisfied)  then
                    $SET = \emptyset$; $AID^I(i) = nil$; $collision = 0$;
                    for each entry $ART^i[AID, s, map]$ do
                        $SET = SET \cup \{AID\}$;
                        if $AID(ART^i) = AID^I(p)$ then
                            if $s(ART^i) = p$ then
                                $AID^I(i) = f_i(s)[AID(ART^i)]$
                            else
                                $collision = 1$
                            end if
                        end if
                        if $map(ART^i) = AID^I(p) \wedge s(ART^i) = p$ then
                            $AID^I(i) = f_i(s)[AID(ART^i)]$
                        end if
                    end for
                    if $collision = 0 \wedge AID^I(i) = nil$ then
                        create entry $ART^i[AID^I(p), p, AID^I(p)]$;
                        $AID^I(i) = f_i(s)[AID^I(p)]$
                    end if
                    if $collision = 1 \wedge AID^I(i) = nil$ then
                        select $a \in LI^i(i) - SET$;
                        create entry $ART^i[a, p, AID^I(p)]$; $AID^I(i) = f_i(v)[a]$
                    end if
                    $D^I(i) = D(i, n(j)^*, s)$;
                    send $I[n(j), AID^I(i), D^I(i)]$ to $s$; return
                end if
            end for (% LFR is not satisfied; Interest may be traversing a loop)
            send $REP[n(j), AID^R(i), \textsf{loop}]$ to $p$
        end if
    end if
end if
```

Algorithm 15 Processing data packet from router s at router i

```
function Data Packet
INPUT: $LIGHT^i$, $LIST^i$, $ART^i$, $DP[n(j), AID^R(s), sp(j)]$;
[o] verify $sp(j)$;
[o] if verification with $sp(j)$ fails then discard $DP[n(j), AID^R(s), sp(j)]$;
$a = f_i^{-1}(s)[AID^R(s)]$; retrieve entry $ART^i[a, p, m]$;
if $ART^i[a, p, m]$ does not exist then drop $DP[n(j), AID^R(s), sp(j)]$;
if  $p = i$  (% router $i$ was the origin of the Interest)  then
    for each $c \in lc[n(j)]$ do
        send $DP[n(j), c, sp(j)]$ to $c$; $lc[n(j)] = lc[n(j)] - \{c\}$
    end for
else
    if $p \in N^i$ then
        $AID^R(i) = m$; send $DP[n(j), AID^R(i), sp(j)]$ to $p$
    end if
end if
if no entry for $n(j)$ exists in $LIGHT^i$ then
   create $LIGHT^i$ entry for $n(j)$: $lc[n(j)] = \emptyset$
end if
store CO in local storage; $p[n(j)]$ = address of CO in local storage
```

Algorithm 16 Process reply from router s at router i

```
function REPLY
INPUT: $LIGHT^i$, $LIST^i$, $ART^i$, $REP[n(j), AID^R(s), \mathsf{CODE}]$;
$a = f_i^{-1}(s)[AID^R(s)]$; retrieve entry $ART^i[a, p, m]$;
if $ART^i[a, p, m]$ does not exist then drop $REP[n(j), AID^R(s), \mathsf{CODE}]$;
if  $p = i$  (% router $i$ was the origin of the Interest)  then
    for each $c \in lc[n(j)]$ do
        send $REP[n(j), c, \mathsf{CODE}]$ to $c$
    end for
    delete entry for $n(j)$ in $LIGHT^i$
else
    if $p \in N^i$ then
        $AID^R(i) = m$; send $REP[n(j), AID^R(i), \mathsf{CODE}]$ to $p$
    end if
end if
```

Chapter 6

Efficient Multicasting in Content-Centric Networks Using Datagrams

6.1 Multicasting in CCN-GRAM

6.2 Preliminaries

CCN-GRAM assumes that Interests are retransmitted only by the consumers that originated them, and that routers use exact Interest matching. A router that advertises being an origin of a name prefix stores all the content objects associated with that prefix at a local content store. Routers know which interfaces are neighbor routers and which are local users within a finite time, and forward Interests on a best-effort basis. For convenience, it is assumed that a request for content from a local user is sent to its local router in the form of an Interest.

The name of content object (CO) j is denoted by $n(j)$ and the name prefix that is the best match for name $n(j)$ is denoted by $n(j)^*$. Similarly, the name of a multicast group j is denoted by $g(j)$ and the name prefix that is the best match for name $g(j)$ is denoted by $g(j)^*$. The set of neighbors of router i is denoted by N^i.

6.2.1 Information Exchanged and Stored

Like NDN and CCNx, CCN-GRAM uses Interests, data packets, and replies to Interests. Routers differentiate between unicast and multicast Interests. A unicast Interest requests a content object (CO) by name, while a multicast Interest requests content from a multicast group by name.

A multicast Interest $MI[g(j), D^I(i), mc^I(i)]$ sent by router i to router n states: the name of the multicast group $g(j)$ from which content is requested, the distance from router i to the source of the multicast group ($D^I(i)$), and a multicast counter ($mc^I(i)$) used for pacing.

A multicast data packet $MP[g(j), sp(j), mc^R(i)]$ states the name of the multicast group $g(j)$ from which content is being sent, a security payload ($sp(j)$) used optionally to validate the CO, a multicast counter ($mc^R(i)$), plus the content object (CO) corresponding to the value of the multicast counter.

A reply sent by router i in response to a multicast Interest is denoted by $MR[g(j), CODE, mc^R(i)]$ and contains the name of the multicast group for which the Interest was sent, a code (CODE) indicating the reason why the reply is sent, and the current value of the multicast counter stored at the router sending the repoy. Possible reasons for sending a reply include: an Interest loop is detected, or no route is found towards requested content.

Router i uses four tables to forward multicast traffic: a forwarding information base (FIB^i), a multicast anonymous routing table ($MART^i$), and a group membership table (GMT^i).

FIB^i is indexed using known content name prefixes. The entry for name prefix $g(j)^*$ states the distance reported by each next-hop neighbor router for the prefix. The distance stored for neighbor q for name prefix $g(j)^*$ in FIB^i is denoted by $D(i, g(j)^*, q)$. Each entry in FIB^i is stored for a maximum time determined by the lifetime of the corresponding entry in the routing table of the router.

$MART^i$ maintains forwarding state to the receivers of multicast groups. Each entry of the MART specifies a multicast group name, the current value of the multicast counter (mc) for the group, and a list NH of next hops to the group of receivers who have sent multicast Interests for the group. $MART^i[g(j), mc, NH]$ denotes the entry for group $g(j)$ in $MART^i$. GMT^i lists the mappings of multicast group names to the lists of local receivers that requested to join the groups. If in-network caching is used as part of multicsting, the entry for group $g(j)$ also states a pointer $p[g(j)]$ to the content that has been cached for the group listing each CO by the value of the multicast counter used to retrieve COs for $g(j)$.

6.2.2 Multicast Content Dissemination

Multicast content dissemination is based on the forwarding of Interests along multicast forwarding trees (MFT) to the sources of multicast groups, followed by the forwarding of multicast data packets on the reverse paths traversed by Interests. Forwarding multicast Interests is based on the information stored in the FIBs maintained by routers. In contrast to NDN, CCN-GRAM maintains forwarding state for Interests on a per-group basis rather than on a per-Interest basis.

Algorithms 17 to 19 outline the steps taken by routers to process and forward multicast Interests, and return multicast data packets or replies for the case of real-time multicasting.

To compare multicasting in CCN-GRAM directly with NDN, we assume pull-based dissemination of real-time multicast content, such that a single CO is sent to multicast receivers in response to an Interest sent to the source of a multicast group over the MFT.

We assume that each router is initialized properly, knows the identifiers used to denote local consumers, and knows all its neighbors. We assume that a routing protocol (e.g., DCR [10, 111]) operating in the control plane updates the entries of routing tables listing one or multiple next hops towards name prefixes. Routers populate their FIBs with routes to name prefixes based on the data stored in their routing tables.

The value of the multicast counter for group $g(j)$ in $MART^i$ is denoted by $mc^i[g(j)]$, and the set of next-hop routers listed in $MART^i$ for receivers in $g(j)$ is denoted by $NH^i[g(j)]$. The local receivers for group $g(j)$ listed in GMT^i is denoted by $GMT^i[g(j)]$.

The forwarding of multicast Interests towards the sources of multicast groups is assumed to rely on the selection of next hops towards name prefixes listed in FIBs that provide the best matches to the multicast group names stated in the Interests. We assume that routers with local consumers maintain caches of multicast content. The first content object (CO) of a multicast group is labeled by the name of the group and a multicast counter equal to one, and an empty entry for a multicast group is initialized with a multicast-counter value equal to zero. We assume that all initial requests to join a group state a multicast counter equal to one, and that forwarding state for a group stored in the MART of a router is deleted after a timeout if no Interests are received for the group.

For simplicity, we do not include the steps taken by routers to respond to the failures or additions of interfaces with neighbor routers or local consumers. Furthermore, we assume that Interests and responses to them are transmitted reliably between any two neighboring routers. In essence, forwarding state related to a failed interface must be deleted and the corresponding replies with negative acknowledgments must be sent to previous next hops to remote receivers or local receivers as needed. Forwarding state associated with new interfaces is instantiated as a result of new Interests being forwarded.

Algorithm 17 shows the steps taken by router i to process Interests received from local consumers. For convenience, multicast content requests from local consumers are assumed to be Interests stating the name of a group, the name of the consumer, and an empty distance to the content assumed to denote infinite. The same format of data packets and replies used among routers is used to denote the responses a router sends to local consumers. Consumers increase the values of their multicast counters by one to request the next pieces of multicast content from multicast sources.

Router i adds consumer c as a local receiver in group $g(j)$ by adding an entry for $g(j)$ in GMT^i with c as a local receiver for the group, and indicates that it has local receivers in $MART^i$ by adding itself as a next hop towards receivers of the group. Router i forwards a single copy of a multicast Interest requesting more content from a multicast source independently of how many local receivers or neighbor routers send multicast Interests to router i. This is done by means of the multicast counter (mc) maintained by each router and multicast receiver, and the multicast-counter field included in Interests and responses to them.

A content consumer c asks to join a multicast group $g(j)$ as a receiver by sending an Interest $MI[g(j),\ D^I(c) = nil, mc^I(c) = 1]$. If the value of the multicast counter for the group stored by the router is larger, the router responds with the latest multicast data packet corresponding to the current value of the multicast counter maintained by the routers for the multicast group. A router sends a negative acknowledgment to an Interest from a local consumer with a multicast-content value different than the next expected value to force a retransmission and keep all local consumers in the same multicast group using the same current value of the multicast counter, while reducing end-to-end latencies incurred in delivering multicast content to consumers far away from group sources. A consumer requests more content from a multicast group by sending a multicast Interest after incrementing the value of the multicast counter for the group.

A router forwards Interest $MI[g(j), D^I(i), mc^I(i)]$ towards the source of multicast group $g(j)$ based on the information in its FIB.

Algorithm 18 shows the steps taken by router i to process an Interest received from a neighbor router p. Router i follows similar steps to those in Algorithm 1 to respond to an

Interest with a multicast data packet to the neighbor router if the content is local and the multicast counter in the Interest is smaller than the current value of the multicast counter at the router. If the Interest requests the next CO from the group and the group source is local, the multicast data packet is sent to all next hops along the MFT. Alternatively, if the multicast source is remote, the router forwards the Interest ensuring that no forwarding loops occur. Let $S^i_{g(j)^*}$ denote the set of next-hop neighbors of router i for prefix $g(j)^*$. The following rule is used to ensure that multicast Interests cannot traverse routing loops, even if the routing data stored in FIBs regarding name prefixes is inconsistent and leads to routing-table loops.

Loop-Free Forwarding Rule (LFR):

Router i accepts $MI[n(j), D^I(k), mc^I(k)]$ from router k if:

$$\exists\ v \in S^i_{g(j)^*}(\ D^I(k) > D(i, g(j)^*, v)\ ) \tag{6.1}$$

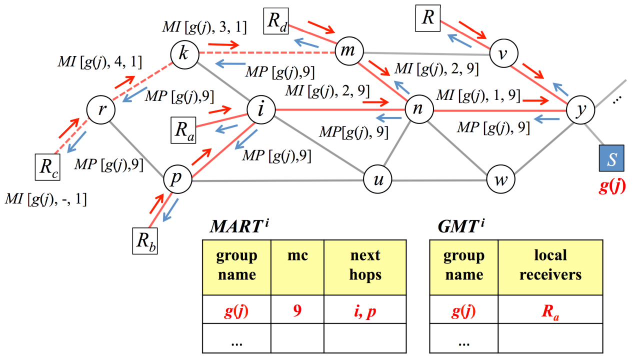
Router i tries to forward the Interest to a next hop s for the best prefix match for $n(j)$ that satisfies LFR. The highest-ranked router satisfying LFR is selected as the successor for the Interest and router i. If no neighbor is found that satisfies LFR, a reply is sent stating that a loop was found.

LFR is based on the same approach proposed previously to eliminate Interest looping in NDN and CCNx [36, 37]. It ensures loop-free forwarding of Interests by ensuring that routers forward Interests only to next hops that are closer to the intended name prefix.

Algorithm 19 outlines the processing of multicast data packets. If local consumers requested the content in the data packet, it is sent to those consumers based on the information stored in GMT^i. If the router has neighbor routers that are next hops towards remote receivers of the multicast group, router i forwards the data packet to all neighbors listed for $g(j)$ in $NH^i[g(j)]$ other than router i itself if there are local receivers. Routers take similar steps in the forwarding of replies to multicast Interests when retransmissions are done by consumers, i.e., routers simply forward replies back to the consumers along the MFT created by the forwarding of multicast Interests.

6.3 Example of Multicast Dissemination in CCN-GRAM

Figure 6.1 illustrates the forwarding of multicast Interests and multicast data packets in CCN-GRAM. As the figure shows, router i maintains a forwarding table ($MART^i$) specifying the next hops to multicast receivers for each multicast-group name, and a table (GMT^i) listing the local receivers for each multicast-group name. As the figure shows, the entry for group $g(j)$ in $MART^i$ lists router i as a next hop, which indicates the presence of local receivers; the one local receiver (R_a) for group $g(j)$ is listed in GMT^i.



group name	mc	next hops
g(j)	9	i, p
...		

group name	local receivers
g(j)	R_a
...	

Figure 6.1: Native multicast support in CCN-GRAM

The entries in the MARTs and GMTs maintained by routers define the forwarding multicast trees (FMT) of all multicast groups created in the network, and are established by the forwarding of Interests, just as in NDN or CCNx. However, the use of multicast counters eliminates the need to maintain per-Interest forwarding state. In the figure, dashed lines represent links along the path from consumer R_c joining the multicast group after the source has disseminated CO with $mc = 9$ to the rest of the MFT. The late joiner is brought up to the current state of the multicast group by the multicast counter carried in each data packet.

6.4 Performance Comparison

We compare the average table sizes and end-to-end delays for multicast traffic in CCN-GRAM and NDN by running experiments based on implementations of CCN-GRAM and NDN in the ndnSIM simulation tool [1]. The implementation of CCN-GRAM is based on the algorithms presented, and NDN implementation from ndnSIM is used without modification. The simulation scenario includes 200 nodes distributed uniformly in a $100m \times 100m$ area. Nodes with a distance of 15 meters or less from each other are connected with a point-to-point link of 15ms delay. Data rates are set to 1Gbps to eliminate or reduce the impact of an inefficient implementation of either NDN and CCN-GRAM on the results. Using on-path caching strategy, each router in these experiments can cache up to 1000 content objects. In each simulation scenario, there are multiple multicast groups, and each group contains multiple consumers and one producer. Consumer and producer nodes for each group is selected at random from the 200 routing nodes in the network.

We compared forwarding table sizes in NDN and CCN-GRAM by four different varying parameters: Multicast groups count, multicast group size, Interest request rate, and link delay. We also compared average end-to-end delay in NDN and CCN-GRAM for different request rates. For this purpose we consider minimum download rate of 1.5 Mbps for audio/radio streaming, 5 Mbps for HD video streaming, and 25Mbps for Ultra HD video streaming. Considering the standard packet size of 4KB advocated in NDN, we compared different scenarios with constant rate of 50 to 800 interests per second from each consumer application.

6.4.1 Size of Forwarding Tables

Figure 6.2 shows the results of a simulation experiment that includes 20 multicast groups, each with 20 consumers and one producer.

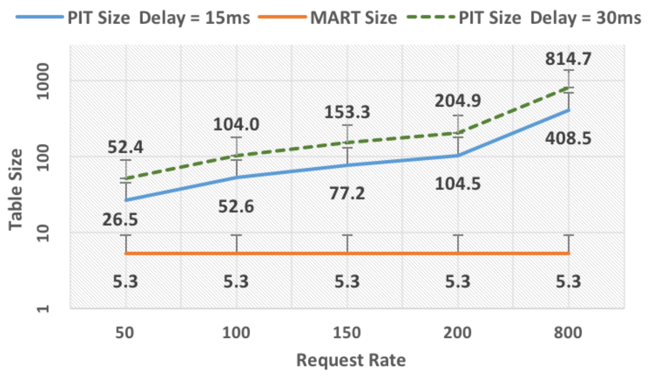


Figure 6.2: Average size of forwarding tables for varying request rates

The above figure shows the average size of a forwarding table in logarithmic scale as a function of Interest (request) rates. Given that CCN-GRAM adds a single entry per multicast group, the number of entries in a MART is independent of request rate. By contrast, the size of PITs in NDN is a function of the rate at which Interests arrive at routers. The maximum MART size for this scenario is 20, and the average size of a MART table is 5.26 independently of the request rates. As the figure shows, the number of PIT entries is highly affected by the Interest rates from consumers. For the case of a 15ms link delay, increasing the Interest rate to 800 results in average PIT size of 408 and tables as large as 1300 entries for routers.

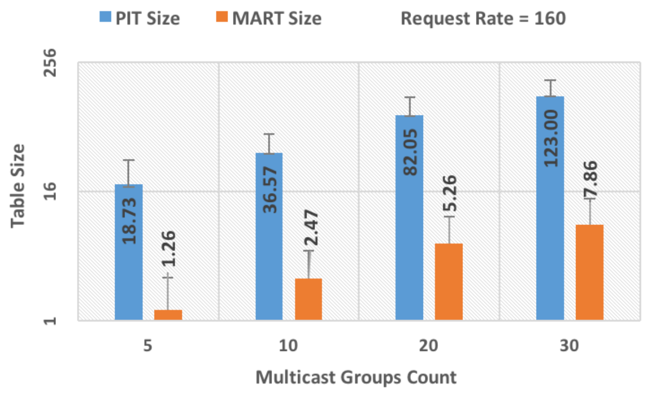


Figure 6.3: Average size of forwarding tables vs. number of multicast groups

Figure 6.3 shows the average MART size versus the average PIT size for varying number of multicast groups from 5 to 30 groups, with each group having 20 consumers with Interest (request) rate of 160 Interests per second, which is enough to support HD video streaming with each data packet being 4KB. In CCN-GRAM, the number of entries of MART tables cannot exceed the total number of multicast groups. On the other hand, as the figure shows, the number of entries in the PIT of a router is directly related to the number of interests received by the router, which in turn depends on the number multicast groups and the request rate per group. Accordingly, the average PIT size can grow dramatically.

Figure 6.4 shows the average size of PITs and MARTs for varying multicast group sizes from 10 to 40 consumers per group. As the size of a multicast group increases, more routers become involved in forwarding multicast Interests and multicast data packets in both NDN and CCN-GRAM. This results in larger average sizes of both PITs and MARTs. However, the grow rate for NDN is higher because of entries are added to PITs on a per Interest basis.

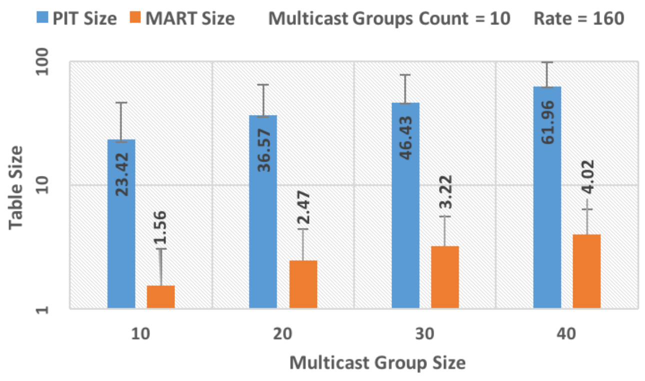


Figure 6.4: Average size of forwarding tables for varying size of multicast groups

6.4.2 Average Delays

As Figure 7.8 shows, the average delay for CCN-GRAM is shorter than the delays incurred in NDN. According to Algorithm 3, the first multicast Interest received by the producer, results in multicast of data toward current members of multicast group in CCN-GRAM, even if the Interest from a member or previous hop relay in the MFT has not been received yet. On the other hand, in NDN, if one consumer node is far from the producer compared to other consumer nodes such that its interests is not aggregated with the same interests from other consumers, request of that node for a multicast data will be processed separate from other group members, which results in lower throughput and higher delays. The operation of NDN could be modified to mimic the way in which CCN-GRAM forwards multicast data over MFTs, in which case end-to-end latencies would be similar.

The results shown in Figures 6.3, 6.4, and 7.8 are for link delays of 15ms, which results in end-to-end delays of around 110 ms. However if higher link delays occur, the average number of PIT entries can grow to much larger values, because the number of pending interests in each router increases. As shown in Figure 6.2, link delays of 30ms

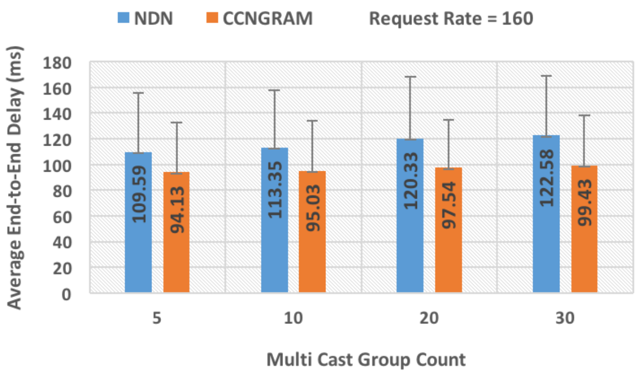


Figure 6.5: Average end-to-end delay for varying number of multicast groups

results in 240 end-to-end delays and much larger PITs. For the case of 800 Interests per second, the average PIT size in NDN increases to 814 entries some relaying routers storing as many as as 2700 PIT entries. On the other hand, the average size of MARTs is not affected by varying link delays and remains constant, which makes CCN-GRAM more scalable and reliable, and more attractive for real-time streaming applications requiring high throuhput.

6.5 Conclusions

We presented CCN-GRAM, the first approach for multicasting in content-centric networks that eliminates the need to maintain per-Interest forwarding state and still operates based on the forwarding of Interests in the data plane, without the need for a multicast routing protocol in the control plane.

Simulation experiments were used to show that the storage requirements for CCN-GRAM are orders of magnitude smaller than for NDN, and that end-to-end delays in CCN-GRAM are similar if not smaller than in NDN, especially when high data rates for multicast streaming is needed.

Additional work is needed to define efficient mechanisms to react to resource failures with minimum disruption to MFTs, reliable multicasting support, and new flow control approaches that allow more than one CO to be delivered per multicast Interest. In addition, CCN-GRAM could be applied when sources disseminate content without the need for receivers to submit Interests after joining a group [109].

Algorithm 17 Processing Interest from user c at router i

```
function Interest_Source
INPUT: $GMT^i$, $FIB^i$, $MART^i$, $MI[g(j), D^I(c) = nil, mc^I(c)]$
if $g(j)^* \in FIB^i$  (% Route to $g(j)$ exists) then
    if $MART^i$ entry for $g(j)$ does not exist then
        $mc^i[g(j)] = 0$; $NH^i[g(j)] = \emptyset$;
        create entry $MART^i[g(j), mc, NH]$; $GMT^i[g(j)] = \emptyset$;
    end if
    $GMT^i[g(j)] = GMT^i[g(j)] \cup c$; $NH^i[g(j)] = NH^i[g(j)] \cup i$;
    if $mc^I(c) \neq mc^i[g(j)] + 1$ then
        if $p[g(j)] \neq nil$ then
            retrieve CO for $mc^i[g(j)]$; $mc^R(i) = mc^i[g(j)]$;
            send $MP[g(j), sp(j), mc^R(i)]$ to $c$
        else
            $mc^R(i) = mc^i[g(j)]$; send $MR[g(j), \text{Interest error}, mc^R(i)]$ to $c$
        end if
    else
        if $i$ is the source for $g(j)$ then
            $mc^R(i) = mc^i[g(j)]$;
            send $MP[g(j), sp(j), mc^R(i)]$
            to receivers in $GMT^i[g(j)]$ and next hops in $NH^i[g(j)]$
        else
            $mc^I(i) = mc^i[g(j)]$;
            $D^I(i) = Min\{D(i, g(j)^*, u)$ for $u \in S^i_{g(j)^*}\}$;
            for each $v \in N^i$ by rank in $FIB^i$ do
                if $D(i, g(j)^*, v) = D^I(i)$ then
                    send $MI[g(j), D^I(i), mc^I(i)]$ to $v$; return
                end if
            end for
        end if
    end if
else
    $mc^R(i) = mc^i[g(j)]$; send $MR[g(j), \text{no route}, mc^R(i)]$ to $c$
end if
```

Algorithm 18 Processing multicast Interest from router p at router i

```
function Interest_Forwarding
INPUT: $GMT^i$, $FIB^i$, $MART^i$, $MI[g(j), D^I(p), mc^I(p)]$;
if $g(j)^* \in FIB^i$  (% Route to $g(j)$ exists) then
    if $MART^i$ entry for $g(j)$ does not exist then
        $mc^i[g(j)] = 0$; $NH^i[g(j)] = \emptyset$;
        create entry $MART^i[g(j), mc, NH]$; $GMT^i[g(j)] = \emptyset$
    end if
    $NH^i[g(j)] = NH^i[g(j)] \cup p$;
    if $mc^I(p) \neq mc^i[g(j)] + 1$ then
        $mc^R(i) = mc^i[g(j)]$;
        if $p[g(j)] \neq nil$ then
            retrieve CO for $mc^i[g(j)]$; send $MP[g(j), sp(j), mc^R(i)]$ to $p$
        else
            send $MR[g(j), \textsf{Interest error}, mc^R(i)]$ to $p$
        end if
    else
        $mc^i[g(j)] = mc^i[g(j)] + 1$;
        if $i$ is the source for $g(j)$ then
            retrieve CO for $mc^i[g(j)]$; $mc^R(i) = mc^i[g(j)]$;
            send $MP[g(j), sp(j), mc^R(i)]$ to
            receivers in $GMT^i[g(j)]$ and next hops in $NH^i[g(j)]$
        else
            $D^I(i) = Min\{D(i, g(j)^*, u)$ for $u \in S^i_{g(j)^*}\}$;
            for each $v \in N^i$ by rank in $FIB^i$ do
                if $D^I(p) > D^I(i)$  (% LFR is satisfied) then
                    $mc^I(i) = mc^i[g(j)]$; send $MI[g(j), D^I(i), mc^I(i)]$ to $v$;
                    return
                end if
            end for
            $mc^R(i) = mc^i[g(j)]$; send $MR[g(j), \textsf{loop}, mc^R(i)]$ to $p$
        end if
    end if
else
    $mc^R(i) = mc^i[g(j)]$; send $MR[g(j), \textsf{no route}, mc^R(i)]$ to $p$
end if
```

Algorithm 19 Processing multicast data packet from router s at router i

```
function Multicast Data Packet
INPUT: $GMT^i$, $MART^i$, $MP[g(j), sp(j), mc^R(s)]$;
[o] verify $sp(j)$;
[o] if verification with $sp(j)$ fails then discard $MP[g(j), sp(j), mc^R(s)]$;
if $NH^i[g(j)] \neq \emptyset$ then
    $mc^R(i) = mc^R(s)$; if $mc^i[g(j)] < mc^R(s)$ then $mc^i[g(j)] = mc^R(s)$;
    if $GMT^i[g(j)] \neq \emptyset$ (% router $i$ has local receivers in group $g(j)$) then
        for each $c \in GMT^i[g(j)$ do
            send $MP[g(j), sp(j), mc^R(i)]$ to $c$
        end for
    end if
    if $NH^i[g(j)] - \{i\} \neq \emptyset$ then
        for each $h \in NH^i[g(j)] - \{i\}$ do
            send $MP[g(j), sp(j), mc^R(i)]$ to $h$
        end for
    end if
    [o] store CO in local storage at $p[g(j)]$ indexed with $mc^R(i)$
else
    drop $MP[g(j), sp(j), mc^R(s)]$
end if
```

Chapter 7

Content-Centric Networking at Internet Scale through The Integration of Name Resolution and Routing

Several Information-Centric Networking (ICN) architectures have been proposed [116, 118, 137] to improve on the performance of the IP Internet by enabling packet forwarding based on the names of content or services required, rather than the addresses where they may be hosted. They attempt to accomplish this by means of new ways to integrate name resolution (mapping of names to locations) and routing (establishing paths between locations) functions.

Most architectures keep name resolution and routing independent of each other, but offer major improvements over the current use of the Domain Name System (DNS) to map domain names to IP addresses. By contrast, the Content-Centric Networking (CCNx) [4] and the Named Data Networking (NDN) [17, 27] architectures *merge* name resolution with routing to content and services in order to allow consumers to request content objects (CO) or services by name. Routers provide this service using three tables. A content store (CS) lists the COs that are cached locally. A Pending Interest Table (PIT) keeps forwarding state for each Interest (a request for a CO) processed by a router, such that a single copy of an Interest for the same CO is forwarded and responses to Interests can be sent over the reverse paths traversed by the Interests. A forwarding information base (FIB) listing the next hops to name prefixes is used to forward Interests towards content producers.

The simplicity of merging name resolution with routing is very attractive. However, it comes at a big price, in terms of storage requirements associated with FIBs and PITs [33, 132, 41, 42, 43, 113] and additional vulnerabilities to DDoS attacks associated with PITs [78, 113].

Consider a network in which a name-based routing protocol establishes routes to all known name prefixes and to those routers that announce name prefixes being locally available, which we call anchors of the prefixes. Section 7.1 shows that the path traversed by an Interest when routers maintain FIBs with entries for name prefixes is the same as the path traversed by the Interest if the first router binds the name prefix that provides the best match for the CO name to the name of an anchor of that prefix, and routers forward the Interest towards that anchor.

Section 7.1.1 describes CCN-RAMP, which is based on two key innovations. First, CCN-RAMP integrates name resolution with routing to name-prefix anchors using the name-based content routing protocol running in the control plane (e.g., [10, 124, 125]). The approach takes advantage of the result presented in Section 7.1, and the consequence is that routers can forward Interests using forwarding tables that are orders of magnitude smaller than the FIBs required in NDN and CCNx, and even smaller than the FIBs needed in the IP Internet. Second, CCN-RAMP extends recent results in [108, 109] to eliminate forwarding loops, and to replace PITs with small forwarding tables listing the next hops towards the origins of Interests, without identifying such origins.

Section 7.2 compares the performance of CCN-RAMP with NDN when either no caching or on-path caching is used in a 153-router network. CCN-RAMP incurs very similar Interest traffic than NDN to retrieve content, requires orders of magnitude fewer entries in forwarding tables than NDN, and needs fewer table look ups to retrieve any given CO than NDN.

A major impediment for the IP Internet to provide efficient access to content and services by name is the poor interaction between name resolution and routing that it currently supports. A client must be an integral part of name resolution and is required to bind the name of the CO or service to be requested to the location where the content or service is offered. A client first interacts with a local Domain Name System (DNS) server to obtain the mapping of a domain name to an IP address in order for a request for content or service can be sent to a specific IP address. Adding to this, the DNS is based on a hierarchical, static caching structure of servers hosting the mappings of domain names to addresses built and maintained independently of routing and requiring servers to be configured on how to contact other servers over the Internet.

7.0.1 Name Resolution and Routing

TRIAD [122] was one of the first projects to advocate using names for routing rather than addresses. Since then, many ICN architectures have been proposed that support name resolution and content routing functionality to enable consumers to ask for content objects (CO) or services by name. These architectures differ on how a CO name is mapped to a producer or source that can provide the requested CO or service, and the way in which paths are established for requests for COs or services and the associated requests.

Some ICN architectures [116, 118, 137], including DONA, PURSUIT, SAIL, COMET, and MobilityFirst, implement name resolution and routing as independent functions. In these architectures, name resolution servers (called by different names) are organized hierarchically, as multi-level DHTs, or along trees spanning the network [118], and consumers and producers contact such servers to publish and subscribe to content in various ways. Consumers obtain the locations of publishers from name resolution servers, and send their content requests to those locations to get the required content or services, and address-based routing is used to establish paths between consumers and subscribers or between resolution servers and subscribers or consumers.

A major limitation of keeping name resolution independent of routing stems from the complexity incurred in keeping name-resolution servers consistent with one another, and allowing consumers and producers to interact with the name-resolution system. Enabling the updates of name-to-address mapping is a non-trivial problem using hierarchical structures, spanning trees, or DHT-based organizations of servers. Another design consideration in these architectures is that a solution is still needed to preserve the anonymity of the sources of requests for COs or services, which may induce additional complexity in the forwarding plane or require the use of the forwarding mechanisms used in NDN and CCNx.

In contrast to most prior ICN architectures, NDN and CCNx *merge* name resolution and routing, such that routers are the facto name resolvers by establishing routes to name prefixes on a hop-by-hop basis. A major advantage of doing this is that it eliminates the complexity of designing and maintaining a network of name-resolution servers that replace the DNS. This merging of functionalities is supported by: (a) a name-based routing protocol operating in the control plane, which updates the entries in FIBs listing the next hops to known name prefixes, and (b) forwarding of Interests based on the longest prefix match (LPM) between the CO name in the Interest and a name prefix listed in the FIBs.

However, as attractive as the simplicity of merging of name resolution with routing in NDN and CCNx is, it comes at a very big price. Because the name of a CO or service is bound directly to a route on a hop-by-hop basis, each router along the path traversed by an Interest must look up a FIB listing the known name prefixes. To operate at Internet scale, FIB sizes in NDN are acknowledged to eventually reach billions of entries [132]. This is easy to imagine, given that the number of registered domains in the IP Internet was more than 300 million by the end of 2015. This is orders of magnitude larger than the largest FIB size for the IP Internet, which is smaller than 600,000 today.

To make matters worse, the name prefixes assumed in ICN architectures are variable length and much more complex than IP addresses. This means that efficient LPM algorithms developed for the IP Internet cannot be applied directly to NDN and CCNx. Indeed, it has been noted that NDN and CCNx cannot be deployed at Internet scale without further advances in technology [128].

7.0.2 Limitations of Using PITs

The size of a PIT grows linearly with the number of distinct Interests received by a router as consumers pipeline Interests (e.g., to support HD video streams) or request more content, or more consumers request content [33, 41, 42]. Unfortunately, as the following paragraphs summarize, PITs do not deliver substantial benefits compared to much simpler Interest-forwarding mechanisms, and can actually be counter-productive.

We have shown [109] that the percentage of aggregated Interests is minuscule when in-network caching is used, even Interests exhibit temporal correlation. We have also shown that per-Interest forwarding state is not needed to preserve the privacy of consumers issuing the Interests [108, 109].

Supporting multicast content delivery efficiently in the data plane has been viewed as a major reason to use PITs. However, as we demonstrate in [121], maintaining per-Interest forwarding state is unnecessary to implement pull-based multicast content dissemination. In a nutshell, a source-pacing algorithm can be used with routers maintaining per-source rather than per-Interest forwarding state. Routers forwarding traffic from a given multicast source maintain the most recent *multicast-counter value* (mv) for the source. A router forwards an Interest towards a source only if the mv stated in the Interest is larger than the value it currently stores for the source and discards the Interest otherwise. The net effect is the same as Interest aggregation, but without the large overhead of per-Interest forwarding state [121].

We have also shown [36, 37, 120] that Interest aggregation combined with the Interest-loop detection mechanisms used in NDN and CCNx can lead to Interests being aggregated while traversing forwarding loops without such loops being detected. This results in aggregated Interests "waiting to infinity" for responses that never come. In addition, using PITs makes routers vulnerable to Interest-flooding attacks [78, 43, 112, 113] in which malicious users can send malicious Interests aimed at making the size of PITs explode. Unfortunately, the countermeasures that have been proposed for these attacks [114] simply attempt to reduce the rates at which suspected routers can forward Interests, and this can be used to mount other types of denial-of-service attacks.

7.0.3 Limitations of Using FIBs Listing Name Prefixes

A number of proposals have been advanced to allow Interest forwarding in NDN based on LPM to keep up with new wire speeds while coping with the required FIB sizes and name-prefix structures. A big challenge for name-based Interest forwarding is to attain 100 Gbps rates or higher, given that FIBs listing name prefixes at Internet scale are much too large to fit into SRAM or Ternary Content Access Memory (TCAM) [128].

Several proposals for the implementation of FIBs for NDN rely on the use of tries and massive parallelism in order to avoid bottlenecks in the encoding process needed to use the tries [135, 136]. Other approaches are based on hash tables for FIB lookups [40, 131, 134, 138], which requires larger memory footprints and is not scalable to prefix names with a large number of name components. Hash-based approaches for name-based forwarding are based on DRAM technology and rely on massive parallel processing, because they would require hundreds of MiBs for just a few million prefixes.

Given the major limitations of using LPM in FIBs containing billions of name prefixes, a few proposals have been advanced to either reduce the size of FIBs listing name prefixes or eliminate the use of such FIBs.

Song et al. [132] introduced the concept of "speculative forwarding" based on longest-prefix classification (LPC) rather than LPM. LPC behaves just like LPM when a match is found in the FIB for the name stated in an Interest; however, with LPC a packet is forwarded to a next hop given by the FIB even if no match is found. Unfortunately, as described in [132], Interests may be forwarded along loops. Although the NDN forwarding strategy can prevent Interests from traversing the same forwarding loop multiple times, it cannot guarantee that Interests will not be aggregated while they traverse forwarding loops [37].

SNAMP [115] and PANINI [130] reduce FIB sizes by means of default routes and default-free zones. Edge routers resolve and keep track of local name prefixes, and forward all other interests toward backbone routers that create a default-free zone and map name prefixes to globally-routed names.

TagNet [127] uses content descriptors and host locators for forwarding. Content descriptors are variable-length sets represented with fixed-length Bloom filters for forwarding. Host locators are routing labels assigned to routers and hosts along one or multiple spanning trees. The labeling approach used in TagNet to assign locators is due to Thorup and Zwick [133]. TagNet results in FIBs that are much smaller than the FIBs required in NDN; however, it has a number of limitations. Content requests (Interests) must state the locators of their sources, which eliminates the anonymity provided in NDN and CCNx. Like any other scheme based on compact-routing, the paths traversed over the labeled spanning trees can have some stretch over the shortest paths. The Thorup-Zwick labeling used in TagNet is a depth-first search approach, and entire spanning trees may have to be relabeled after a single link failure. No prior work exists showing that TagNet or similar routing schemes based on interval-routing labels are suitable for large networks subject to topology changes or mobility of hosts and routers.

7.1 Routing to Name prefixes vs. Routing to Anchors

We use the term *anchor* to denote a router that, as part of the operation of the name-based routing protocol, announces the content corresponding to a name prefix being locally available. If multiple mirroring sites host the content corresponding to a name prefix, then the routers attached to those sites announce the same name prefix. However, a router simply caching COs from a name prefix does not announce the name prefix in the name-based routing protocol.

An anchor announcement can be done implicitly or explicitly as part of the operation of the name-based routing protocol running in the control plane, and routers simply caching COs are not anchors. For example, in NDN [125] and other name-based routing protocols based on link-state information [124], routers exchange link-state announcements (LSA) corresponding to either name prefixes or adjacencies to networks or routers. In this context, any router that originates an LSA for a name prefix is an anchor of the prefix. On the other hand, DCR [10] and other name-based content routing protocols (e.g., [111]) based on distance information to name prefixes use the names of anchors to ensure the correctness of the multi-path route computation [11].

Figure 7.1 shows an example of a content-centric network in which router y is an anchor for prefixes P^*, Q^*, and R^*; and router z is an anchor for prefixes A^*, C^*, and P^*. Dark solid arrowheads show the best next hop towards name prefixes. A red arrowhead indicates the best next hop to name prefixes with COs locally available at router y, and a dashed arrow head indicates the best next hop towards name prefixes with COs locally available at router z. The FIB entries at router p are shown when FIB entries are maintained for all instances of each name prefix, the nearest instances of a name prefix, or only for the anchors of name prefixes.

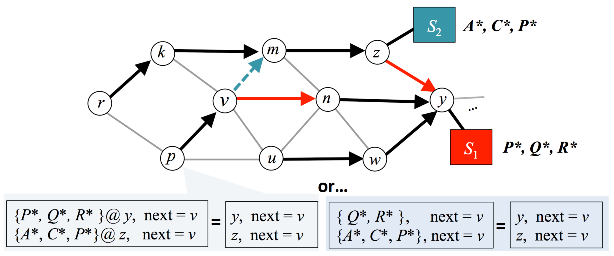


Figure 7.1: FIB entries for name prefix instances and FIB entries for anchors of name prefix instances

This example illustrates the fact that routes to instances of name prefixes must also be routes to the anchors announcing those instances. Therefore, the paths obtained from FIB entries listing name prefixes are the same as the paths obtained from FIB entries listing the anchors of name prefixes.

It is important to note that a router acts as the anchor of a name prefix over time scales that are many orders of magnitude larger than either the time scale at which congestion varies in a network due to traffic or even topology changes, or the time needed for all routers to respond to congestion changes.

Even though current Internet routing protocols do not handle congestion or multi-path routing well, the necessary and sufficient conditions for minimum-delay routing (or minimum-congestion routing) are well known [119] and practical approaches that provide very good approximations to optimum routing using datagrams have existed for some time [126, 21]. Given the long time periods over which routers act as anchors of name prefixes, these approaches can be used in either routing to name prefixes or routing to the anchors of name prefixes. Furthermore, existing name-based routing protocols can easily apply mechanisms to react to congestion, namely: multi-path routing, maintaining congestion information about local interfaces, and path-based measurements of congestion. Accordingly, it can be safely assumed that congestion-oriented multi-path routing is attained independently of whether routes are established for name prefixes or anchors.

The following theorem formalizes the result illustrated in Figure 1 for the case of shortest-path routing (single path or multi-path) to name prefixes. We show that, if the same name-based routing protocol is used to establish routes to anchors and to name prefixes, the paths traversed by Interests are the same or mostly the same.

Theorem 7. *The paths to name prefixes obtained using forwarding entries listing the name prefixes are the same as the paths obtained using forwarding entries listing the anchors of name prefixes in a stable network in which a correct name-based routing protocol is executed.*

Proof. Consider a content-centric network in which forwarding entries list the next hops to the anchors of name prefixes. Assume that the routing protocol computes the paths to all the known anchors by time t_0 and that no changes occur in the network after that time.

Consider a name prefix P for which router a is an anchor, and assume for the sake of contradiction that, at some time $t_1 > t_0$, there is a route from a router r to the instance of prefix P announced by anchor a that is shorter than any of the routes from r to a implied by the forwarding tables maintained by routers at t_1. This is a contradiction to the definition of an anchor and the assumption that the routing protocol computes correct routes to all anchors by time t_0 and no changes occur after that. $\square$

Corollary 1. *For any name prefix that has a single anchor, the paths to the name prefix obtained using forwarding entries listing name prefixes are the same as the paths obtained using forwarding entries listing the anchors of name prefixes.*

Proof. The proof follows from Theorem 1 for the case of a stable topology. If forwarding tables are inconsistent due to network dynamics, the result follows from the one-to-one correspondence between a name prefix and its anchor, given that the prefix is hosted at a single site. $\square$

Figure 7.2 illustrates a content-centric network in which name prefix P^* is multi-homed. Assume that router o receives an Interest for a CO with a name in prefix P^* from consumer c, and the nearest anchor of P^* to router o is a_i. In the example, D_{ij} denotes the length of the pat from router i to router j, and $D^*_{ra_i}$ denotes the distance from r to a_i along a path that does not include p.

If routing is based on anchor names, router o binds the CO name to anchor a_i. Assume that the shortest path from o to a_i includes router r and that p is the next hop from r to a_i. Consider the case in which, because link (r, p) fails or becomes too congested, router r must find an alternative path to forward an Interest intended for anchor a_i. Anchor a_j is the closest anchor of prefix P^* to router r after the change in link (r, p). Routing based on name prefixes can provide more efficient forwarding than routing based on anchor names only if one of the following conditions is true:

$$D^*_{ra_i} < \infty \ \wedge \ D_{ra_j} < D^*_{ra_i} \tag{7.1}$$

$$D^*_{ra_i} = \infty \ \wedge \ D_{ra_j} < D_{ro} + D_{oa_j} \tag{7.2}$$

Only a small number of Interests pertaining to name prefixes that are multi-homed may be forwarded more efficiently if routing based on name prefixes is used. The reason for this is that Equations 1 and 2 cannot be satisfied over extended time periods. The name-based routing protocol operating in the control plane must make router o update its distances to anchors reflecting the change in link (r, p). Once router o updates its forwarding table, it must select a_i or another anchor as the new nearest anchor for P^* and Interests would traverse new shortest paths. We observe that this is the case even if mobility of hosting sites or consumers occurs, because the efficiency of Interest forwarding is determined by the distances between the routers attached to consumers and the routers attached to hosting sites (anchors).

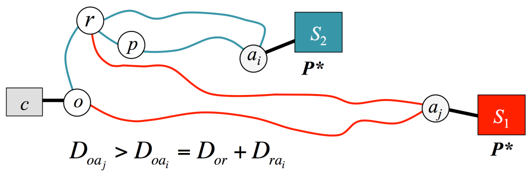


Figure 7.2: Forwarding Interests based on routes to anchors or routes to name prefixes

7.1.1 CCN-RAMP

7.1.2 Design Overview

CCN-RAMP allows routers to forward Interests by looking up small forwarding tables listing next hops to anchors. Clearly, for this to work, routers need to first obtain the mapping of the CO name stated in an Interest to an anchor of the name prefix that best matches the CO name [139].

CCN-RAMP takes advantage of the fact that existing name-based routing protocols communicate the anchor of a name prefix as part of an LSA or a distance update for the prefix [10, 124, 125]. Using the information disseminated in the name-based routing protocol, a router builds and maintains two tables: A Forwarding to Anchors Base (FAB) listing the routes to anchors, and a Prefix Resolution Table (PRT) listing the anchors of each name prefix.

A router receiving an Interest from a local consumer (call it origin router) uses its PRT to bind the CO name to the nearest anchor for the name prefix that is the best match for the CO name. To allow relaying routers to use only their FABs to forward Interests, an Interest states the name of the anchor chosen by the origin router. The origin router and other relaying routers forward the Interest as needed using their FABs and the anchor name in the Interest.

Routers in CCN-RAMP use exactly the same amount of routing signaling as routers in NDN and CCNx, and a PRT in CCN-GRAM has as many entries as a FIB in NDN and CCNx. However, the amount of routing signaling is not a problem, because anchors of name prefixes change infrequently and the name-based routing protocol can send updates regarding anchor-prefix bindings independently of updates regarding how to reach anchors. The limitation of using a FIB listing name prefixes is not its size but the need to look up the FIB in real time for each Interest being forwarded, which either requires very expensive memory or renders very slow lookup times.

In NDN and CCNx, each router forwarding an Interest must look up a FIB based on name prefixes because name resolution and routing are merged into one operation. By contrast, name resolution in CCN-RAMP is delegated to the origin routers that receive Interests from local consumers. Each router is in effect a name resolver. An origin router that binds a CO name to an anchor by looking up its PRT in response to an Interest from a local consumer replaces the role of the DNS to resolve a name into an address. However, consumers are relieved from being involved in name resolution. What can take hundreds of milliseconds using the DNS takes just a lookup of the PRT, which can be implemented using tries and slow storage. Once the binding of a CO name to an anchor is done by an origin router, forwarding an Interest involves fast lookups of FABs that can be even smaller than the FIBs used in the IP Internet today because only anchors are listed.

The tradeoff made in CCN-RAMP compared to prior approaches that separate name resolution from address-based routing is the need for each router to store a PRT. This is acceptable, given the cost of memory today and the infrequency with which PRT entries must be updated.

The design of CCN-RAMP eliminates forwarding loops by ordering the routers forwarding Interests based on their distances to destinations [37]. To attain this, each Interest carries the distance to an anchor and FABs list the next hops *and* the distances to anchors.

CCN-RAMP extends our prior work [108, 109] to eliminate the use of PITs while providing the same degree of Interest anonymity enabled in NDN and CCNx. A router maintains a Label Swapping with Anchors Table (LSAT) to remember the reverse paths traversed by Interests, and uses anonymous identifiers (AID) with local scope to denote the origins of Interests. The origin of an Interest is denoted with an AID that is swapped at each hop.

7.1.3 Assumptions

We make a few assumptions to simplify our description of CCN-RAMP; however, they should not be considered design requirements. For convenience, a request for content from a local user is sent to its local router in the form of an Interest.

Interests are retransmitted only by the consumers that originated them, rather than routers that relay Interests, routers forward Interest based on LPM, and a router can determine whether or not the CO with the exact same CO name is stored locally. Routers know which interfaces are neighbor routers and which are local consumers, and forward Interests on a best-effort basis.

Allowing anchors with only subsets of the COs in the name prefixes they announce requires routers to determine which of the anchors announcing the name prefix actually host the requested CO. This can be accomplished in CCN-RAMP using the multi-instantiated destination spanning trees (MIDST) described in [10, 11]. The pros and cons of allowing anchors to host only subsets of the COs in name prefixes, and the search mechanisms needed to support it, are the subject of another publication. We assume that each anchor of a name prefix is required to have *all* the COs in the name prefix locally available.

The name of content object (CO) j is denoted by $n(j)$ and the name prefix corresponding to the longest prefix match for name $n(j)$ is denoted by $n(j)^*$. The set of neighbors of router i is denoted by N^i.

subsectionInformation Exchanged

An Interest forwarded by router k requesting CO with name $n(j)$ is denoted by $I[n(j), AID^I(k), a^I(k), D^I(k)]$, and states the name of the CO $(n(j))$, a fixed-length anonymous identifier $(AID^I(k))$ denoting the origin router of the Interest, the anchor selected by the first router processing the Interest $(a^I(k))$, and the distance $(D^I(k))$ from k to $a^I(k)$.

A data packet sent by router i in response to an Interest is denoted by $DP[n(j), AID^R(i), sp(j)]$ and states, in addition to a CO, the name of the CO being sent $(n(j))$, an anonymous identifier $(AID^R(i))$ denoting the router that should receive the data packet, and a security payload $(sp(j))$ used optionally to validate the CO.

An error message sent by router i in response to an Interest is denoted by $ERR[n(j)$, $AID^R(i)$, $a^R(i)$, $\mathsf{CODE}]$ and states the name of a CO $(n(j))$, an anonymous identifier $(AID^R(i))$ that states the intended recipient of the reply, the selected anchor for the name prefix $(a^R(i))$, and a code (CODE) indicating the reason why the reply is sent. Possible reasons for sending a reply include: an Interest loop is detected, no route is found towards requested content, no content is found, and an upstream link is broken.

7.1.4 Information Stored

Router i maintains three tables for packet forwarding: A Prefix Resolution Table (PRT^i), a Forwarding to Anchors Base (FAB^i), and a Label Swapping with Anchors Table

($LSAT^i$). If router i has local consumers, it maintains a Local Request Table (LRT^i). Router i maintains a Content Store (CS^i) if it provides content caching locally.

PRT^i is indexed by the known name prefixes advertised by their anchors. Each entry of the PRT^i states the names of the selected anchors that advertised the prefix. Depending on the specific approach, the list may state the nearest anchors or all the anchors of the name prefix. However, with the assumption that anchors must have all COs of the name prefixes they announce, listing the nearest anchors for a name prefix suffices.

FAB^i is indexed by anchor names and each entry in FAB^i states available next hops to the anchor. The distance stored for neighbor q for anchor a in FAB^i is denoted by $D(i, a, q)$. This information is updated by means of a name-based routing protocol running in the control plane.

$LSAT^i$ is indexed by anonymous identifiers denoting origin routers. An anonymous identifier (AID) is simply a fixed-length number. Each entry in $LSAT^i$ states an AID locally created or received in Interests from a previous hop, the previous hop ($PH^i[AID]$) that provided the AID, a next hop ($NH^i[AID]$), the mapped AID ($MAP^i[AID]$) that should be used for the next hop, and the distance ($D^i[AID]$) to the anchor that should receive the forwarded Interests.

LRT^i lists the names of the COs requested by router i on behalf of local consumers. The entry for CO name $n(j)$ states the name of the CO ($n(j)$) and a list of the identifiers of local consumers (denoted by $lc[n(j)]$) that have requested the CO. CS^i lists the COs cached locally. The entry for CO name $n(j)$ states a pointer to the content of the CO (denoted by $p[n(j)]$).

7.1.5 Avoiding Forwarding Loops

Let S_a^i denote the set of next-hop neighbors of router i for anchor a. Router i uses the following rule to ensure that Interests cannot traverse forwarding loops, even if the forwarding data maintained by routers regarding name prefixes and anchors is inconsistent or contains routing-table loops.

Anchor-Based Loop-Free Forwarding (ALF):

If router i receives $I[n(j), AID^I(k), a^I(k) = a, D^I(k)]$ from router k, it can forward $I[n(j), AID^I(i), a^I(i) = a, D^I(i)]$ if:

1. $AID^I(k) \notin LSAT^i \wedge \ \exists v \in S_a^i \ (\ D^I(k) > D(i,a,v)\ )$
2. $AID^I(k) \in LSAT^i \ \wedge \ (\ D^I(k) > D^i[AID^I(k)]\ )$

Theorem 8. *No Interest can traverse a forwarding loop in a content-centric network in which CCN-RAMP is used.*

Proof. Consider a network in which CCN-RAMP is used and assume that, following the operation of CCN-RAMP, there is a router v_0 that originates an Interest for CO $n(j)$, uses longest match prefix to obtain the name prefix $n(j)^*$, and binds that name prefix to anchor a. The Interest sent by v_0 is $I[n(j), AID^I(v_0), a^I(v_0) = a, D^I(v_0)]$.

For the sake of contradiction, assume that routers in a forwarding loop L of h hops $\{v_1, v_2, ..., v_h, v_1\}$ forward the Interest for CO $n(j)$ originated by v_0 along L, with no router in L detecting that the Interest has traversed loop L.

Given that L exists by assumption, router $v_k \in L$ must forward $I[n(j), AID^I(v_k), a^I(v_k) = a, D^I(v_k)]$ to router $v_{k+1} \in L$ for $1 \leq k \leq h-1$, and router $v_h \in L$ must forward $I[n(j),$

$AID^I(v_h),\ a^I(v_h) = a, D^I(v_h)]$ to router $v_1 \in L$. According to ALF, if router v_k forwards Interest $I[n(j),\ AID^I(v_k), a, D^I(v_k)]$ to router v_{k+1} as a result of receiving $I[n(j),$ $AID^I(v_{k-1}),\ a, D^I(v_{k-1})]$ from router v_{k-1}, then it must be true that

$$AID^I(v_{k-1}) \notin LSAT^{v_k} \wedge [\ D^I(v_{k-1}) > D(v_k, a, v_{k+1})]$$

or

$$AID^I(v_{k-1}) \in LSAT^{v_k} \wedge [\ D^I(v_{k-1}) > D^{v_k}[AID^I(v_{k-1})\ ].$$

Similarly, if router v_1 forwards Interest $I[n(j), AID^I(v_1),\ a, D^I(v_1)]$ to router v_2 as a result of receiving $I[n(j),\ AID^I(v_h),\ a,\ D^I(v_h)]$ from router v_h, then

$$AID^I(v_h) \notin LSAT^{v_1} \wedge [\ D^I(v_h) > D(v_1, a, v_2)]$$

or

$$AID^I(v_h) \in LSAT^{v_1} \wedge [\ D^I(v_h) > D^{v_1}[AID^I(v_h)]\ ].$$

Given that each router in loop L that forwards an Interest for a given AID for the first time must create an entry in its LSAT, it follows from the above argument that, for loop L to exist and be undetected when each router in the loop uses ALF to forward the Interest originated by router v_0, it must be true that

$$D^I(v_{k-1}) > D^{v_k}[AID^I(v_{k-1})]\ for\ 1 < k \leq h \tag{7.3}$$

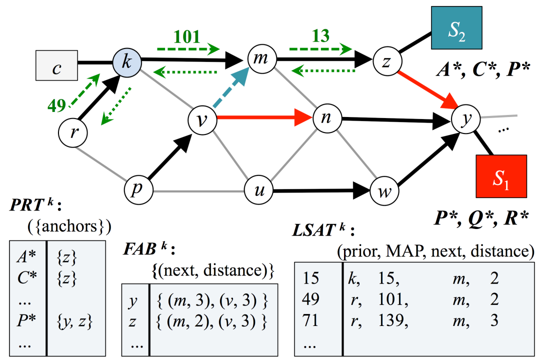
$$D^I(v_h) > D^{v_1}[AID^I(v_h)]. \tag{7.4}$$

However, Eqs. (3) and (4) constitute a contradiction, because they imply that $D^I(v_k) > D^I(v_k)$ for $1 \leq k \leq h$. Therefore, the theorem is true. $\square$

Theorem 2 is independent of whether the network is static or dynamic, the specific caching strategy used in the network, the retransmission strategy used by content consumers or relay routers after experiencing a timeout or receiving a reply, or whether routers use multiple paths or a single path to forward Interests towards a given anchor. We should also point out that ALF is a sufficient condition to ensure loop-free Interest forwarding, and it is possible that more flexible loop-free forwarding rules could be found. This is the subject of future work.

7.1.6 Interest Forwarding

Figure 7.3 illustrates the forwarding of Interests in CCN-RAMP showing two anchors (y and z) and five name prefixes, one of them (P^*) is multi-homed at y and z. The figure shows the forwarding tables used at router k (PRT, FAB and LSAT). Not shown are the content store and the LRT at router k.



PRT^k:
({anchors})

A*	{z}
C*	{z}
...	
P*	{y, z}
...	

FAB^k:
{(next, distance)}

y	{ (m, 3), (v, 3) }
z	{ (m, 2), (v, 3) }
...	

$LSAT^k$:
(prior, MAP, next, distance)

15	k, 15, m, 2
49	r, 101, m, 2
71	r, 139, m, 3
...	

Figure 7.3: Interest forwarding in CCN-RAMP

When router k receives an Interest from a local consumer c for a CO with name

$n(j)$, it looks up its content store (CS^i) to determine if the CO is stored locally. If the CO is remote, router k adds c to an entry in LRT^k stating the list of consumers that have requested $n(j)$ and proceeds to create Interest $I[n(j),\ AID^I(k),\ a^I(k),\ D^I(k)]$. The LRT entries allow router k to demultiplex responses it receives for AIDs that it originates and send the responses to the correct consumers.

Router k looks up PRT^k for the name prefix that provides the best match and selects an anchor of that prefix to be included in the Interest ($a^I(k)$). The router then looks up FAB^k to obtain the next hop m and the distance to the selected anchor. Router k includes its distance to the anchor ($D^I(k)$) in its Interest, so that forwarding routers can apply ALF as described in Section 4.5. If no entry exists in $LSAT^k$ with Router k as the prior hop, it selects an anonymous identifier ($AID^I(k)$) to denote itself as the origin of the Interest, such that the identifier is not being used by k to denote any other origin of Interests in $LSAT^k$. To select new AIDs, router i maintains a hash table or an array of bits that keeps track of previously used random numbers. An alternative approach could be using a counter that is increased after creating a new mapped AID (MAP).

In the example of Figure 7.3, router k has forwarded an Interest from a local consumer and used 15 as the anonymous identifier (AID) to identify itself. Router k can use the same AID in all Interests it sends towards any anchor on behalf of local consumers, or use different AIDs.

If router k forwards Interest $I[n(j),\ AID^I(r),\ a^I(r),\ D^I(r)]$ from neighbor r to neighbor m, ALF is satisfied, and no entry for $AID^I(r)$ exists in $LSAT^k$, then router k computes an AID that is not used as the MAP in any entry in $LSAT^k$. Router k then creates the entry for $AID^I(r)$ in $LSAT^k$ stating: $PH^k(AID^I(r)) = r$, $NH^k(AID^I(r)) = m$, $MAP^k\ (AID^I(r)) = n$, and $D^k(AID^I(r)) = D(k, a^I(r), m)$.

Note that only the ingress router receiving an Interest from a local consumer (e.g., router k receiving an Interest from c in the example) needs to look up its PRT, which is of the same size as a FIB in NDN and CCNx.

Forwarding routers use only their FABs and LSATs. Furthermore, a router needs to lookup its FAB only when no forwarding state exists in its $LSAT$ for the AID given in an Interest received from a neighbor router. A forwarding router receiving an Interest looks up its LSAT. If forwarding state is already set in its LSAT for the AID stated in an Interest from a given neighbor, the router forwards the Interest without involving its FAB.

The forwarding state in $LSAT^k$ specifies the next hop and AID to be used to forward an Interest received with a given AID from a previous hop. In the example, router k maps $(AID = 49, prior = r)$ in the Interests received from r to $(AID = 101, next = m)$ in the Interests it forwards to m towards anchor z. The dashed green arrows in Figure 7.3 show the flow of Interests from r to anchor z and the flow of responses from z to r.

The AID swapping approach adopted in CCN-RAMP simplifies the approaches we introduced in [108, 109]. Its intent is to make Interest forwarding as simple and fast as label swapping in IP networks, without revealing the identities of the consumers or routers that originate the Interests.

Figure 7.4 helps to illustrate the level of anonymity provided in NDN and CCNx. Interests in NDN only state the name of the requested CO and a nonce, and the per-Interest forwarding state stored in PITs is what allows routers to forward responses to Interests over the reverse paths traversed by the Interests. A third party monitoring traffic

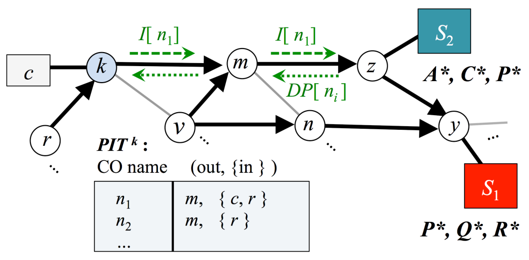


Figure 7.4: Interest forwarding in NDN

cannot determine the origin of an Interest simply from the information in the headers of Interests. However, the origin of an Interest can be obtained if routers collaborate and trace back the path the Interest traversed using the PIT entries listing the CO name in the Interest.

The same type of anonymity is provided in CCN-RAMP, but without the need for per-Interest forwarding state. A third party monitoring traffic cannot determine the source of an Interest simply by reading the information in the header of the Interest, because a local identifier is used to denote the source of an Interest at each hop. However, routers can collaborate to trace back the origin of Interests by means of the LSAT entries stored by the routers.

7.1.7 Updating Forwarding State

Algorithms 1 to 4 show the steps taken by routers to maintain the forwarding state needed to forward Interests, COs and error messages. The algorithms assume that FAB^i and PRT^i are initialized and maintained by a routing protocol operating in the control plane (e.g., NLSR [125] or DCR [10]).

Algorithm 20 shows the steps taken by a router to process an Interest received from a local consumer or a neighbor router, which were discussed in Section 4.6. An Interest from a consumer is assumed to specify the name of a requested CO with the rest of the information being nil.

Algorithm 21 shows the steps taken when a data packet from router s is received at router i. Like an interest, a data packet contains an anonymous identifier $AID^R(s)$. Router looks up $LSAT^(i)$ for $AID^R(i)$. If no entry with $MAP = AID^R(i)$ exists, the router does not forward the packet any further. If a matching entry is found, the router checks if the previous hop stated in the $LSAT^(i)$ entry is a neighbor router or the router itself. If it is a neighbor node, it forwards the packet to the previous hop PH stated in the matched *entry*. Otherwise, the data packet is forwarded to the local consumers listed in $lc[n(j)]$ of the entry for $n(j)$ in LRT^i.

Algorithm 22 shows the steps taken when the link connecting router i to router s fails. In such a case, for each entry in $LSAT^i$ that states the next hop as router s, an error message including the AID of the *entry* is created and is sent back to the previous hop stated in the AID entry. This way, router i informs neighbor routers of the link failure. Router i also invalidates all the matching entries in $LSAT^i$.

Algorithm 23 shows the steps followed after an error message from router s is received at router i. The received error message contains an AID set by the neighbor router. Router i looks up $LSAT^i$ and for any entry with $MAP = AID^R(s)$, it creates an error message containing $AID(entry)$ and sends it to the previous hops or the local consumers. The router also invalidates the entry found.

7.2 Performance Comparison

We implemented CCN-RAMP in ndnSIM [1] based on Algorithms 1 to 4 and used the NDN implementation in ndnSIM without modifications to compare NDN with CCN-RAMP. We also compare CCN-RAMP against our previous proposal for the elimination of PITs using datagrams, CCN-GRAM [109], which also relies on name-based routing in the control plane and uses FIBs listing the next hops to known name prefixes to forward Interests.

The performance metrics used for comparison are the average sizes of forwarding tables, the average number of table lookups needed to obtain one CO, the average end-to-end delays, and the average number of Interests sent by routers. DCR [10] is used in the control plane to update FIBs listing name prefixes in NDN and CCN-GRAM, or the Prefix Resolution Tables (PRT) and Forwarding to Anchor Bases (FAB) used in CCN-RAMP. Accordingly, we do not need to consider the signaling overhead of the name-based routing protocol, because it is exactly the same in the three approaches we consider.

We considered networks with no caching and with on-path caching, with each cache being able to store only 1000 COs. We used the AT&T network topology, which is considered to be a realistic topology for simulations [123]. This topology includes 153 nodes and 184 point-to-point links with 30 ms delay. To reduce the effects derived from sub-optimal implementations of CCN-RAMP, NDN, or CCN-GRAM, we set the data rate of point-to-point links to 10Gbps.

We selected 70 nodes randomly to have a consumer application simulating local consumers. All consumers generate Interests requesting COs from all name prefixes following a Zipf distribution with parameter $\alpha = 0.7$. The total number of COs is only 10^7, with 1000 COs per name prefix. We used a relatively small content population, because using larger numbers for COs and name prefixes would just make CCN-RAMP look better compared to NDN.

We selected 20 of the nodes randomly to be anchors of 500 different name prefixes each, and each name prefix has a single anchor.

Given that prefixes are not multi-homed in our simulation scenarios, the results in Section 3 indicate that the paths traversed by Interests should be the same whether forwarding tables maintain routes to anchors or to prefixes. This is the case for both single-path and multi-path routing. Accordingly, for simplicity, our simulation experiments assume single-path routing and static topologies. The difference between CCN-RAMP and CCN-GRAM [109] (its counter-part based on FIBs listing name prefixes) is that the latter incurs more forwarding overhead by its use of FIBs.

7.2.1 Average Table Sizes

Figure 7.5 shows the average table sizes for NDN, CCN-RAMP, and CCN-GRAM on a logarithmic scale as a function of the rate at which Interests arrive at routers with local consumers, ranging from 100 to 2000 Interests per second.

The number of entries in the PRTs used in CCN-RAMP is the same as the number of entries in the FIBs used in NDN and CCN-GRAM, because both list an entry for each known name prefix. Given that the topology contains 20 producer nodes and each node has 500 different name prefixes, each PRT and FIB table is expected to have 10,000 entries.

For CCN-RAMP, the average number of FAB entries for all Interest rates is only

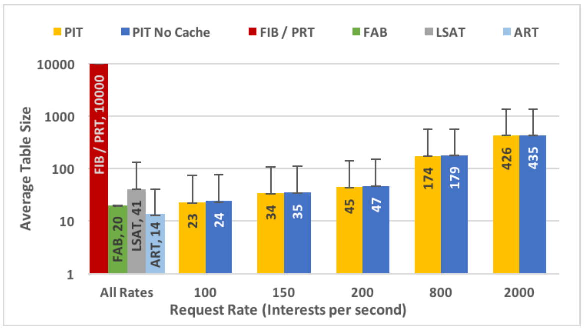


Figure 7.5: Average number of entries in forwarding tables for NDN, CCN-GRAM, and CCN-RAMP

20 and the average size of an LSAT, which is used to forward responses to Interests back to consumers, is only 41 entries for all values of Interest rates. The small sizes of FABs and LSATs should be expected, because there are only 20 routers acting as anchors, and each router acts as a relay of only a fraction of the paths to such anchors.

The average size of the forwarding table used in CCN-GRAM to send responses to Interests towards consumers (called ART) is only 14 entries for all Interest rates.

By contrast, the number of PIT entries depends on network conditions and traffic load. The average PIT size varies from 23 or 24 to 426 or 435, depending on whether on-path caching is used. The size of PITs at some core routers can be more than 1000 entries when the request rate at routers with local consumers is 2000 Interests per second. Interestingly, the average number of PIT entries is not much smaller when on-path caching is used compared to the case in which no caching is used.

7.2.2 Average Number of Table Lookups

Figure 7.6 shows the average number of table lookups required to retrieve a single CO, and includes all lookups done in forwarding the Interest for the CO and sending back the corresponding CO to the requesting origin router.

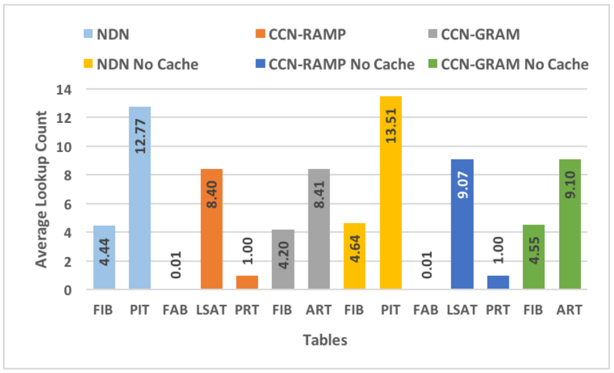


Figure 7.6: Average number of table lookups needed to retrieve one CO

In NDN, forwarding an Interest requires a PIT lookup at each hop along the path from consumer to caching site or anchor. If no PIT entry is found, a FIB lookup is required to obtain the next hop for the Interest. Forwarding a data packet sent from an anchor or a caching site requires a PIT lookup at every hop along the way to the consumer.

In CCN-RAMP, retrieving a remote CO includes one PRT lookup at the router that receives the Interest from a local consumer. That router binds the CO name to an anchor and each router along the path towards the anchor must do one FAB lookup to forward the first Interest with an AID that does not exist in the LSAT of the router, and one LSAT lookup for every Interest being forwarded. Once forwarding state is established along a path from an originating router to an anchor, no FAB lookups are needed for Interests that carry AIDs already listed in the LSATs of the relaying routers. As Figure 7.6 shows, the average number of FAB lookups per Interest is a very small fraction, because only a small fraction of Interests are forwarded without having any forwarding state already established in the LSATs of routers.

In CCN-GRAM, retrieving a remote CO involves one FIB lookup at each hop along the path from consumer to caching site or anchor, as well as a lookup of the ART in order to carry out the proper swapping of AIDs. Forwarding a data packet sent from an anchor or a caching site requires only an ART lookup at each hop along the way to the consumer.

Compared to NDN and CCN-GRAM, CCN-RAMP results in relay routers doing fewer lookups of tables that are three orders of magnitude smaller than FIBs listing name prefixes, even for the small scenario we consider.

It can be inferred from Figure 7.6 that the average hop count for paths traversed by Interests is around four or five hops, because this is the approximate number of average FIB lookups needed in NDN and CCN-GRAM when no caching is used. As should be expected, the average number of table lookups in NDN, CCN-GRAM, and CCN-RAMP is slightly larger when no on-path caching is available, because the average paths between consumers requesting COs and the anchors are longer than the average paths between consumers and caching sites.

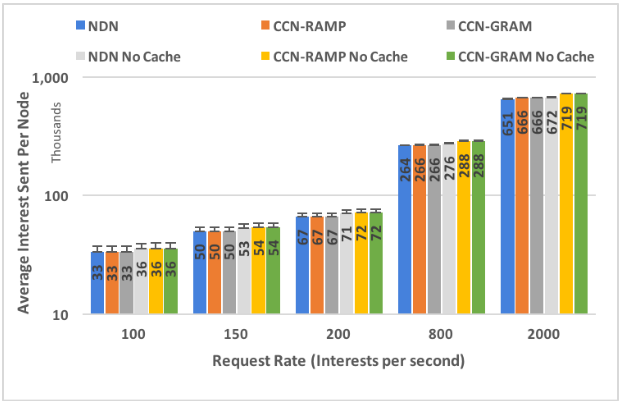


Figure 7.7: Average number of Interests forwarded per router

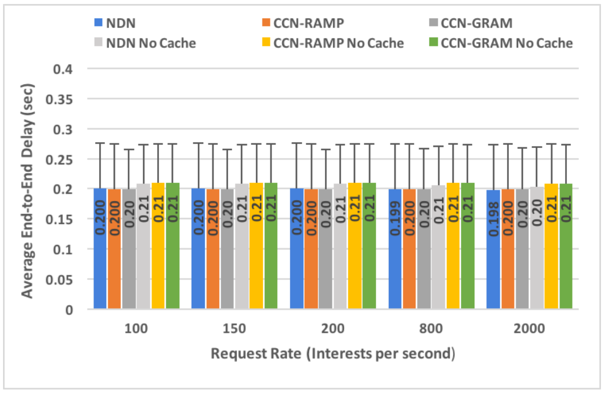


Figure 7.8: Average end-to-end delays

7.2.3 Average Number of Interests and End-to-End Delays

Figure 7.7 shows the average number of Interests sent by each router. As the results indicate, the average encumber of Interests sent by each router is essentially the same for all three approaches and the percentage of Interests that benefit from aggregation using PITs is insignificant.

Figure 7.8 shows that the average end-to-end delays are very similar in all cases for NDN and CCN-RAMP. Given that the simulations assume zero delays for table lookups (i.e., the differences in forwarding table sizes are not taken into account), these simulation results indicate that the paths traversed by Interests are the same for NDN, CCN-GRAM, and CCN-RAMP. This confirms that the paths traversed by Interests when routers maintain FIBs with entries for name prefixes tend to be the same as the paths traversed by Interests if the origin routers select the anchors of name prefixes and routers forward Interests towards anchors.

7.3 Conclusions

Scaling has been identified as a major research problem for content-centric networks [71]. We introduced CCN-RAMP, a new approach to content-centric networking that can be deployed at Internet scale and is able to handle billions of name prefixes, because it eliminates the need to lookup large FIBs listing name prefixes, and the use of PITs that make routers vulnerable to DDoS attacks on the routing infrastructure. CCN-RAMP provides all the benefits sought by NDN and CCNx, including native support of multicasting without the need for a new multicast routing protocol (see [109, 121]). In contrast to NDN and CCNx, Interests cannot traverse forwarding loops and no Interest-flooding attacks can be mounted.

The results of simulation experiments based on implementations of NDN, CCN-GRAM, and CCN-RAMP in ndnSIM show that CCN-RAMP is more efficient than NDN and CCN-GRAM. CCN-RAMP rendered similar end-to-end delays, incurred similar Interest overhead in the data plane, and resulted in forwarding state with a number of entries that can be orders of magnitude smaller than the forwarding state required in NDN.

Algorithm 20 Processing Interest from p

```
function Interest_Forwarding
INPUT: $LIST^i$,$FAB^i$, $PRT^i$, $LSAT^i$;
INPUT: $I[n(j), AID^I(p), anchor, D^I(p)]$;
if $p[n(j)] \in CS$ then
    retrieve CO $n(j)$; send $DP[n(j), AID^R(i), sp(j)]$ to $p$
else
    if p is consumer then
        $lc[n(j)] = lc[n(j)] \cup c$;
        for each $a$ by rank in $PRT^i(n(j))$ do
            $anchor = a$; break;
        end for
        $aid = f(anchor)$;
    else
        $aid = AID^I(p)$;
    end if
    $entry = nil$;
    for each $e \in LSAT^i_{aid}(aid)$ do
        if $PH(e) = p$ then
            $entry = e$; $break$;
        end if
    end for
    $loop = true; noRoute = true$;
    if $entry = nil$ then
        for each $s \in N^i$ by rank in $FAB^i(anchor)$ do
            NoRoute=false;
            if $D^I(p) > D(i, n(j)^*, s)$ (% ALF is satisfied) then
                loop=false; $D^I(i) = D(i, n(j)^*, s)$; $NH = s$;
                break;
            end if
        end for
        select unused random number $map$;
        $entry$ =create entry $LSAT^i[aid, p, map, NH, D(i, n(j)^*, s)]$;
    end if
    if $entry \neq nil$ then
        send $I[n(j), MAP(entry), anchor, D(entry)]$ to $NH(entry)$; return;
    end if
    if $noRoute = false \wedge loop = true$ then
        send $ERR[n(j), AID^I(p), anchor,$ loop$]$ to $p$ ;
    else
        send $ERR[n(j), AID^I(p), anchor,$ no route$]$ to $p$
        (% No route to $n(j)^*$ exists);
    end if
end if
```

Algorithm 21 Processing data packet from router s

```
function Data Packet
INPUT: $LIST^i$, $LST^i$, $DP[n(j), AID^R(s), sp(j)]$;
[o] verify $sp(j)$;
[o] if verification with $sp(j)$ fails then
     discard $DP[n(j), AID^R(s), sp(j)]$;
$entry = LSAT^i_{map}(AID^R(s))$; ( %LST in case of CCN-RAMP)
if $entry = nil$ then
    $drop$; $return$;
end if
if  $preHop(entry) = local$  (% router $i$ is the origin)  then
    for each $c \in lc[n(j)]$ do
        send $DP[n(j), nil, sp(j)]$ to $c$; $lc[n(j)] = lc[n(j)] - \{c\}$
    end for
else
    send $DP[n(j), AID(entry), sp(j)]$ to $preHop(entry)$;
end if
store CO in CS
```

Algorithm 22 Failure of link l connected to router i to p

```
function Data Packet
INPUT: $LSAT^i$;
for each $entry \in LSAT^i$ with $NextHop(entry) = p$ do
    $preHop = getNodeID(AID(entry))$;
    send $ERR[nil, AID(entry), \text{link failure}]$ to $preHop$
    INVALIDATE(entry);
end for
```

Algorithm 23 Processing Error Message from Router s at router i

```
function ERR
INPUT: $LSAT^i, ERR[n(j), AID^R(s), reason]$;
for each $entry \in LSAT^i_{map}(AID^R(s))$ do
    $preHop = getNodeID(AID(entry))$;
    if  $preHop = local$  (% router $i$ was the origin of the Interest)  then
        for each $c \in lc[n(j)]$ do
            send $DP[n(j), c, sp(j)]$ to $c$; $lc[n(j)] = lc[n(j)] - \{c\}$
        end for
    else
        send $ERR[nil, AID(entry), \text{reason}]$ to $preHop$;
    end if
    INVALIDATE(entry);
end for
```

Chapter 8

Making On-Demand Routing Efficient with Route-Request Aggregation

Many routing protocols have been proposed for mobile ad-hoc networks (MANET), and can be categorized as proactive, reactive, and hybrid routing protocols [79, 80, 90, 94, 99]. Proactive or table-driven routing protocols maintain routes to every network destination independently of the data traffic being forwarded. Reactive or on-demand routing protocols maintain routes for only this destinations for which there are data packets to be forwarded. Hybrid protocols use proactive and on-demand mechanisms.

The proactive routing approach has the potential of high packet-delivery ratios and shorter end-to-end delays, because routes are established before data packets requiring those routes are offered to the network. The price paid for such responsiveness is that signaling overhead is incurred even for those destinations that are not needed, which may be too high. In theory, on-demand routing is designed to address this problem by requiring signaling overhead only for active destinations at the expense of incurring slightly longer latencies, because some data packets must wait for routes to be found. However, as prior comparative analysis of the performance of on-demand versus proactive routing schemes show [81, 83, 87, 96, 103], on-demand routing protocols end up incurring more overhead than proactive routing protocols in MANETs when topology changes that impact existing data flows increase.

Many techniques (e.g., see [79, 80, 94]) have been proposed to reduce the overhead incurred in the dissemination of each route request (RREQ), including clustering, location information, dominating sets, and virtual coordinates. However, no prior work has addressed the impact of having relay routers aggregate RREQs they need to forward when they are intended for the same destinations.

The main contribution of this chapter is the introduction of a fault-tolerant approach for routers to aggregate RREQs originated by different sources and intended for the same destinations. The proposed *route-request aggregation* approach can be applied to any on-demand routing protocol (e.g., AODV or DSR [94]) and can make any routing protocol that uses on-demand routing techniques more efficient.

Section 8.1 introduces the Ad-hoc Demand-Aggregated Routing with Adaptation (ADARA) protocol as a specific example of the RREQ aggregation approach. Like AODV, ADARA uses destination-based sequence numbers to prevent routing-table loops and request identifiers to denote each RREQ uniquely as in AODV. ADARA introduces route-request aggregation and the use of broadcast signaling packets (RREQs, route replies and route errors) to substantially reduce signaling overhead.

Section 8.2 presents an example of the operation of ADARA and how it improves performance compared to AODV [94]. However, the approach used in ADARA can be applied with proper modifications to on-demand routing based on source routes (e.g., DSR [94]) or path information [86]. It can also be used in combination with prior techniques aimed at reducing signaling overhead, such as the use of geographical coordinates of destinations [89, 102], virtual coordinates, connected dominating sets [101], address aggregation [100], and clustering [80, 93].

Section 8.3 presents the results of simulation experiments used to compare ADARA with two routing protocols that are representative of the state of the art in proactive routing and on-demand routing for MANETs, namely OLSR [82] and AODV. The experiments were designed to study the impact of node speed, pause times, number of sources, and network size on the packet-delivery ratio, average end-to-end delay, and signaling overhead. The results show that ADARA performs better than OLSR and AODV in all cases. The key reason for this is that ADARA is able to establish routes on demand incurring far less overhead than AODV and OLSR.

8.1 ADARA

The design rational for ADARA is twofold. First, for the performance of an on-demand routing protocol to be comparable to or better than the performance of a proactive routing protocol, the number of RREQs that sources initiate in the route-discovery process must be kept to a minimum when the network supports many data flows and experiences topology changes [140]. Second, if the number of data flows intended for the same destination node is larger than the number of neighbors of that destination, the routes from the sources of the flows to the destinations *must* have some routing relays in common. Accordingly, allowing routers to aggregate RREQs intended for the same destination is bound to have a positive effect on the overall performance of the network.

ADARA (Ad-hoc Demand-Aggregated Routing with Adaptation) is the first on-demand routing protocol in which a router aggregates RREQs from different sources intended for the same destination.

ADARA adopts the use of destination-based sequence numbers as in AODV to avoid routing-table loops, as well as the use of the source address and a request identifier created by the source to identify each RREQ. Other approaches have been proposed to avoid routing-table loops when routers maintain routes on-demand [84, 86, 91, 95, 97] and can be used instead of the specific approach based on destination sequence numbers.

8.1.1 Information Exchanged and Stored

ADARA uses four types of signaling packets, all of which are sent in broadcast mode.

A Route Request (**RREQ**) is denoted by $REQ[RID,\, o,\, on,\, d, dn,\, ho,\, HSN]$ and contains: A request identifier (RID), the address of the origin or source of the RREQ (o), a sequence number created by the origin (on), the address of the intended destination (d), the most recent sequence number known from d (dn), a hop count to the origin of the RREQ (ho), and a HELLO sequence number (HSN).

A Route Reply (**RREP**) is denoted by $REP[d\ , dn, hd, LDN, HSN]$ and contains: the address of the destination (d), the most recent sequence number known from d (dn), a hop count to the destination (hd), a list of designated neighbors (LDN) from which valid RREQs for destination d have been received, and a HELLO sequence number (HSN).

A Route Error (**RERR**) is denoted by $RE[HSN, LUA]$ and contains a HELLO sequence number (HSN) and a list of unreachable addresses (LUA).

A Hello message (**HELLO**) is denoted by $H[HSN]$ and contains the sequence number of the sending node.

Each router i maintains a routing table (RT^i) and a pending request table (PRT^i). Each entry of RT^i specifies: the address of the destination, a sequence number created by the destination, a hop count to the destination, next hop to the destination, a list of precursor neighbors for the destination, and a lifetime.

PRT^i is used to keep track of the RREQs received by router i, aggregate RREQs received for the same destination, and discard duplicates of the same RREQ. An entry in PRT^i lists a destination address, a list of precursor tuples, and a lifetime. Each precursor tuple consists of: the address of an origin node, the RID stated by that node, and the address of the precursor neighbor from which a RREQ was received.

ADARA is a soft-state protocol. Each entry in PRT^i and RT^i has a finite lifetime to allow router i to delete entries that become obsolete as a result of topology changes (e.g., the network is partitioned or a node fails).

8.1.2 Updating Neighbor Connectivity

Whenever a router receives a Hello message, a RREQ, a RREP, or a RERR, it calls the Hello Process function shown in Algorithm 24 to update routes to neighbor routers. This process uses the HSN included in each signaling packet. The HSN a router includes in a RREQ, RREP or RERR is simply the value of its current sequence number.

Algorithm 24 Processing Hello

```
function Process_Hello
INPUT: $sender$, $r\_table^i$, $HelloSeqNo$;
$route = r\_table^i.lookup(sender)$;
$route.setHop(1)$;
$route.SetDes(sender)$;
$route.SetNextHop(sender)$;
$route.SetSeqNum(HelloSeqNo)$;
$route.mark(Valid)$;
$r\_table.update(route)$;
```

8.1.3 Route Discovery Process

A router originates a RREQ when it has no valid route to an intended destination as a result of topology changes or because a new destination is of interest to the router. Algorithm 25 shows the steps taken by a router to process a RREQ it receives from a neighbor.

After the neighbor information is updated according to Algorithm 24, router i updates its routing information regarding the origin of the RREQ. Router i uses Algorithm 26 to process the RREQ based on its origin, the RID created by the origin, and the entries in PRT^i.

Router i sends back a RREP to the RREQ it receives if it is the intended destination or RT^i contains a valid entry for the destination stated in the RREQ with a sequence number that is higher than or equal to the destination sequence number stated in the RREQ. The RREP is broadcast to all neighbors and states the hop count to the destination, the destination sequence number, a HELLO sequence number for itself, and the list of designated neighbors.

If router i has no valid route to the intended destination in the RREQ and there is no entry in PRT^i for that destination, router i creates a PRT^i for the destination and broadcasts the RREQ to its neighbor routers with its own HSN and its own hop count to the origin of the RREQ. On the other hand, if there is an entry for the destination in PRT^i, there are various cases to consider.

If the RREQ is a replica of a RREQ received from the same origin (i.e., there is a pending RREQ for the destination from the same origin and with the same RID), the RREQ is silently dropped. If the RREQ is not a replica of a RREQ already received, but is a retransmission of a RREQ from one of the origins of the request, it means that the origin is retransmitting its RREQ due to a timeout expiration. Accordingly, router i updates the RID of the corresponding precursor tuple and broadcasts the RREQ to its neighbor routers. Lastly, if the RREQ is from a different source than those listed in PRT^i, router i simply adds a precursor tuple PRT^i with the address of the origin, the RID created by the origin, and the address of the neighbor that sent the RREQ. We say that the RREQ is aggregated in such a case.

When router i receives a RREP, it updates its neighbor information according to Algorithm 24. Router i accepts the information in the RREP and updates RT^i for the

Algorithm 25 Process RREQ from router s at router i

```
function Process_RREQ
INPUT: $rreq$,$org$, $r\_table^i$,$Destination$;
$des = rreq.getDestination()$;
$processHello(s, RREQ.HelloSeqNo)$;
$aggregated = PRT.Aggregate(RREQ)$;
$UpdateReversePath(RREQ, org)$;
$rt = r\_table^i.lookup(des)$;
if $(rt) \wedge (rt.seq \geq rreq.Seq) \wedge (rt == VALID)$ then
    $rrep = create\_rrep(rt)$;
    $rrep.SetHelloSeq(LocalSeq)$
    $Broadcast(rrep)$;
else
    if !$aggregated$ then
        $rreq.SetHelloSeq(LocalSeq)$
        $Broadcast(rreq)$;
    end if
end if
```

destination stated in the RREP if either the destination sequence number is higher than the destination sequence number in RT^i or the sequence numbers are the same but the hop count to the destination in the RREP is smaller than the corresponding hop count in RT^i.

For the case of a valid RREP, router i creates or updates the entry in RT^i for the destination. The entry states the destination sequence number obtained in the RREP, its hop count to the destination, and the list of precursor neighbors for the destination. The precursor neighbors are simply those neighbors listed in precursor tuples for the destination in PRT^i. If the router is a member of LDN of RREP, then the router i broadcasts the RREP to its neighbors stating its own hop count to the destination, its own HELLO sequence number, and a list of designated neighbors of router i that need to process and perhaps forward the RREP. Router i can then delete the entry for the destination in PRT^i . In case

Algorithm 26 Aggregate RREQ i

```
function Aggregate_RREQ
INPUT: $rreq$, $PRT^i$;
$des = rreq.getDestination()$;
$org = rreq.getOrigin()$;
$id = rreq.getId()$;
if $\exists enrty \in PRT \wedge entry_{org} = org \wedge entry_{id} = id$ then
    $drop(rreq)$; //Duplicate RREQ
    $return\ true$;
end if
if $\exists e \in PRT \wedge e_{org} = org \wedge e_{des} = des \wedge e_{id} \neq id$ then
    $update(entry, rreq)$; //Retransmitted RREQ
    $return\ false$;
end if
if $\exists e \in PRT \wedge e_{org} \neq org \wedge e_{des} = des$ then
    $PRT.AddEntry(rreq)$; // Aggregate
    $return\ true$;
end if
$PRT.AddEntry(rreq)$;
$return\ false$;
```

the router is not in LDN, after updating the routes, router will drop the RREP to limit the region within which the RREP is re-broadcast.

8.1.4 Handling Errors and Topology Changes

Route error messages are created when no route is found toward a destination router or a link break is detected. A router assumes that a link with a neighbor is down when it fails to receive any signaling packet within interval defined for the reception of signaling packets from a neighbor. An error message states all the destinations for which routes are broken as a result of the link failure.

Algorithm 28 shows the steps taken by router i to process a RERR from a neighbor. Router i invalidates all the routes to destinations listed in the RERR that require the router

Algorithm 27 Processing RREP from router s at router i

```
function Process_RREP
INPUT: $rrep$,$sender$, $r\_table^i$,$Destination$;
$des = rrep.getDestination()$;
$processHello(sender, rrep.GetHelloSeqNo())$;
$rt = r\_table^i.lookup(des)$;
$intended = false$;
if $currentNode \in RREP.LDN()$ then
    $designated = true$;
end if
if $(rt\_des \neq empty)$ then
    if $(rrep.seq > rt\_des.seq) \vee (rrep.seq = rt\_des.seq \wedge rrep.hop < rt\_des.hop)$ then
        $rt\_des.update(rrep)$;
    end if
else
    $rt\_des = r\_table^i.AddRoute(RREP)$;
end if
if $designated \neq true \wedge PRT.lookup(Des_{RREP}).Count \leq 1$ then
    $return$;
end if
$RREP.ClearLDN()$;
for each $entry \in PRT.lookup(Des_{RREP})$ do
    $PRT.remove(des)$;
    $rt\_org = r\_table^i.lookup(entry.org)$;
    $RREP.LDN.Add(entry.PrecursorNeighbor)$;
    $rt\_des.AddPrecursor(entry.PrecursorNeighbor)$
end for
$RREP.setHelloSeq(LocalSeq)$;
$Broadcast(RREP)$;
```

sending the RERR as the next hop. Router i broadcasts a RERR it receives if at least one precursor neighbor exists for the destinations listed in the RERR. Accordingly, only routers that established routes to destinations by forwarding RREQs may have to forward RERRs.

Algorithm 28 Processing RERR from router s at router i

```
function Process_RERR
INPUT: $rerr$, $r\_table^i$,$unreachable$;
$processHello(s)$;
$rtList$ = Get All entries in $r\_table^i$ that use $s$ toward unreachable routers;
$hasPrecursor = false$;
for each $rt \in rtList$ do
    if $rt.precursorCount() > 0$ then
        $hasPrecursor = true$;
    end if
    $invalidate(rt)$;
end for
if $hasPrecursor$ then
    $RERR.SetHelloSeq(LocalSeq)$;
    $Broadcast(RERR)$;
end if
```

8.2 ADARA Example

Figure 8.1 shows a small wireless network in which ADARA is used. The network consists of six relay routers (m, n, o, p, q, and r), three source routers (S, A, and B), and one destination router D. The example assumes that no router has valid routes for destination D, and shows router S generating and broadcasting a RREQ for destination D at time t_1. The propagation of this RREQ is indicated by thin arrows in the figure, and the propagation of RREPs is shown with thick blue arrows. The RREQ from router S states $REQ[RID_S, S, on_S, D, dn = 0, ho = \infty, HSN_S]$

When router m receives the RREQ from source S, it adds a route for destination router S as a destination in RT^m with a hop count of one and S as the next hop. Router m also creates an entry for D in PRT^m listing the precursor tuple $[S, RID_S, S]$, which states S as the origin of the RREQ with a RID equal to RID_S and source S as the neighbor from

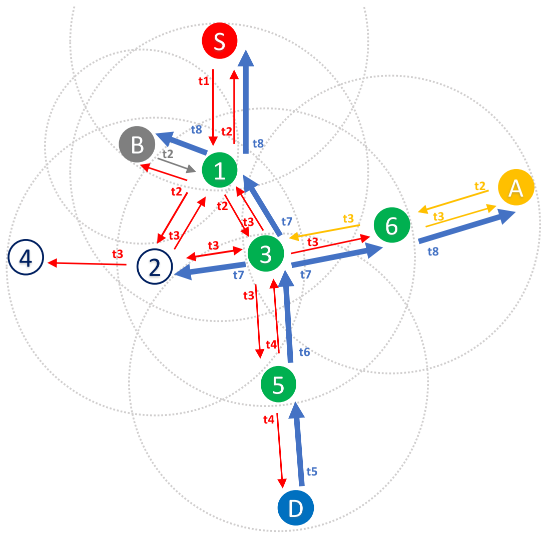


Figure 8.1: Dissemination of RREQs and RREPs in ADARA

which the RREQ was received.

The example shows routers A and B originating RREQs for destination D at time $t_2 > t_1$. As the figure shows, router r forwards the RREQ at time $t_3 > t_2$. However, when router m receives the RREQ from router B for destination D shortly after time t_2, it simply aggregates the RREQ, because PRT^m contains an entry for D. Router m does this by adding the precursor tuple $[B, RID_B, B]$ to its entry for destination D in PRT^m.

Router o creates an entry for D in PRT^o after receiving the RREQ forwarded by router m, and that entry lists the precursor tuple $[S, RID_S, m]$. Accordingly, when router o receives the RREQ forwarded by router r shortly after time t_3, it can simply aggregate the RREQ. It does this by adding the precursor tuple $[A, RID_A, r]$ to the entry for destination D in PRT^o. Similarly, when router r receives the RREQ forwarded by router o (originated by source S) shortly after time t_3, it already has an entry for destination D in PRT^r and hence aggregates the RREQ received from router o by adding the precursor tuple $[S, RID_S, o]$ to its list of precursor tuples for destination D.

We note that, shortly after time t_3, routers n and o receive the RREQ originated by source S from each other. Both routers simply ignore the replicas of the RREQ originated by router S because they each have an entry for destination D in their PRTs listing a precursor tuple with the same source router and source sequence number than the ones included in the RREQ they receive from each other.

As the RREQs from sources S, A, and B are disseminated in the network, relaying routers add precursor tuples to their PRTs for destination D. These tuples allow each relay router to decide whether to broadcast a RREP for D when it receives a RREP from a neighbor. Destination D generates a RREP for itself at time t_5 when it receives the RREQ from router q. Starting with router q, the RREP is disseminated back to the sources that originated RREQs for D along the reverse paths traversed by the RREQs thanks to the precursor tuples maintained in the PRTs of routers. Each relaying router re-broadcasts the RREP for destination D if it has at least one precursor tuple for D in its PRT, which results in RREPs being disseminated along a directed acyclic graph as illustrated in Figure 1. Each router that forwards a RREP copies the precursor neighbors for D to its RT.

A RREP contains the list of designated neighbors (LDN) that may forward the RREP as needed, and is based on the precursors stated for a given destination in the PRTs of routers. In the example, the LDN of the RREP from router q lists router o, and the LDN of the RREP from router o states routers m and r. Accordingly, as shown in Figure 1, when router n receives the RREP from router o, it does not forward the RREP, given that it is not listed in the LDN of the RREP from router o. However, router n adds a routing entry for D in RT^n. Routers m and r forward the RREPs they receive from router o.

Router r forwards the RREP with an LDN listing routers o and A. Router o simply ignores the RREP from r, and source A is able to start sending data packets to D. By the same token, routers that receive RREPs from the next hops to the sources of RREQs can ignore the RREPs because they are not listed in the LDNs of those RREPs.

In contrast to the above, AODV and other on-demand routing protocols would require the dissemination of the RREQs from S, A and B throughout the entire network, and for each origin router, a RREP would be sent on the path from source to destination.

Figure 8.2 shows the number of signaling packets sent in the topology of Figure 8.1. The number of RREQs in ADARA is much smaller compared to AODV, which is a direct result of RREQ aggregation. Using ADARA, the RREQ generated by A is only sent by routers A and r, and the RREQ from router B is sent once by router B and aggregated at router m. On the other hand, using AODV, the RREQs from routers S, B, and A are flooded in the network. The number of RREPs sent over the network in ADARA is also lower than AODV as a result of the aggregation or RREQs. In ADARA, RREPs are sent once on the path up to an aggregation point. In AODV, each RREP sent once on each path. As a result, the number of RREQs and RREPs in AODV is 2.5 times larger than in ADARA for this example. Furthermore, since all RREPs are broadcast messages, routers on the path from a source to a destination do not generate Hello messages for a time interval. For the case of AODV, Hello messages are generated independently of the RREPs being sent.

	ADARA	AODV
RREQ	11	27
RREP	5	12
SUM	16	39

Figure 8.2: **Dissemination of RREQs and RREPs in ADARA**

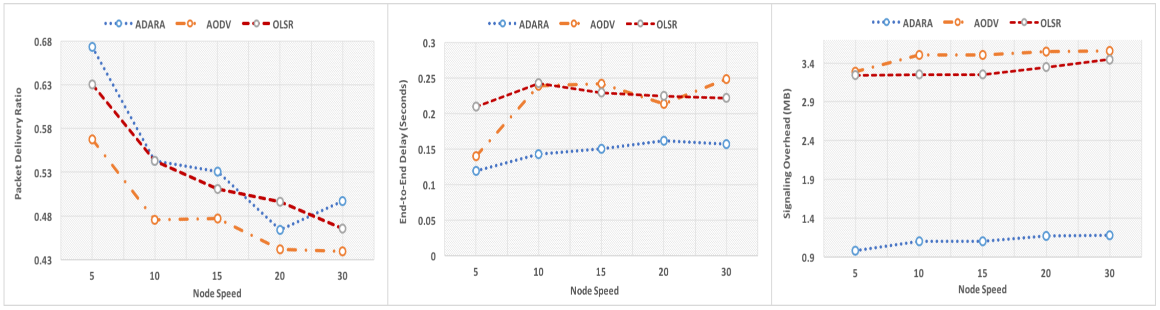


Figure 8.3: Performance comparison as a function of router speed.

8.3 Performance Comparison

8.3.1 Simulation Model and Parameters

We implemented ADARA in ns3 and used the ns3 implementations of AODV and OLSR without modifications to compare their performance. Figure 8.7 shows simulation-environment settings for AODV, OLSR, and ADARA.

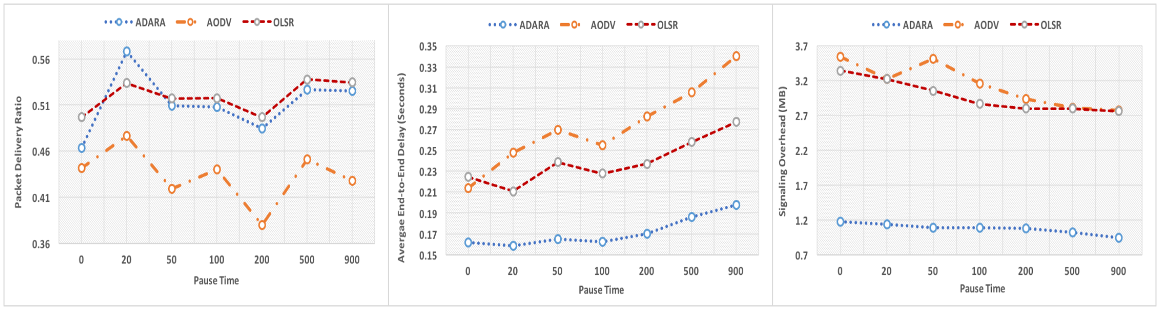


Figure 8.4: Performance comparison as a function of Pause Time.

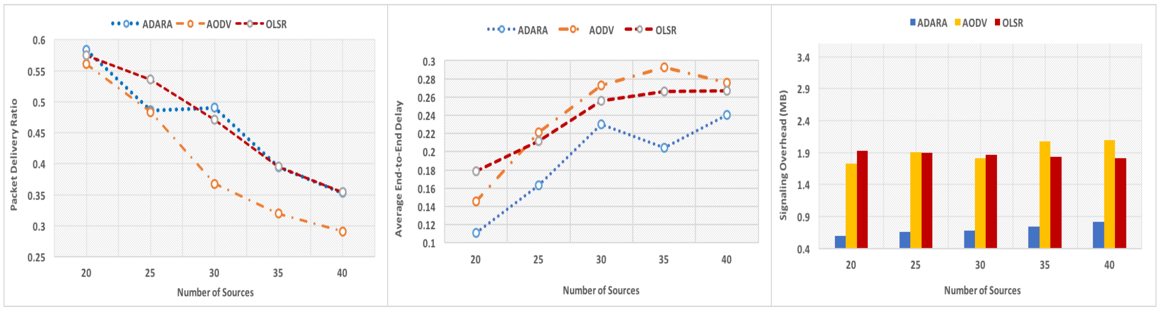


Figure 8.5: Performance comparison as a function Number of Sources.

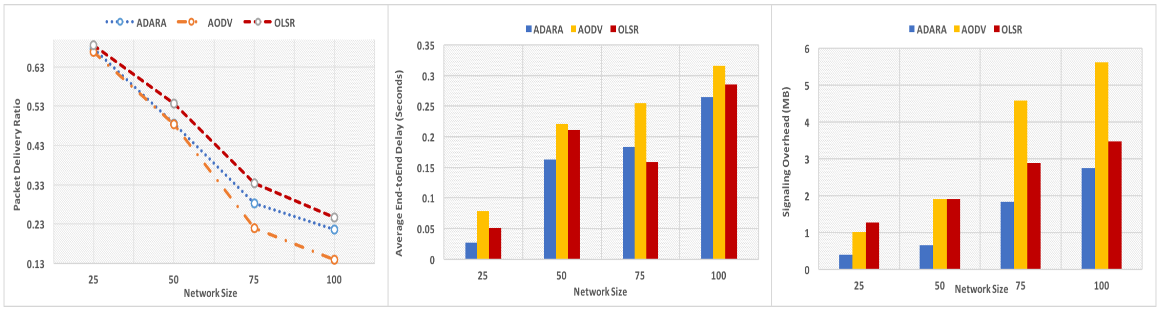


Figure 8.6: Performance comparison as a function Network Size.

Hello Interval	2 sec
Active Route Timeout	3 sec
Net Diameter	5
Max Queue Length	64
RREQ Rate Limit	10
RREQ Retries	2
TC Interval (OLSR)	5 sec

Figure 8.7: **Simulation Configuration for ADARA, AODV, and OLSR**

The Distributed Coordination Function (DCF) of IEEE 802.11n 2.4 Ghz with rate of 2Mbps is used as the MAC-layer protocol for unicast data transmission. To avoid incorrect paths resulting from transmission-range differences between unicast and broadcast transmissions, we made sure that both broadcast and unicast packets are sent with the same rate (2Mbps) and range. Transmission power is adjusted to fix the transmission range to 250 meters. Both AODV and ADARA use a sending buffer of 64 packets. These buffers store packets waiting for RREP message to the desired destination for 30 seconds.

Simulations include 50 routers spread uniformly in a $300m \times 1500m$ area. For other scenarios, 25, 75, and 100 routers are uniformly spread in a $300m \times 1500m$, $300m \times 1500m$, and $300m \times 1500m$ respectively. Routers use the random-waypoint mobility model with a randomly-chosen moving speed between 0 and 20 m/s and pause time of 0 seconds. A router chooses a destination location randomly and moves toward that destination with a randomly chosen speed between zero and the specified maximum speed. When a destination location is reached, the router remains there for a specified pause time.

The scenarios include 25 data flows from 25 different source routers. The destination router for each flow is a specific router with probability 0.5 and is chosen randomly from all routers with probability 0.5. Traffic sources are on-off applications with on and off time of 1 second, which generate packets of size 512 bytes and rate of 15 packets per second. For network sizes of 25 and 50 routers, simulations are run for 900 seconds, and for networks sizes of 75 and 100 routers, the simulation time is 500 seconds.

The signaling overhead in AODV includes its five types of packets: RREQ, RREP, RERR, Route Reply Ack, and HELLO messages. In OLSR, the signaling overhead includes, Topology Control (TC) messages and HELLO messages. In ADARA, the signaling overhead includes all its different types of signaling packets.

We compared ADARA, AODV and OLSR based on the packet delivery ratios (PDR), the average end-to-end delay, and the number of signaling packets sent by all routers. PDR indicates the number of packets received by destination routers divided by number of packets sent by the source routers. The average end-to-end delay is the time elapsed from the time a packet is sent by a source until it is received by its destination. For the case of ADARA and AODV, this delay includes the duration packet is buffered waiting for RREPs.

The scenarios used to compare the three routing protocols were chosen to stress all three protocols, rather than to attain good performance for either on-demand or proactive routing.

8.3.2 Effect of Mobility

In this scenario, 50 routers are spread randomly in a $300m \times 1500m$ area, with 25 of the routers generating traffic, each with 15 packets per second. The destination for each flow is a specific router with probability of 0.5 and is chosen randomly from all routers with probability of 0.5. We tried different maximum mobility speeds of 5 m/s to 30 m/s with a zero pause.

Figure 8.3 shows the PDR, average end-to-end delays, and the signaling overhead incurred by the three protocols as a function of router speed.

Higher router speed results in more topology changes. The drastic drop in PDR in all protocols is due to routes breaking due to router mobility and the time needed by the routing protocols to obtain new routes. OLSR requires routers to detect link failures and additions based on the absence or reception of HELLO messages, and TC messages to inform all routers of the topology so that new routes can be established. Given that TC messages are sent periodically listing one or multiple link states, the signaling overhead in OLSR remains fairly constant as a function of router speed. However, this means that more link changes take place between periodic transmissions of TC messages as router speed increases, which results in more data packets being lost as they traverse paths that are broken.

Link failures in AODV and ADARA are detected by the absence of a number of consecutive Hello messages, and a route discovery process is done to inform sources of new routes to destinations. Because of the delays incurred in detecting link failures and in establishing new routes after that, as router speed increases more and more data packets traversing failed routes end up being dropped.

The lower delays obtained with ADARA can be attributed to the aggregation of RREQs, which reduces the number of RREQs being flooded and hence reduces network congestion, as well as the fact that each signaling packet is sent in broadcast mode containing the current sequence number of the transmitting router, which helps routers detect link failures and repair routes more quickly.

The enormous impact of RREQ aggregation in ADARA is evident in Figure 8.3. In OLSR, TC messages must be disseminated by MPRs throughout the network and in AODV, each RREQ is flooded throughout the network. By contrast, a RREQ in ADARA is disseminated throughout the network only when no other RREQ asking for a route for the same destination has been forwarded recently. The size of TC messages in OLSR is much larger than the size of RREQs in AODV, which accounts for the similarity in signaling overhead between the two even though routers in OLSR disseminate fewer signaling packets than in AODV.

Figure 8.4 shows the performance comparison of the three protocols as a function of mobility pause time. For this simulation runs, we considered 50 routers in a area of $300m \times 1500m$ with 25 flows as described previously. Pause times vary from 0 seconds (i.e., constant mobility) to to 900 seconds (i.e., almost static routers). The speed of routers is chosen randomly between zero and 20 meters per second. As it is can be seen from the figure, the packet delivery ratio for ADARA is very close to that of OLSR, while AODV is much lower for all pause times. Average delays in ADARA are much lower than those attained in OLSR and AODV over all pause times, and AODV renders the higher delays in all cases. It is also evident that ADARA incurs far less signaling overhead than OLSR and AODV for all pause times. As should be expected, less signaling overhead is incurred by all protocols as the pause time increases.

8.3.3 Effect of Number of Flows

Figure 8.5 shows the comparison of the three protocols as a function of number of sources in the network. For all cases, sources are different routers. For each flow, one specific router is selected as the destination with probability 0.5 and a random destination is selected with probability 0.5.

The PDR decreases and the average end-to-end delays increase for all three protools as the number of sources increases. These results can be explained from the additional congestion created in the channel as a result of having more data packets when more sources are added. The results for signaling overhead as a function of the number of sources clearly show the benefits of RREQ aggregation in ADARA compared to AODV and OLSR. Although the signaling traffic in ADARA does increase as the number of sources increases, many of those sources share common destinations and this results in many RREQs being aggregated, which in turn results in much smaller overhead than with the other two protocols.

8.3.4 Effect of Network Size

Figure 8.6 shows the performance of the three protocols as a function of the number of routers. We considered different network sizes of 25, 50, 75, and 100 routers spread randomly in a area of $300m \times 1000m$, $300m \times 1500$, $300m \times 2000$, and $500m \times 2200$ with 15, 25, 40 and 50 flows respectively. Similar to the previous scenarios, a destination is a specific router with probability of 0.5 and it chosen randomly with probability of 0.5.

As we have stated, the scenarios were selected to stress all protocols, rather than to show likely operating points. For all three protocols, as the network size increases the PDR drops, end-to-end delays increase, and signaling overhead increases. This is unavoidable, given that OLSR must send more link states, and AODV and ADARA must send more route requests as the network size increases. However, it is clear that ADARA is more scalable than OLSR and AODV, and is far more efficient than AODV.

8.4 Conclusions

We introduced route-request aggregation as an effective mechanism to significantly reduce the signaling overhead incurred in route discovery, and presented ADARA as an example of the basic approach.

ADARA uses destination-based sequence numbers to avoid routing-table loops like AODV does, and uses RREQ aggregation and broadcast signaling packets to reduce signaling overhead. We compared the performance of ADARA, AODV, and OLSR and analyzed the effect of mobility, number of flows, and network size on the performance of the protocols. The simulation results show that, in terms of packet delivery ratio, ADARA performs much better than AODV and performs very close to OLSR in all cases. The signaling overhead incurred with ADARA is much smaller than the overhead in AODV and OLSR. Furthermore, the use of RREQ aggregation and broadcast signaling packets in ADARA leads to fewer packets contenting for the channel and results in lower end-to-end delays for ADARA compared to AODV and OLSR.

As we have stated, the basic approach of using RREQ aggregation can be applied to any on-demand routing protocol. Accordingly, our results offer a great opportunity to improve the performance of on-demand routing protocols being considered for standardization. Our results indicate that RREQ aggregation can make AODV, DSR, or other on-demand routing protocols, far more attractive compared to OLSR and other proactive routing protocols.

Our description of ADARA assumed single-path routing. However, multi-path routing [91, 97] can be easily supported as well. Furthermore, as we we have stated, RREQ aggregation can be used together with other techniques that have been proposed to improve the performance of on-demand routing in MANETs.

The next steps for our work on unicast routing include the definition and analysis of multi-path routing based on ADARA, the use of loop-avoidance techniques other than destination-based sequence numbers in the context of route-request aggregation, the use of geographical coordinates as in [89, 102], and the use of clustering techniques. In addition, it is clear that the use of route-request aggregation can be applied to improve the performance of on-demand multicast routing [80, 85].

Chapter 9

Conclusion and Future Work

In this thesis we introduce new architecture and forwarding strategy which improves the performance of the content centric networking in different aspects:

- **Loop free interest forwarding** Current interest based architecture suffer from undetected looping of interest packets when interest aggregation is used at the same time. Undetected interest loop causes high delays in response time and also results in higher memory consumption of routers for stateful architectures. The proposed forwarding strategy eliminates the chance of undetected interest loops by ensuring that interest is getting closer to the producer at each hop.

- **Smaller Forwarding Tables** We have shown that PIT tables can get very large with millions of entries specially for core internet routers. Considering each interest packet or data packet triggers a lookup and an alteration on this tables, such tables should be implemented in highly advanced and extremely expensive memories such as TCAMs which might be capable of handling large volume of traffic with very large tables. The proposed architecture in this dissertation, removes the need for stateful forwarding and PIT tables by introducing label swapping mechanism and anonymous datagram to forward interest and data packets. This strategy decreases the table size in orders of magnitude which makes it an efficient and feasible architecture for the internet with much lower processing overhead.

- **Native support for efficient Multicasting** The multicasting strategy used in the proposed architecture, with no need for extra signaling messages, improves the delay and storage consumption compared to other ICN architectures.
- **Scalability** The proposed strategy removes the need for large FIB tables listing name prefixes by early binding of names to content provider anchors. This architecture can be deployed at Internet scale and handle billions of name prefixes.

We have shown that interest aggregation can decrease the signaling overhead in networks significantly, when there is no in-network caching. Using this idea, we proposed a new MANET routing protocol which increases the network performance by decreasing the signaling overhead using route request aggregation and broadcast signaling messages. This method can be used in any on-demand routing protocol.

9.1 Future Work

The ideas presented in this dissertation opens new avenues in computer networking specially for Content Centric Networking. We presented the big picture and general ideas of the new architecture for content centric networking, although the current methods can be improved in various ways such as failure handling and forwarding across AS, much more research needs to be done to cover different aspects of it:

- **Congestion Control** Many works have been done to address congestion control in NDN and CCN. Although the proposed works are handling congestion more efficiently compared to TCP, but they all suffer from limitation brought by NDN and CCN architectures such as flow detection and bottle neck problem. We believe that our proposed architecture introduces great opportunities to address the congestion control problems and to introduce new techniques to improve the networking performance.
- **DDoS Mitigation** The anonymity of interests in ICN architectures, makes it difficult to mitigate DDoS attacks. Also stateful forwarding of NDN and CCN makes the network highly vulnerable to interest flood attacks. The PIT-less architecture presented in this dissertation makes the network more tolerable to such attacks. Also using the label swapping technique presented in this architecture, we believe there are more efficient ways to mitigate DDoS attacks and still preserving the interest anonymity.
- **Dynamic Name Based Routing Protocols** There is a good opportunity to design and deploy name based routing protocols which adapt congestion and topology changes. Analyzing the impact of such protocols on the network performance can be very helpful.
- **Apply the techniques to IP world** Ideas presented in this dissertation and in other ICN approaches can be applied to current IP architecture: Route request aggregation to multicast on-demand routing protocols, loop-free forwarding using hop count in-

formation as presented in this dissertation, multipath routing based on ADARA and hybrid routing protocols that deploy route request aggregation.

The research presented in this dissertation has been published in the following proceedings:

- *"Understanding optimal caching and opportunistic caching at the edge of information-centric networks"*, ICN '14
- *"Enabling Correct Interest Forwarding and Retransmissions in a Content Centric Network"*, ANCS '15
- *"A Light-Weight Forwarding Plane for Content-Centric Networks"*, ICNC '16
- *"Content-Centric Networking Using Anonymous Datagrams"*, IFIP Networking '16
- *"Efficient Multicasting in Content-Centric Networks Using Datagrams"*, Globecom '16
- *"Content-Centric Networking at Internet Scale through The Integration of Name Resolution and Routing"*, ICN '16
- *"Making On-Demand Routing Efficient with Route-Request Aggregation"*, MSWIM '16

In addition we have submitted the following journal paper:

- *"Content Centric Networks without Pending Interest Tables"*, Transactions on Networking

Bibliography

[1] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN Simulator for ns-3", *University of California, Los Angeles, Tech. Rep*, 2012.

[2] B. Ahlgren et al., "A Survey of Information-centric Networking," *IEEE Commun. Magazine*, July 2012, pp. 26–36.

[3] J. Behrens and J.J. Garcia-Luna-Aceves, "Hierarchical Routing Using Link Vectors," *Proc. IEEE INFOCOM '98*, April 1998.

[4] Content Centric Networking Project (CCN) [online]. http://www.ccnx.org/releases/latest/doc/technical/

[5] A. Dabirmoghaddam et al., "Understanding Optimal Caching and Opportunistic Caching at "The Edge" of Information-Centric Networks," *Proc. ACM ICN '14*, Sept. 2014.

[6] E.W. Dijkstra and C.S. Scholten "Termination Detection for Diffusing Computations," *Information Processing Letters*, Vol. 11, No. 1, 1980.

[7] E.W. Dijkstra, W. Feijen, and A.J.M. van Gasteren, "Derivation of a Termination

Detection Algorithm for Distributed Computations," *Information Processing Letters*, Vol. 16, No. 5, 1983.

[8] T. Koponen et al., "A Data Oriented (and Beyond) Network Architecture," *Proc. ACM SIGCOMM 07*, 2007.

[9] J.J. Garcia-Luna-Aceves, "A Unified Approach to Loop-Free Routing Using Distance Vectors or Link States," *Proc. ACM SIGCOMM '89*, Aug. 1989.

[10] J.J. Garcia-Luna-Aceves, "Name-Based Content Routing in Information Centric Networks Using Distance Information," *Proc. ACM ICN '14*, Sept. 2014.

[11] J.J. Garcia-Luna-Aceves, "Routing to Multi-Instantiated Destinations: Principles and Applications," *IEEE ICNP '14*, Oct. 2014.

[12] J.J. Garcia-Luna-Aceves, "Efficient Multi-Source Multicasting in Information Centric Networks," *Proc. IEEE CCNC '15*, Jan. 2015.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom '00*, 2000.

[14] V. Jacobson et al., "Networking Named Content," *Proc. IEEE CoNEXT '09*, Dec. 2009.

[15] A.K.M. Mahmudul-Hoque et al., "NSLR: Named-Data Link State Routing Protocol," *Proc. ACM ICN '13*, 2013.

[16] J. Matocha and T. Camp, “A Taxonomy of Distributed Termination Detection Algorithms,” *Journal of Systems and Software*, 1998.

[17] NDN Project [online]. http://www.named-data.net/

[18] Zhang, Lixia, et al. ”Named data networking.” ACM SIGCOMM Computer Communication Review 44.3 (2014): 66-73.

[19] M. Spohn and J.J. Garcia-Luna-Aceves, “Scalable Link-State Internet Routing,” *Proc. IEEE ICNP ‘98*, Oct. 1998.

[20] I. Solis and J.J. Garcia-Luna-Aceves, “Robust Content Dissemination in Disrupted Environments,” *Proc. ACM CHANTS ‘08*, Sept. 2008.

[21] S. Vutukury and J.J. Garcia-Luna-Aceves, “A Simple Approximation to Minimum-Delay Routing,” *Proc. ACM SIGCOMM ‘99*, Aug. 1999.

[22] G. Xylomenos et al., “A Survey of Information-centric Networking Research,” *IEEE Communication Surveys and Tutorials*, July 2013.

[23] C. Yi et al., “Adaptive Forwarding in Named Data Networking,” *ACM CCR*, Vol. 42, No. 3, July 2012.

[24] C. Yi et al., “A Case for Stateful Forwarding Plane,” *Computer Communications*, pp. 779-791, 2013.

[25] W.T. Zaumen and J.J. Garcia-Luna-Aceves, “Dynamics of Distributed Shortest-Path Routing Algorithms,” *Proc. ACM SIGCOMM ‘91*, Sept. 1991.

[26] W.T. Zaumen and J.J. Garcia-Luna-Aceves, "System for Maintaining Multiple Loop-free Paths between Source Node and Destination Node in Computer Network," US Patent 5,881,243, 1999.

[27] L. Zhang et al., "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, Vol. 44, No. 3, July 2014.

[28] Lagutin, Dmitrij, Kari Visala, and Sasu Tarkoma. "Publish/Subscribe for Internet: PSIRP Perspective." *Future internet assembly 84* , 2010.

[29] A. Afanasyev et al., "Interest Flooding Attack and Countermeasures in Named Data Networking," *Proc. IFIP Networking '13*, May 2013.

[30] AT&T, "The Quality of Internet Service: AT&T's Global IP Network Performance Measurements," 2003.
http://ipnetwork.bgtmo.ip.att.net/pws/paper.pdf

[31] A. Carzaniga et al., "Content-Based Publish/Subscribe Networking and Information-Centric Networking," *PRoc. ACM ICN '11*, August 2011.

[32] L. Ciavatone et al., "Standardized Active Measurements on a Tier 1 IP Backbone," *IEEE Comm. Magazine*, June 2003.

[33] H. Dai el al., "On Pending Interest Table in Named Data Networking," *Proc. ACM ANCS '12*, Oct. 2012.

[34] S. Fayazbakhsh et al., "Less Pain, Most of the Gain: Incrementally Deployable ICN," *Proc. ACM SIGCOMM '13*, 2013.

[35] C. Fricker et al., “Impact of traffic mix on caching performance in a content-centric network,” *Proc. IEEE NOMEN Workshop ‘12*, 2012.

[36] J.J. Garcia-Luna-Aceves, “A Fault-Tolerant Forwarding Strategy for Interest-based Information Centric Networks,” *Proc. IFIP Networking 2015*, May 2015.

[37] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, “Enabling Correct Interest Forwarding and Retransmissions in a Content Centric Network,” *Proc. ACM ANCS ‘15*, May 2015.

[38] B.N. Levine et al., “The Case for Reliable Concurrent Multicasting Using Shared ACK Trees,” *Proc. ACM Multimedia ‘96*, Nov. 1996.

[39] G. Markowsky and F.H. Moss, “An Evaluation of Local Path ID Swapping in Computer Networks,” *IEEE Trans. Commun.*, Vol. COM-29, pp. 329-336, March 1981.

[40] W. So et al., “ Toward Fast NDN Software Forwarding Lookup Engine Based on Hash Tables,” *Proc. ACM ANCS ‘12*, 2012.

[41] C. Tsilopoulos et al., “Reducing Forwarding State in Content-Centric Networks with Semi-Stateless Forwarding,” *Proc. IEEE INFOCOM 2014*, April 2014.

[42] M. Varvello et al., “On The Design and Implementation of a Wire-Speed Pending Interest Table,” *Proc. IEEE Infocom NOMEN Workshop*, April 2013.

[43] M. Virgilio et al., “PIT Overload Analysis in Content Centric Networks,” *Proc. ACM ICN ‘13*, Aug. 2013.

[44] S. Vutukury and J.J. Garcia-Luna-Aceves, “A Multipath Framework Architecture for Integrated Services,” *Proc. IEEE Globecom ‘00*, Nov. 2000.

[45] Chandler, S.,“Calculation of Number of Relay Hops Required in Randomly Located Radio Network ” *Electron. Lett.*, 1989.

[46] Marco Zuniga and Bhaskar Krishnamachari,“Analyzing the Transitional Region in Low Power Wireless Links ” *Proc. IEEE SECON*,2004.

[47] D. Faria,“Modeling Signal Attenuation in IEEE 802.11 Wireless LANs - Vol. 1 ” *Stanford University*, 2005.

[48] Varga, Andras,“The omnet Discrete Event Simulation System ” *Proc. ESM*, 2001.

[49] Jim Kurose,“Information-Centric Networking: The Evolution from Circuits to Packets to Content” *Comput. Netw.*, 2014.

[50] Ahlgren, Bengt and Dannewitz, Christian and Imbrenda, Claudio and Kutscher, Dirk and Ohlman, Börje,“A Survey of Information-Centric Networking ” *IEEE Commun. Mag.*, 2012.

[51] Traverso, Stefano and Ahmed, Mohamed and Garetto, Michele and Giaccone, Paolo and Leonardi, Emilio and Niccolini, Saverio,“Temporal Locality in Today’s Content Caching: Why It Matters and How to Model It” *ACM SIGCOMM Comput. Commun. Rev.*, 2013.

[52] Rosensweig, Elisha and Kurose, Jim and Towsley, Don,“Approximate Models for General Cache Networks” *Proc. IEEE INFOCOM*, 2010.

[53] arofiglio, Giovanna and Gallo, Massimo and Muscariello, Luca,“Bandwidth and Storage Sharing Performance in Information Centric Networking” *Proc. ACM SIGCOMM Workshop on ICN*, 2011.

[54] Fricker, Christine and Robert, Philippe and Roberts, James,“A Versatile and Accurate Approximation for LRU Cache Performance” *Proc. ITC*, 2012.

[55] Roberts, James and Sbihi, Nada,“Exploring the Memory-Bandwidth Tradeoff in an Information-Centric Network” *Proc. ITC*, 2013.

[56] Daley, Daryl and Vere-Jones, David,“An Introduction to the Theory of Point Processes” *I: Elementary Theory and Methods*, 2003.

[57] Hawkes, Alan,“Spectra of Some Self-Exciting and Mutually Exciting Point Processes” *Biometrika*, 1971.

[58] Breslau, Lee and Cao, Pei and Fan, Li and Phillips, Graham and Shenker, Scott,“Web Caching and Zipf-like Distributions: Evidence and Implications” *Proc. IEEE INFOCOM*, 1999.

[59] Gill, Phillipa and Arlitt, Martin and Li, Zongpeng and Mahanti, Anirban,“Youtube Traffic Characterization: A View from the Edge” *Proc. ACM IMC*, 2007.

[60] Dan, Asit and Towsley, Don,“An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes” *ACM SIGMETRICS Perform. Eval. Rev.*, 1990.

[61] Jelenković, Predrag and Radovanović, Ana,“Least-Recently-Used Caching with Dependent Requests” *Theor. Comput. Sci.*, 2004.

[62] Che, Hao and Tung, Ye and Wang, Zhijun,“Hierarchical Web Caching Systems: Modeling, Design and Experimental Results” *IEEE J. Sel. Areas Commun.*, 2002.

[63] Rodriguez, Pablo and Spanner, Christian and Biersack, Ernst,“Analysis of Web Caching Architectures: Hierarchical and Distributed Caching” *IEEE/ACM Trans. Netw.*, 2001.

[64] Valentina Martina and Michele Garetto and Emilio Leonardi,“A Unified Approach to the Performance Analysis of Caching Systems” *Proc. IEEE INFOCOM*, 2014.

[65] Psaras, Ioannis and Clegg, Richard and Landa, Raul and Chai, Wei and Pavlou, George,“Modelling and Evaluation of CCN-Caching Trees” *IFIP Networking*, 2011.

[66] Psaras, Ioannis and Chai, Wei Koong and Pavlou, George,“Probabilistic In-Network Caching for Information-Centric Networks” *Proc. ACM SIGCOMM Workshop on ICN*, 2012.

[67] Massimo Gallo and Bruno Kauffmann and Luca Muscariello and Alain Simonian and Christian Tanguy,“Performance Evaluation of the Random Replacement Policy for Networks of Caches” *Perform. Evaluation*, 2014.

[68] Melazzi, N and Bianchi, G and Caponi, A and Detti, A,“A General, Tractable and Accurate Model for a Cascade of LRU Caches” *IEEE Commun. Lett.*, 2014.

[69] Chai, Wei and He, Diliang and Psaras, Ioannis and Pavlou, George,“Cache “Less for More” in Information-Centric Networks (Extended Version)” *Comput. Commun.*, 2013.

[70] Yonggong Wang and Zhenyu Li and Tyson, G. and Uhlig, S. and Gaogang Xie,"Optimal Cache Allocation for Content-Centric Networking" *Proc. IEEE ICNP*, 2013.

[71] Ghodsi, Ali and Shenker, Scott and Koponen, Teemu and Singla, Ankit and Raghavan, Barath and Wilcox, James,"Information-Centric Networking: Seeing the Forest for the Trees" *Proc. ACM HotNets*, 2011.

[72] Rosensweig, Elisha and Menasche, Daniel and Kurose, Jim,"On the Steady-State of Cache Networks" *Proc. IEEE INFOCOM*, 2013.

[73] Tyson, Gareth and Kaune, Sebastian and Miles, Simon and El-khatib, Yehia and Mauthe, Andreas and Taweel, Adel,"A Trace-Driven Analysis of Caching in Content-Centric Networks" *Proc. ICCCN*, 2012.

[74] Danzig, Peter and Hall, Richard and Schwartz, Michael,"A Case for Caching File Objects Inside Internetworks" *SIGCOMM Comput. Commun. Rev.*, 1993.

[75] Rossi, Dario and Rossini, Giuseppe,"Caching Performance of Content Centric Networks Under Multi-Path Routing (and More)" *Telecom ParisTech*, 2011.

[76] Fonseca, R. and Almeida, V. and Crovella, M. and Abrahao, B.,"On the Intrinsic Locality Properties of Web Reference Streams" *Proc. IEEE INFOCOM*, 2003.

[77] Cherkasova, Ludmila and Gupta, Minaxi,"Analysis of Enterprise Media Server Workloads: Access Patterns, Locality, Content Evolution, and Rates of Change" *IEEE/ACM Trans. Netw.*, 2004.

[78] Gasti, P., Tsudik, G., Uzun, E., Zhang, L.,"Dos and ddos in named data networking"

I: International Conference on Computer Communications and Networks (ICCCN), 2013.

[79] M. Abolhasan et al., “A Review of Routing Protocols for Mobile Ad Hoc Networks,” *Ad hoc networks*, 2004.

[80] A. Boukersche, *Handbook of Algorithms for Wireless and Mobile Computing*, Chapman and Hall, 2006.

[81] J. Broch et al., “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols.” *Proc. ACM/IEEE MobiCom ‘98*, 1998.

[82] T. Clausen and P. Jacquet, “Optimized link state routing protocol (OLSR)” *RFC 3626*, 2003.

[83] T. Clausen, “Comparative Study of Routing Protocols for Mobile Ad-hoc networks,” Research Report RR-5135, INRIA, 2004.

[84] S. Dabideen and J.J. Garcia-Luna-Aceves, “Ordering in Time: A New Routing Approach for Wireless Networks,” Proc. IEEE MASS 2010, San Francisco, CA, November 8-12, 2010.

[85] J.J. Garcia-Luna-Aceves and R. Menchaca-Mendez, “PRIME: An Interest-Driven Approach to Integrated Unicast and Multicast Routing in MANETs,” *IEEE/ACM Trans. on Networking*, March 2011.

[86] J.J. Garcia-Luna-Aceves and S. Roy, “On-Demand Routing in Ad Hoc Networks Using Link Vectors,” *IEEE JSAC*, Vol. 23, No. 3, March 2005.

[87] H. Jiang and J.J. Garcia-Luna-Aceves, “Performance Comparison of Three Routing Protocols for Ad Hoc Networks,” *Proc. IEEE IC3N ‘01*,Oct. 2001.

[88] J. Jubin and J. Tornow, “The DARPA Packet Radio Network Protocols,” *Proc. of The IEEE*, january 1987.

[89] Y. Ko and N.Vaidya, “Location-Aided Routing (LAR) in Mobile Ad Hoc Networks,” *Wireless Networks*, Vol. 6, No. 4, 2000.

[90] P. Mohapatra et al., *Ad Hoc Networks: Technologies and Protocols*, Springer, 2005.

[91] M. Mosko and J.J. Garcia-Luna-Aceves, “Multipath Routing in Wireless Mesh Networks,” *Proc. IEEE WiMesh ‘05*, 2005.

[92] S. Murthy and J.J. Garcia-Luna-Aceves, “An Efficient Routing Protocol for Wireless Networks.” *Mobile Networks and Applications*, 1996.

[93] G. Pei et al., ”A Wireless Hierarchical Routing Protocol with Group Mobility,” *Proc. IEEE WCNC ‘99*, 1999.

[94] C. Perkins et al., *Ad Hoc Networkiing*, Addison-Wesley, 2008.

[95] H. Rangarajan and J.J. Garcia-Luna-Aceves, “On-demand Loop-Free Routing in Ad hoc Networks Using Source Sequence Numbers,” *Proc. IEEE MASS ‘05*, Nov. 2005.

[96] J. Raju and J.J. Garcia-Luna-Aceves, “A Comparison of On-Demand and Table Driven Routing for Ad-Hoc Wireless Networks,” *Proc. IEEE ICC ‘00*, June 2000.

[97] J. Raju and J.J. Garcia-Luna-Aceves, “A New Approach to On-Demand Loop-Free Multipath Routing,” *Proc. IEEE IC3N ‘99*, Oct. 1999.

[98] S. Roy and J.J. Garcia-Luna-Aceves, "Node-Centric Hybrid Routing for Ad Hoc Networks," *Proc. 10th IEEE/ACM MASCOTS '02*, 2002.

[99] E. Royer and C.K. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, 1999.

[100] C. Shiflet, E. M. Belding-Royer, and C.E. Perkins. "Address Aggregation in Mobile Ad Hoc Networks," *Proc. IEEE ICC '04*, 2004.

[101] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *Proc. ACM Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999.

[102] Y. Wang, C. Westphal, and J.J. Garcia-Luna-Aceves, "Using Geographical Coordinates To Attain Efficient Route Signaling in Ad Hoc Networks," *Proc. IEEE WoWMoM '13*, June 2013.

[103] X. Wu et al., "A Unified Analysis of Routing Protocols in MANETs," *IEEE Trans. on Communications*, March 2010.

[104] IEEE Computer Society. IEEE 802.11 and 802.11n Standards, 1999.

[105] J. Chen et al., "COPSS: An Efficient Content Oriented Publish/Subscribe System," *Proc. ACM/IEEE ANCS 2011*, Oct. 2011.

[106] S. Deering et al., "The PIM architecture for wide-area multicast routing," *IEEE/ACM Trans. on Networking*, Vol. 4, No. 2, April 1996.

[107] J.J. Garcia-Luna-Aceves, “New Directions in Content Centric Networking,” *Proc. IEEE CCN‘15*, Oct. 19, 2015.

[108] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, “A Light-Weight Forwarding Plane for Content Centric Networks,” *Proc. IEEE ICNC ‘16*, Feb. 2016.

[109] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, “Content-Centric Networking Using Anonymous Datagrams,” *Proc. IFIP Networking 2016*, May 2016.

[110] M. Parsa and J.J. Garcia-Luna-Aceves, “A Protocol for Scalable Loop-free Multicast Routing,” *IEEE JSAC*, April 1997.

[111] J. Raju et al., “System and Method for Information Object Routing in Computer Networks,” U.S. Patent 7,552,233, June 23, 2009

[112] M. Wahlisch et al., “Lessons from the Past: Why Data-driven States Harm Future Information-Centric Networking,” *IFIP Networking ‘13*, May 2013.

[113] M. Wahlisch et al., “Backscatter from The Data Plane Threats to Stability and Security in Information-Centric Network Infrastructure,” *Computer Networks*, Vol. 57, No. 16, Nov. 2013.

[114] A. Afanasyev et al., “Interest-flooding Attack and Countermeasures in Named Data Networking,” *Proc. IFIP Networking ‘13*, May 2013.

[115] A. Afanasyev et al., “SNAMP: Secure Namespace Mapping to Scale NDN Forwarding,” *in Proc. IEEE Global Internet Symposium ‘15*, 2015.

[116] B. Ahlgren et al., “A Survey of Information-Centric Networking,” *IEEE Commun. Magazine*, July 2012, pp. 26–36.

[117] P. Baran, “On Distributed Communications: I. Introduction to Distributed Communication Networks,” Memorandum RM-3420-PR, The RAND Corporation, Aug. 1964.

[118] M.F. Bari et al., “A Survey of Naming and Routing in Information-Centric Networks,” *IEEE Commun. Magazine*, July 2012, pp. 44–53.

[119] R.G. Gallager, “A Minimum delay Routing Algorithm Using Distributed Information,” *IEEE Trans. Commun.*, 1977.

[120] J.J. Garcia-Luna-Aceves, “Eliminating Undetected Interest Looping in Content Centric Networks,” *Proc. IEEE NOF ‘15*, Sept. 30-Oct. 2, 2015.

[121] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, “Efficient Multicasting in Content-Centric Networks Using Datagrams,” *IEEE Globecom ‘16*, Washington, D.C., 4–8 Dec. 2016.

[122] M. Gritter and D. Cheriton, “An Architecture for Content Routing Support in The Internet,” *Proc. USENIX Symposium on Internet Technologies and Systems*, Sept. 2001.

[123] O. Heckmann et al., “On Realistic Network Topologies for Simulation,” *Proc. ACM SIGCOMM MoMeTools ‘03*, Aug. 2003.

[124] E. Hemmati and J.J. Garcia-Luna-Aceves, “A New Approach to Name-Based Link-State Routing for Information-Centric Networks,” *Proc. ACM ICN ‘15*, Oct. 2015.

[125] V. Lehman et al., "A Secure Link State Routing Protocol for NDN," Technical Report NDN-0037, Jan. 2016.

[126] S. Murthy and J.J. Garcia-Luna-Aceves, "Congestion-Oriented Shortest Multipath Routing," *Proc. IEEE INFOCOM '96*, 1996.

[127] M. Papalini et al., "Scalable Routing for Tag-Based Information Centric Networking," *Proc. ACM ICN '14*, Sept. 2014.

[128] D. Perino and M. Varvello, "A Reality Check for Content Centric Networking," *ACM ICN '11*, 2011.

[129] "www-dsg.stanford.edu/triad/"

[130] T.C. Schmidt et al., "Let's Collect Names: How PANINI Limits FIB Tables in Name Based Routing," *Proc. IFIP Networking '16*, May 2016.

[131] W. So et al., "Named Data Networking on A Router: Fast and DoS-Resistant Forwarding with Hash Tables," *Proc. ACM ANCS '13*, 2013.

[132] T. Song et al., "Scalable Name-Based Packet Forwarding: From Millions to Billions," *Proc. ACM ICN 2015*, Sept. 2015.

[133] M. Thorup and U. Zwick, "Compact Routing Schemes," *Proc. ACM Symp. Parallel Algorithms and Architectures*, 2001.

[134] M. Varvello et al., "Caesar: A Content Router for High Speed Forwarding," *Proc. ACM ICN '12*, 2012.

[135] Y. Wang et al., “Scalable Name Lookup in NDN Using Effective Name Component Encoding,” *Proc. ICDCS ‘12*, 2012.

[136] Y. Wang et al., “Wire Speed Name Lookup: A GPU-based Approach,” *USENIX NSDI ‘13*, 2013.

[137] G. Xylomenos et al., “A Survey of Information-centric Networking Research,” *IEEE Communication Surveys and Tutorials*, July 2013.

[138] H. Yuan and P. Crowley, “Reliably Scalable Name Prefix Lookup,” *proc. IEEE ICCCN ‘12*, 2012.

[139] Garcia-Luna-Aceves, J. J., Maziar Mirzazad-Barijough, and Ehsan Hemmati. ”Content-Centric Networking at Internet Scale through The Integration of Name Resolution and Routing.” *proc. ICN 16*, 2016.

[140] Mirzazad-Barijough, Maziar, and J. J. Garcia-Luna-Aceves. ”Making On-Demand Routing Efficient with Route-Request Aggregation.” *proc. MSWIM 16*, 2016.