A More Scalable Approach to Content-Centric Networking

J.J. Garcia-Luna-Aceves*†
*Palo Alto Research Center, Palo Alto, CA 94304
†Department of Computer Engineering, University of California, Santa Cruz, CA 95064
Email: jj@soe.ucsc.edu

***Abstract*—On-demand Content Exchange with Adaptive Naming (OCEAN) is introduced as an alternative to content-centric networking approaches (NDN and CCN) that require the maintenance of forwarding state of each Interest traversing a router in a Pending Interest Table (PIT). Content routers in OCEAN maintain data answer routing tables (DART) to support the correct forwarding of Interests and responses to such Interests (content or negative acknowledgments) without divulging the identities of consumers who originate Interests. The size of a DART is proportional to the number of routes used by Interests traversing a router, rather than the number of Interests traversing a router. It is shown that undetected Interest loops cannot occur in OCEAN, while the same is not true for CCN and NDN, and that Interests and responses to Interests are forwarded correctly in the absence of failures. OCEAN attains similar latencies than NDN and CCN but incurs orders of magnitude less storage overhead.**

I. Introduction

Several information-centric networking (ICN) architectures have been proposed as an alternative to today's Internet [2]. A leading ICN approach can be characterized as *Interest-based*, and consists of: populating forward information bases (FIB) maintained by routers with routes to name prefixes denoting content, sending content requests (called Interests) for specific named data objects (NDO) over paths implied by the FIBs, and delivering content along the reverse paths traversed by Interests. The original content-centric networking (CCN) proposal [9] was the first example of an Interest-based ICN architecture in which Interests need not be flooded and do not state the identity of the sender. Today, named data networking (NDN) [13] and CCNx [5] are the leading approaches on Interest-based ICN.

Since the introduction of CCN and NDN [9], [13], the research community has assumed four major premises for Interest-based content-centric networking: (a) Stating the name of requested content or the content name and a nonce is needed to detect Interest looping; (b) forwarding state for each Interest traversing a router must be maintained in its Pending Interest Table (PIT) to allow Interests and responses to such Interests (content or negative acknowledgments) to be forwarded without divulging the sources of Interests; (c) Interests from different consumers requesting the same content need to be aggregated to attain efficiency; and (d) NDOs need to be sent over the reverse paths created by Interests.

This paper shows that maintaining PITs is not necessary–and indeed is undesirable–in order to attain correct and efficient forwarding of Interests and content in a content-centric network. Section II addresses the performance impact of Interest aggregation on the efficiency of NDN and CCN in a practical deployment, and the problem of undetected Interest looping in NDN and CCN.

Section III introduces **OCEAN** (*On-demand Content Exchange with Adaptive Naming*). Instead of a PIT, a content router maintains a data answer routing table (DART), which allows routers to return data objects to the correct neighbors who requested them and without requiring Interests to state their origins. OCEAN attains this using a data structure inspired by some of the earliest work on virtual circuit networking with local identifiers [11], [14]. An Interest in OCEAN states the name of the requested content, a hop count, a destination-and-return token (*dart*), and a nonce. The hop count is used for Interest loop detection. The dart is used by forwarding routers to leave a trace of the path traversed by the Interest using local identifiers of the previous hop and the current hop, so that a named data object (NDO) or a negative acknowledgment (NACK) can be sent back to the content requestor, without the producer or caching site knowing the source of the Interest. The nonces in Interests are used by a content producer or caching site to associate multiple paths traversed by Interests from the same consumer, without knowing the identity of the consumer.

Section IV proves that Interest loops cannot occur and be undetected if OCEAN is used, and that that Interests, NDO messages and NACKs traverse the correct paths in the absence of failures. Section V compares the complexity of NDN and CCN with the complexity of OCEAN, which has a storage complexity that is orders of magnitude smaller than that of NDN and CCN.

II. Interest Aggregation in NDN and CCN

A. Elements of NDN and CCN Operation

In NDN and CCN a given router uses three primary data structures: a forwarding information base (FIB), a pending interest table (PIT), and a content store (CS). The forwarding strategy determines the interaction among these tables needed to forward Interests towards nodes advertising having copies of requested content, send named data objects (NDO) back to consumers who requested them over reverse paths traversed

by Interests, and send any other signal indicating the inability to satisfy an Interest.

A router uses its FIB to route Interests towards the desired content producer advertising a content prefix name. A FIB is populated using content routing protocols or static routes. The FIB entry for a given name prefix lists the interfaces that can be used to reach the prefix. A CS is a cache for content objects. With on-path caching, routers cache the content they receive in response to Interests they forward.

A PIT is used in NDN and CCN to keep track of the neighbor(s) to which NDO messages or NACKs should be sent back in response to Interests, allows Interests to not disclose their sources, and enables Interest aggregation. A PIT entry consists of a vector of one or multiple tuples, one for each nonce processed for the same NDO name. Each tuple states the nonce used, the incoming interfaces and the outgoing interfaces. Each PIT entry has a lifetime, which should be larger than the estimated round-trip time to a site where the requested NDO can be found.

When a router receives an Interest, it checks whether there is a match for the content requested in the Interest in its CS. The Interest matching mechanisms differ in NDN [13] and CCNx [5], with the latter supporting exact Interest matching only. If a match to the Interest is found, the router sends back an NDO over the reverse path traversed by the Interest. If no match is found in the CS, the router determines whether the PIT stores an entry for the same content. In NDN, if the Interest states a nonce that differs from those stored in the PIT entry for the requested content, then the router "aggregates" the Interest by adding the incoming interface from which the Interest was received and the nonce to the PIT entry without forwarding the Interest. On the other hand, if the same nonce in the Interest is already listed in the PIT entry for the requested content, the router sends a NACK over the reverse path traversed by the Interest. In CCNx, aggregation is done if the Interest is received from an interface that is not listed in the PIT entry for the requested content, and a repeated Interest received from the same interface is simply dropped.

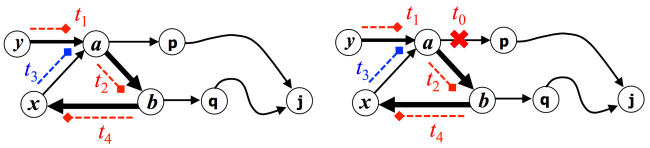
If a router does not find a match in its CS and PIT, the router forwards the Interest along a route (or routes) listed in its FIB for the best prefix match.

B. Interest Aggregation and Undetected Loops

Interest aggregation can provide performance benefits as long as there is a high likelihood that many Interests arrive at a router asking for the same content that has not yet been cached locally. However, given the latencies expected in the Internet today [3], [4], this is not likely. Because of on-path caching and round-trip latencies between a requestor and a router with cached content being a few hundred milliseconds at most, the percentage of Interests that can benefit from aggregation is negligible, especially if on-path caches have large capacities.

On the other hand, Interest propagation is based on FIB entries, which need not correspond to loop-free paths due to topology changes or the ranking of interfaces (neighbors) with respect to content prefixes made at individual routers independently of others.

Interest aggregation has a negative side effect when Interests are aggregated while traversing loops induced by the FIB entries. We have shown that attempting to detect Interest loops using content names or content names and nonces, which is the approach in NDN and CCN, does not always work when Interests are aggregated [7], [8]. Furthermore, we have also shown that no forwarding strategy can be defined that always works correctly, allows Interest aggregation, and detects loops by identifying Interests uniquely using names, names and nonces, or other information.



(a) Long-term loop (b) Temporary loop

Fig. 1. Interest looping in NDN or CCN

Figure 1 illustrates undetected Interest looping in NDN and the original CCN proposal. 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 an interface is better than interfaces shown with thinner lines. Dashed lines indicate the traversal of Interests over paths. The time when an event arrives at a router is indicated by t_i.

Figure 1(a) shows the case of a long-term Interest loop caused by multi-paths implied in FIBs not being loop-free, even though all routing tables are consistent. In this case, the ranking of interfaces in a FIB can be such that a path with a larger hop count may be ranked higher than a path with a smaller hop count, because of the perceived performance of the interfaces or paths towards prefixes.

Figure 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. In both cases, router a aggregates the Interest from x and router x aggregates the Interest from b, and the combined steps preclude the detection of any Interest looping. In this example, it would appear that the looping problems could be avoided by forcing router b to use q rather than x for Interests regarding prefixes announced by j. However, the same looping problems would exist even if link (b, q) were removed in the example, and the ways in which FIBs are populated and interfaces are ranked are independent of updates made to PITs.

III. OCEAN

A. Design Rationale

The design rationale for OCEAN is based on four observations. First, Interest aggregation is unlikely to have a significant impact in an actual deployment at Internet scale. Content caching and the relatively small round-trip latencies make it highly unlikely that many Interests requesting the same content will arrive at a router when the content has not been

cached due to a prior request. On the other hand, Interest aggregation in NDN and CCN interacts negatively with loops resulting from FIB inconsistencies resulting from topology changes or local rankings of interfaces.

Second, the number of routers in a network is orders of magnitude smaller than the number of NDOs 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, a correct Interest forwarding strategy cannot be based on attempting to identify each Interest uniquely, but the information in the FIBs can be used to establish an ordering of the routers that forward Interests.

Lastly, there is no inherent reason to require reverse-path forwarding to be used to forward NDOs or NACKs sent in response to Interests.

For simplicity of exposition, we make a number of assumptions in the description of OCEAN. We assume that Interests are retransmitted only by the consumers that originated them, rather than routers that relay Interests, and that routers forward Interests on a best-effort basis. Given that no prior work shows that any Interest matching policy is better than simple exact matching of Interests, we assume that routers use exact Interest matching. We assume that a request for content from a local content consumer is sent to the router in the form of an Interest stating an infinite hop count to content and an empty dart and nonce. Lastly, a router is assumed to know which interfaces are neighbor routers and which are local consumers.

B. Information Exchanged and Stored

The information used to enable correct forwarding of Interests, NDO messages, and NACKs are the *destination-and-return tokens* (**darts**), and Interest nonces. Darts are local identifiers used to uniquely denote routes established between source and destination routers over which Interests, NDO messages and NACKs are sent. Nonces are global identifiers that can be used to associate two or more routes established between the same source and destination of Interests.

The name of NDO j is denoted by $n(j)$, and the terms neighbor and interface are used interchangeably. An Interest forwarded by node k requesting NDO $n(j)$ is denoted by $I[n(j), h^I(k), ID^I(k), dart^I(k)]$, and states the name of the requested NDO ($n(j)$), the hop count ($h^I(k)$) from node k to the nearest instance of the name prefix $n(j)^*$ that is the best match for $n(j)$, a nonce ($ID^I(k)$) created by the origin of the Interest, and the dart ($dart^I(k)$) used to establish anonymous routes back to the sources of Interests.

An NDO message sent in response to an Interest is denoted by $D[n(j), sig(j), ID^I(i), dart^I(i)]$, and states the name of the Interest ($n(j)$), a signature payload ($sig(j)$) used optionally to validate the content object, the nonce ($ID^I(i)$) and the dart ($dart^I(i)$) to be used to forward the message, and the NDO itself.

The NACK sent in response to an Interest is denoted by $NI[n(j), \mathsf{CODE}, ID^I(i), dart^I(i)]$ and states the name of the NDO ($n(j)$), the nonce ($ID^I(i)$) and the dart ($dart^I(i)$) to be used to forward the NACK, and a code (CODE) indicating the reason why the NACK is sent. Possible reasons for sending a NACK include: an Interest loop is detected, no route is found towards requested content, no content is found, and the DART entry expired.

Router i maintains five tables: an optional content store (CS^i), a FIB (FIB^i), a data-answer routing table ($DART^i$), an origin nonce table (ONT^i), and a destination nonce table (DNT^i). CS^i is the same as in NDN and CCN, lists the NDOs stored in local caches, and is indexed on the names of NDOs. The other tables are different.

A *predecessor* for Interests regarding $n(j)^*$ is a router that forwarded an Interest to router i regarding NDO $n(j)$, which matches name prefix $n(j)^*$. A *successor* for Interests related to $n(j)^*$ is a router to whom router i forwards an Interest regarding $n(j)$, which matches name prefix $n(j)^*$. An *anchor of a prefix* is a router that has advertised the prefix.

FIB^i is indexed using content name prefixes. The entry for prefix $n(j)^*$ consists of a set of tuples, one for each next hop in the set of successors of router i for $n(j)^*$ ($S^i_{n(j)^*}$). The tuple corresponding to neighbor $q \in S^i_{n(j)^*}$ states:

1) $h(i, n(j)^*, q)$: The hop count to $n(j)^*$ through q.
2) $a(i, n(j)^*, q)$: The nearest anchor through q for $n(j)^*$.

The minimum hop count from i to $n(j)^*$ through any neighbor listed in FIB^i is denoted by $h(i, n(j)^*)$.

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

1) $a^i(a, p)$: The anchor a for which the forwarding state is established at router i.
2) $p^i(a, p)$: The predecessor p of route to a.
3) $pd^i(a, p)$: The *predecessor dart*, which equals the dart received in Interests from p routed towards anchor a.
4) $s^i(a, p)$: The name of router s, selected by router i to forward Interests received from p towards anchor a.
5) $sd^i(a, p)$: The *successor dart* included in Interests sent towards anchor a through successor s.
6) $h^i(a, p)$: The number of hops to anchor a through successor s when the dart entry was established.
7) $LT^i(a, p)$: The lifetime for the entry.

The lifetime of a DART entry is decremented while the router stores it and the entry is deleted when the lifetime reaches the 0 value. An entry in a DART can remain in storage for long periods of time in the absence of topology changes, and the removal of a DART entry causes only a minor slow down of some Interests and the most likely case in a stable network is for the replacement of the DART entry to state the same information as the entry that was erased.

ONT^i stores the mappings between the names of local consumers and the nonces assigned to them by router i. The entry for nonce n^i contains the local identifier of a local consumer c and is denoted by $ONT^i(n^i)$.

DNT^i is indexed using the nonces received in Interests.

The entry for a nonce $ID^I(k)$ received in an Interest from router k is denoted by $DNT^i(ID^I(k))$ and contains a list of one or more tuples, each stating the name of a router that sent an Interest containing the same nonce $ID^I(k)$, and the dart stated in that Interest.

C. Interest Loop Prevention and Detection

A salient feature of OCEAN is that Interest loops resulting from inconsistencies in FIB entries maintained at different routers are avoided or detected if they occur. The following rule is used to ensure that router i accepts an Interest from neighbor k only if it is closer to $n(j)^*$ through at least one next hop than k was when it sent its Interest.

Interest Forwarding Rule (IFR):
Router i accepts $I[n(j), h^I(k), ID^I(k), dart^I(k)]$ from router k if the following condition is satisfied:

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

Figures 2(a) and (b) illustrate how OCEAN prevents Interests from traversing loops when a multi-path routing protocol is used to populate the FIBs and FIB entries state next hops to prefixes that lead to forwarding loops. The pair of numbers next to a node in Figure 2(a) indicate the hop count from that node to $n(j)$ over an interface and the ranking of the interface according to the FIB of the node. Let (v, h, r) denote the triplet indicating an interface, its hop count and its ranking.

In Figure 2(a), FIB^a states $(b, 4, 1)$, $(p, 4, 2)$, and $(x, 6, 3)$; 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 2(b) shows, router a receives $I[n(j), h^I(y) = 5, ID^I, dart^I(y)]$ from router y at time t_1. Router a forwards $I[n(j), h^I(a) = 4, ID^I, dart^I(a)]$ to b, because $5 = h^I(y) > h(a, n(j)^*, b) = 4$ and b is ranked above p. 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 must use neighbor q as the next hop for the Interest, because q is the highest ranked neighbor satisfying IFR. Similarly, the Interest generated by router x is forwarded to router q towards j without traversing a loop, because each relaying router must satisfy IFR.

Figures 2(c) to (e) illustrate how OCEAN operates when single-path routing is used and a temporary routing-table loop exists in the FIBs. Each router has a single next hop and hop count for each prefix in its FIB. The distance from a router to name prefix $n(j)^*$ need not be directly proportional to the hop counts of the paths. For example, link (b, q) may have limited bandwidth or long delays and hence b prefers the path through x to reach $n(j)^*$. Router b updates its FIB at time time t_0 as shown in Figure 2(c), and routers have inconsistent FIB states for $n(j)$ while Interests are being forwarded. As shown in Figure 2(d), router b *must* send $NI[n(j), \text{loop}, ID^I, dart^I(b)]$ to a, because $4 = h^I(a) \not> h(b, n(j)^*, x) = 6$. In turn, a forwards a NACK to y. The Interest from x also prompts a NACK from b because IFR is not satisfied. Within a finite time, FIB^a, FIB^x, and FIB^b are updated to reflect the new topology state, and Interests from y regarding objects in $n(j)^*$ can be forwarded along the chain of nodes a, b, and q towards $n(j)^*$. Similarly, within a finite time, Interests from x regarding $n(j)^*$ can be forwarded to b and q towards $n(j)^*$. Section IV proves that no Interest loops can go undetected in OCEAN.

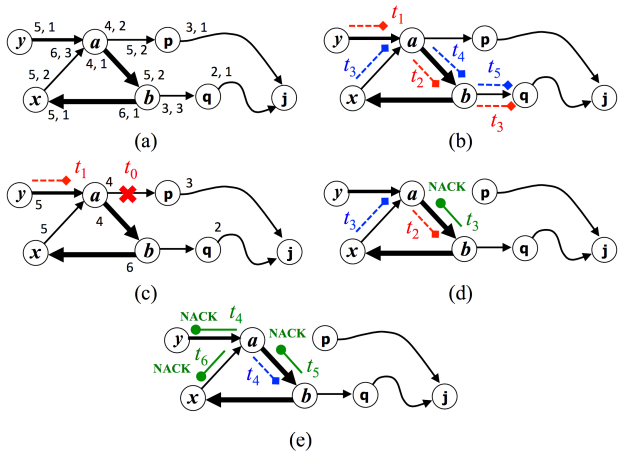


Fig. 2. OCEAN ensures Interest-loop detection

D. Maintaining Forwarding State

Routers in OCEAN maintain routes to those sites advertising name prefixes (anchors), and populate their routing tables using a routing protocol operating in the control plane (e.g., [6], [10]). Routers populate their FIBs with routes to anchors based on the data stored in their routing tables.

In NDN and CCN, the route taken by the response to an Interest is the reverse path traversed by that Interest, which is incrementally stored in the PITs of the routers along that path. In OCEAN, DARTs and ONTs replace the PITs, and DNTs enable the forwarding of an NDO message or NACK over a path that need not be the reverse of the path traversed by the Interest.

Routers create nonces to unambiguously associate local consumers with the Interests sent on their behalf, and without revealing their identities. The dart mappings stored in DARTs are equivalent to the label mappings first introduced for packet switching based on virtual circuits [11], [14]. They are used to allow multiple Interests asking for NDOs associated with the same name prefix to be multiplexed in the same route segments established between the routers originating the Interests and an anchor or a router that can respond to the Interests using cached content. The nonce mappings stored in ONTs allow a router receiving a response to an Interest to give it to the correct consumer.

The combined use of DNTs with DARTs and ONTs enables a generalization of the label swapping approach that allows NDO messages or NACKs to be sent over paths that need not be the reverse paths traversed by the Interests they answer.

Algorithms 1 to 5 specify the steps taken by routers to process and forward Interests, and return NDO messages or NACKs. In our description, it is assumed that router i has populated ONT^i with all the nonces locally assigned to local consumers, and that DART entries are silently deleted when their lifetimes expire. For convenience, it is assumed that an

anchor of a name prefix stores all the NDOs associated with the prefix in its CS.

In addition, it is assumed 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. Accordingly, if router i detects that connectivity to neighbor k is lost, it deletes all entries in $DART^i$ for which k is the predecessor or the successor of a path towards any anchor.

Algorithm 1 (Interest_Source) 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 an NDO and the name of the consumer, together with an an empty hop count to content and an empty dart. Router i uses the highest ranked neighbor router to forward an Interest (Line 9 of Algorithm 1). If a DART entry exists for the selected successor of the Interest, the corresponding successor dart is used; otherwise, a new successor dart is created and a new DART entry is stored before the Interest is forwarded.

Algorithm 1 Processing Interest from consumer c at i

1: **function** Interest_Source
2: **INPUT:** $CS^i, FIB^i, DART^i, ONT^i, I[n(j), nil, c, nil]$;
3: **if** $n(j) \in CS^i$ **then** send $D[n(j), nil, c, nil]$;
4: **if** $n(j) \notin CS^i \wedge n(j)^* \in CS^i$ **then** send $NI[n(j)$, no content, $nil, c, nil]$;
5: **if** $n(j) \notin CS^i \wedge n(j)^* \notin CS^i$ **then**
6: **if** $n(j)^* \notin FIB^i$ **then**
7: % No route exists to $n(j)^*$:
send $NI[n(j)$, no route, $nil, c, nil]$
8: **else**
9: **for each** $v \in S^i_{n(j)^*}$ **by rank do**
10: % Interest can be forwarded to v:
$a = a(i, n(j)^*, v)$;
11: **if** $\exists DART^i(a, i) \mid s^i(a, i) = v$ **then**
12: % Interest can be forwarded using entry $DART^i(a, i)$:
$ID^I(i) = ONT^i(c)$;
$h^I(i) = h^i(a, i)$; $dart^I(i) = sd^i(a, i)$;
send $I[n(j), h^I(i), ID^I(i), dart^I(i)]$ to $s^i(a, i)$
13: **else**
14: create entry $DART^i(a, i)$:
$a^i(a, i) = a$; compute $SD \neq sd^i(p, q) \; \forall \; DART^i(p, q)$;
$p^i(a, i) = i$; $pd^i(a, i) = SD$;
$s^i(a, i) = v$; $sd^i(a, i) = SD$;
$h^i(a, i) = h(i, n(j)^*, v)$; $LT^i(a, i) = MLT$;
15: create and send Interest:
$ID^I(i) = ONT^i(c)$;
$h^I(i) = h^i(a, i)$; $dart^I(i) = sd^i(a, i)$;
send $I[n(j), h^I(i), ID^I(i), dart^I(i)]$ to $s^i(a, i)$
16: **end if**
17: **end for**
18: **end if**
19: **end if**
20: **end function**

Algorithm 2 (DART_Mapping) shows the steps taken by router i to process an Interest received from a neighbor router k. If the requested content is cached locally, an NDO message is sent back (Line 3 of Algorithm 2). If the content does not exist, a NACK is sent back (Line 4 of Algorithm 2). If the content is remote and a DART entry already exists for the dart stated in the Interest from k (Line 8 of Algorithm 2), then IFR has been satisfied by a previous Interest on the same route to an anchor of the prefix and the existing mapping can be used.

Alternatively, if the Interest must be forwarded and no DART entry exists for the dart stated in the Interest from k, router i must find a successor for the Interest and must create a DART entry. Router i sends a NACK if no entry can be found in FIB^i for $n(j)^*$ (Line 11 of Algorithm 2) or IFR is not satisfied (Line 21 of Algorithm 2). Otherwise, router i selects the highest-ranked neighbor router v that satisfies IFR and creates a DART entry mapping the dart received from k to a new dart created for a route to the selected anchor through v (Lines 14 to 20 of Algorithm 2).

Algorithm 2 describes a simple forwarding strategy in which router i selects the first neighbor v found in the ranked list of interfaces stored in FIB^i for prefix $n(j)^*$, such that v offers a path that has a hop count towards the requested content that is strictly smaller than the hop count stated in the Interest being forwarded.

Algorithm 2 Processing Interest from router k at i

1: **function** DART_Mapping
2: **INPUT:**
$CS^i, FIB^i, DART^i, I[n(j), h^I(k), ID^I(k), dart^I(k)]$;
3: **if** $n(j) \in CS^i$ **then call** Response$(D[n(j), sig(j), ID^I(k), dart^I(k)]$;
4: **if** $n(j) \notin CS^i \wedge n(j)^* \in CS^i$ **then**
5: CODE = no content;
call Response$(NI[n(j)$, CODE, $h^I(k), ID^I(k), dart^I(k)])$;
6: **end if**
7: **if** $n(j) \notin CS^i \wedge n(j)^* \notin CS^i$ **then**
8: **if** $\exists DART^i(a, k) \mid pd^i(a, k) = dart^I(k)$ **then**
9: % Interest can be forwarded using entry $DART^i(a, k)$:
$h^I(i) = h^i(a, k)$; $ID^I(i) = ID^I(k)$; $dart^I(i) = sd^i(a, k)$;
send $I[n(j), h^I(i), ID^I(i), dart^I(i)]$ to $s^i(a, k)$
10: **else**
11: **if** $n(j)^* \notin FIB^i$ **then**
12: % No route exists to $n(j)^*$:
CODE = no route;
call Response$(NI[n(j)$, CODE, $ID^I(k), dart^I(k)])$
13: **else**
14: **for each** $v \in S^i_{n(j)^*}$ **by rank do**
15: **if** $h^I(k) > h(i, n(j)^*, v)$ **then**
16: % Interest can be forwarded to v:
$a = a(i, n(j)^*, v)$;
compute $SD \mid \forall p \forall q \; (SD \neq sd^i(p, q) \in DART^i)$;
17: create entry $DART^i(a, k)$:
$h^i(a, k) = h(i, n(j)^*, v)$; $LT^i(a, k) = MLT$;
$a^i(a, k) = a$; $p^i(a, k) = k$; $pd^i(a, k) = dart^I(k)$;
$s^i(a, k) = v$; $sd^i(a, k) = SD$;
18: create and forward Interest:
$h^I(i) = h^i(a, k)$;
$ID^I(i) = ID^I(k)$; $dart^I(i) = sd^i(a, k)$;
send $I[n(j), h^I(i), ID^I(i), dart^I(i)]$ to v;
return
19: **end if**
20: **end for**
21: % Interest may be traversing a loop:
CODE = loop; **call** Response$(NI[n(j)$, CODE, $dart^I(k)])$
22: **end if**
23: **end if**
24: **end if**
25: **end function**

The entry in $DART^i$ for the anchor-predecessor pair (a, k) establishes a mapping from the dart used by router k in Interests that can be resolved by anchor a (predecessor dart) to the dart used by router i in Interests that can be resolved by anchor a and that it forwards to router v (successor dart). The predecessor-successor mappings stored in the DARTs of routers from router i to an anchor a denotes route segments that can be used to forward Interests towards a unambiguously. Conversely, the sequence of successor-predecessor mappings stored along the same path can be used to forward responses (NDO messages or NACKs) to the origins of Interests unambiguously.

If reverse-path forwarding of NDO messages and NACKs is adopted, routers that relay an Interest and the router that responds to the Interest can send the NDO message or NACK back using the dart mappings stored in their DARTs. The router that originated the Interest on behalf of a local consumer uses the nonce included in the response to forward it to the

correct consumer.

However, Interests can be forwarded to the same anchor over multiple paths, with each such path identified with a different set of dart mappings. The router that responds to Interests can store in its NDT the mappings between the nonces included in Interests and the routes over which the Interests were received (identified by the neighbor routers and dart received in the Interests). Given that nonces are assigned with a low probability of collision, an NDO message or NACK may be sent by the responding router over a path different than the one traversed by the Interest.

Algorithm 3 Responding to a Remote Interest at i

1: **function** Response
2: **INPUT:** $CS^i, FIB^i, DART^i, DNT^i$, $(D[n(j), sig(j), ID^I(k), dart^I(k)] \vee NI[n(j), \mathsf{CODE}, ID^I(k), dart^I(k)])$;
3: $dart^i_k = dart^I(k)$;
add tuple $[k, dart^i_k]$ to the list of tuples for entry $DNT^i(ID^i(k))$
4: select tuple $[n, dart^i_n] \in DNT^i(ID^i(k))$;
5: **if** response $= D[n(j), sig(j), ID^I(k), dart^I(k)]$ **then**
6: send $D[n(j), sig(j), ID^I(k), dart^I(n)]$ to neighbor n
7: **end if**
8: **if** response $= NI[n(j), \mathsf{CODE}, ID^I(k), dart^I(k)]$ **then**
9: send $NI[n(j), \mathsf{CODE}, ID^I(k), dart^I(n)]$ to neighbor n
10: **end if**
11: **end function**

Algorithm 2 calls Algorithm 3 (Response) to send back an NDO message or a NACK in response to a remote Interest. For simplicity, only those routers that respond to Interests from remote consumers store information in their DNTs. Algorithm 3 does not provide a specific approach for the selection of a neighbor out of many, but makes clear the use of nonces and darts for the forwarding of responses to the correct consumer over one of many paths traversed by different prior Interests between the same routers.

Algorithm 4 Processing NDO message at i

1: **function** NDO_Handling
2: **INPUT:** $DART^i, CS^i, D[n(j), sig(j), ID^I(q), dart^I(q)]$;
3: **[o]** verify $sig(j)$;
4: **[o] if** verification fails **then** discard $D[n(j), sig(j), ID^I(q), dart^I(q)]$
5: **if** $\exists DART^i(a,k) \mid sd^i(a,k) = dart^I(q)$ **then**
6: % NDO message can be forwarded to predecessor k:
$dart^I(i) = pd^i(a,k)$; $ID^I(i) = ID^I(q)$;
send $D[n(j), sig(j), ID^I(i), dart^I(i)]$ to k;
[o] store the content with name $n(j)$ in CS^i
7: **end if**
8: **end function**

Algorithm 5 Processing NACK at i

1: **function** NACK_Handling
2: **INPUT:** $DART^i, NI[n(j), \mathsf{CODE}, ID^I(q), dart^I(q)]$;
3: **if** $\exists DART^i(a,k) \mid sd^i(a,k) = dart^I(q)$ **then**
4: % NACK can be forwarded to predecessor k:
$dart^I(i) = pd^i(a,k)$; $ID^I(i) = ID^I(q)$;
send $NI[n(j), \mathsf{CODE}, ID^I(i), dart^I(i)]$ to k
5: **end if**
6: **end function**

Algorithm 4 (NDO_Handling) outlines the processing of NDO messages. A router accepts an NDO message received from a neighbor only if it has a DART entry with a successor dart matching the dart stated in the NDO message (Line 5 in Algorithm 4). A router stores a data object it receives optionally (Step 6 of Algorithm 4).

Algorithm 5 (NACK_Handling) states the steps taken to handle NACKs. Router i forwards the NACK it receives for $n(j)$ only if it has a DART entry with a successor dart matching the dart stated in the NACK (Line 3 in Algorithm 5).

E. OCEAN Forwarding Example

Figure 3 illustrates how darts and nonces are used to label Interests and associate Interests with NDO messages and NACKs. In the example, routers a and x have local consumers originating the Interests, and those Interests are assumed to request NDOs regarding name prefixes advertised by anchor d. The arrowheads in the links of the figure denote the next hops stored in the FIBs of routers, $y(i)$ denotes the ith dart in $DART^y$, and ID^y_c denotes a nonce created by router y to associate Interests submitted from a local consumer c. The figure shows DART entries of routers for anchor d, ONT entries at routers a and x, and DNT entries at router d.

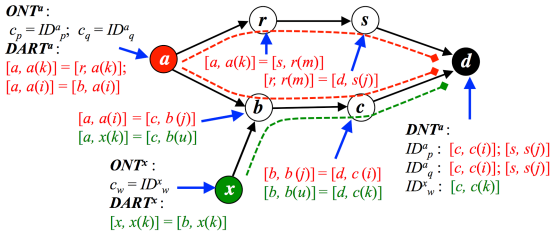


Fig. 3. Forwarding using darts and nonces.

Router a uses nonces $ID^a_p(d)$ and or $ID^a_q(d)$ to label the Interests it originates on behalf of local consumer c_p and c_q, respectively. By the same token, router x uses nonce $ID^x_w(d)$ to label all Interests it originates on behalf of local consumer c_w. Accordingly, ONT^a stores the mappings $c_p \leftrightarrow ID^a_p$, and $c_q \leftrightarrow ID^a_q$; and ONT^x stores the mapping $c_w \leftrightarrow ID^x_w$.

As Figure 3 illustrates, routers a, b, and c establish the following mappings in their DARTs: $[a; a(i)] \leftrightarrow [b; a(i)]$ at a, $[a; a(i)] \leftrightarrow [c; b(j)]$ at b, $[b; b(j)] \leftrightarrow [d; c(i)]$ at c, and $[c; c(i)]$ at d. These mappings denote the routes (a, b, c, d), (b, c, d), and (c, d) uniquely. Similarly, routers establish the DART mappings shown in the figure that denote the routes (a, r, s, d) and (x, b, c, d), as well as subpaths to d in them.

All the Interests originated by consumers c_p, c_q, and c_x regarding content advertised by anchor d can be routed towards d using the same few darts shown in Figure 3. Given that an NDO or NACK specifies the successor dart and the nonce stated the Interest it answers, NDO messages and NACKs can be forwarded correctly from d (or a router along the way to d caching content) to router a or router x unambiguously. Furthermore, routers a and x can use the entries in ONT^a and ONT^x and the nonces in the replies they receive to send them to the correct local consumers.

The above illustrates that OCEAN supports correct Interest forwarding and correct reverse-path forwarding of NDO messages and NACKs using DARTs and ONTs. However, the forwarding of NDO messages and NACKs need not be limited to reverse-path forwarding. As Figure 3 shows, when router d receives Interests from neighbors s and c, it stores the mappings between nonces and the successor darts in those Interests in DNT^d, namely: $ID^a_p(d) \leftrightarrow \{[c, c(i)], [s, s(j)]\}$, $ID^a_q(d) \leftrightarrow \{[c, c(i)], [s, s(j)]\}$ and $ID^x_w(d) \leftrightarrow \{[c, c(k)]\}$. Accordingly, router d can send an NDO message or NACK to the origin of an Interest stating nonce $ID^a_p(d)$ or $ID^a_q(d)$

through neighbor s or c, without having to use the reverse path over which the Interest was received.

IV. Correctness of OCEAN

The following theorems show that OCEAN is correct. Theorem 4.1 shows that OCEAN prevents Interests from being propagated along loops without meeting routers that detect the incorrect forwarding and send NACKs in return. The result of the theorem is independent of whether the topology is static or dynamic or the FIBs are consistent or not.

To discuss the correctness of Interest forwarding in OCEAN, we say that an Interest loop of h hops for an NDO with name $n(j)$ occurs when one or more Interests 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 $v_0 = v_h$.

Theorem 4.1: Interest loops cannot occur and be undetected in a network in which OCEAN is used.

Proof: Consider a network in which OCEAN 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 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 $I[n(j), h^I(v_k), ID^I(v_k), dart^I(v_k)]$ to node $v_{k+1} \in L$ for $1 \leq k \leq h-1$, and $v_h \in L$ must send $I[n(j),\ h^I(v_h),\ ID^I(v_h),\ dart^I(v_h)]$ 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 $I[n(j), h^I(v_k), ID^I(v_k), dart^I(v_k)]$ 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 node v_h sends $I[n(j),\ h^I(v_h),\ dart^I(v_h)]$ 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 and no Interest aggregation occurs in OCEAN, each node in L 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 $I[n(j), h^I(v_{k-1}), ID^I(v_{k-1}), dart^I(v_{k-1})]$ before $RT(PI^{v_k}_{n(j)})$ expires for $1 \leq k < h$, and $v_1 \in L$ must accept $I[n(j), h^I(v_h),\ ID^I(v_h),\ dart^I(v_h)]$ before $RT(PI^{v_1}_{n(j)})$ expires.

According to OCEAN, if v_k sends $I[n(j), h^I(v_k),\ ID^I(v_k), dart^I(v_k)]$ to v_{k+1} as a result of receiving $I[n(j), h^I(v_{k-1}),\ ID^I(v_{k-1}),\ dart^I(v_{k-1})]$ 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 $I[n(j), h^I(v_1),\ ID^I(v_1),\ dart^I(v_1)]$ to v_2 as a result of receiving $I[n(j), h^I(v_1),\ dart^I(v_1)]$ 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 and be undetected when each node in the loop uses IFR 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. ■

Theorem 2 addresses the ability for routers to demultiplex NDO messages and NACKs correctly and send the responses to the correct consumers using only the information stored in their DARTs, ONTs, and DNTs, whether or not Interest loops occur. The theorem assumes that all transmissions are sent correctly.

Theorem 4.2: OCEAN ensures that, in the absence of failures, NDO messages and NACKs are sent correctly to the consumers who submitted the corresponding Interests.

Proof: From the operation of OCEAN, a router can forward an NDO or NACK back towards the source of an Interest only if it receives the Interest correctly and stores forwarding state in its DART. Hence the proof can assume that all the routers from the source of an Interest (s) to a router answering the Interest (d) have established some forwarding state in their DARTs, and have established the necessary state in their ONTs.

It follows from Algorithm 1 that the result is true if a router can resolve an Interest from a local consumer ($s = d$). The rest of the proof must show that each router from d to s can demultiplex correctly the NDO messages and NACKs that traverse paths established by Interests delivered from s to d.

Let h be the number of hops in the path traversed by an NDO or NACK in response to an Interest originated at router s and answered by router d or another router on the path to d, and let f_i denote the router at the ith hop from d to s.

Basis Case: Let $h = 1$. In this case $s = f_1$. Router s labels its Interest with a dart assigned uniquely for its one-hop route to d and a nonce assigned uniquely to the local consumer that created the content request. From Algorithms 2 and 3, router d responds to the Interest from s directly to s and includes the dart and nonce in its response. At s, the dart in the response is associated with s as the origin of the route, and the nonce is associated uniquely with a local consumer. It follows that the basis case is true.

Inductive Step: By assumption, when a router f_j in a path from s to d receives the first Interest from neighbor f_{j+1} requesting content advertised by d and containing dart $dart^I(f_{j+1}) = dt^{j+1}_j(d)$, it creates an entry in $DART^j$ with the mapping $[f_{j+1}; dt^{j+1}_j(d))] \leftrightarrow [f_{j-1}; dt^j_{j-1}(d)]$, and with d as the anchor. The successor dart $dt^j_{j-1}(d)$ is a locally-unique identifier that f_j uses only for Interests received from f_{j+1} with the predecessor dart $dt^{j+1}_j(d)$ and forwarded to the same next hop f_{j-1} towards anchor d. When router f_j receives an NDO message or NACK from f_{j-1} with a dart equal to $dt^j_{j-1}(d)$, it obtains from $DART^{f_j}$ the mapping $[f_{j+1}; dt^{j+1}_j(d))] \leftrightarrow [f_{j-1}; dt^j_{j-1}(d)]$ (Algorithm 2) and forwards the NDO message or NACK to f_{j-1} with a dart equal to $dt^j_{j-1}(d)$ (Algorithms 4 and 5).

The Interest from s either traverses a loop or a simple path towards d or a router that can respond to the Interest when $h > 1$. If the Interest traverses a simple path, it must reach d or a router with the NDO in its CS. If the Interest traverses a loop, it follows from Theorem 1 that a router must issue a NACK with CODE = loop.

Assume that each router up to $k-1$ hops away from d or a router that responds with an NDO message or a NACK for the Interest from s receives such a response correctly. We need to show hat the result is true for $h = k$. From the argument above, when router f_{k-1} forwards an NDO message or NACK originated by d for an Interest created by s to router f_k, its message must state the dart that f_k used in Interests originated at s and directed towards d. Accordingly, it must be true that f_k is able to forward the NDO message or NACK to router f_{k+1} by accessing $DART^{f_{k+1}}$.

If follows that an NDO message or NACK traverses correctly the path of length k hops from either d or a router with the requested content or detecting a loop back to s. Furthermore, the nonce stated in the NDO message or NACK is the one stated in the Interest sent by s on behalf of a local consumer (Algorithms 2 to 5). Because s associates each nonce it creates uniquely with a local consumer, s can forward the response to the correct local consumer. Therefore, the theorem is true. ■

V. Performance Implications

The performance of OCEAN is much the same as that of NDN and CCN regarding the optimality of content delivery, as long as Interest do not traverse loops. OCEAN has a small advantage over NDN and CCN in the presence of loops in FIBs. NDN and CCN must rely on Interest lifetimes expiring before Interests can be retransmitted if they traverse loops that go undetected. By contrast, routers in OCEAN forward Interests over paths that avoid the loops or send NACKs back to consumers within one round-trip time, which is far shorter than an Interest lifetime of seconds. However, the big advantage of OCEAN over NDN and CCN is the small storage overhead it incurs with DARTs compared to the PITs needed in NDN and CCN.

Assume that a network has N routers, all of which can serve as anchors (advertise name prefixes), that each router has D neighbors, and that there are C NDOs that are being requested. NDN and CCN require a router to store the list of pending Interests. The number of PIT entries is $O(C)$. Each PIT entry stores the name of an NDO (and a nonce in the case of NDN), and the identifiers of the neighbors from which the Interest is received and the neighbor to which the Interest is forwarded, which is order $O(D)$. Hence, the storage complexity of NDN and CCN is $S_{NDN} = O(CD)$.

By contrast, in OCEAN, the number of entries in a DART is the number of routes that can traverse a router from any neighbor to any other neighbor towards any anchor. This means $O(D^2N)$ DART entries. Independently of the number of neighbors a router has, each DART entry consists of three router identifiers (anchor, predecessor and successor), two local identifies (processor and successor darts), a hop count and a lifetime. Hence the storage complexity of OCEAN is $S_{OCEAN} = O(ND^2)$.

Given that $N << C$ and that $D < N$ or $D << N$, OCEAN results in enormous storage savings compared to NDN.

VI. Conclusions

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

OCEAN can enjoy the same content security features of CCN and NDN, because it makes no modifications to the way in which content is protected or a name can be securely linked to the payload of an NDO.

We proved that Interests cannot traverse loops in a network running OCEAN without some content router detecting the loop and sending a NACK, and that NDO messages and NACKs are forwarded over the correct paths back to consumers. The delays and signing overhead in OCEAN and NDN are similar, but the storage complexity of OCEAN is orders of magnitude smaller than that of NDN.

The design of multi-path forwarding schemes for routers to establish multi-paths (e.g., [6], [12], [15]) to anchors and for routers to send responses over such multi-paths is an area that deserves further study.

References

[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] 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

[4] L. Ciavatone, A. Morton and G. Ramachandran, "Standardized Active Measurements on a Tier 1 IP Backbone," *IEEE Comm. Magazine*, June 2003.

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

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

[7] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "Enabling Correct Interest Forwarding and Retransmissions in a Content Centric Network," *Proc. ACM/IEEE ANCS '15*, May 7–8, 2015.

[8] J.J. Garcia-Luna-Aceves, "A Fault-Tolerant Forwarding Strategy for Interest-based Information Centric Networks," *Proc. IFIP Networking 2015*, May 20–22, 2015.

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

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

[11] 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.

[12] M. Mosko and J.J. Garcia-Luna-Aceves, "Multipath Routing in Wireless Mesh Networks," *Proc. IEEE WiMesh '05*, Sept. 2005.

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

[14] J. Rindle, "TYMNET 1: An Alternative to Packet Technology," *Proc. 3rd IEEE Int. Conf. Comput. Commun.*, Aug. 1976

[15] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Simple Approximation to Minimum-Delay Routing," *Proc. ACM SIGCOMM '99*, Aug. 1999.