SCARI: A Strategic Caching and Reservation Protocol for ICN

Susmit Shannigrahi
Colorado State University
susmit@cs.colostate.edu

Chengyu Fan
Colorado State University
chengyu.fan@colostate.edu

Christos Papadopoulos
Colorado State University
christos@colostate.edu

ABSTRACT

In the last decade, the Internet service model has shifted from sharing resources to distributing content. Certain applications such as large science data, CDNs, VoIP and multimedia applications transfer a significant amount of data over the Internet in a time-constrained manner that requires guaranteed in-network resources. The networking community often achieves this by creating point-to-point guaranteed bandwidth paths. However, the current resource reservation solutions are end-to-end and often initiated by users without the knowledge of the underlying network conditions. As a result, the data flowing through these reserved tunnels are not reusable, and in-network resources are not optimally utilized.

On the contrary, Information-Centric Networking architectures such as NDN[16] has several properties that can facilitate resource reservations more intelligently. In this work, we present **S**trategic **C**aching **A**nd **R**eservation in **I**CN (SCARI), a protocol for reserving resources on ICN networks. Similar to RSVP[17] in IP networks, SCARI acts as a signaling mechanism before the actual data transfer. In this work, we focus explicitly on scientific dataflows and not on other types of traffic such as real-time audio/video. We show that SCARI reduces network resource consumption by aggregating reservations and strategically caching content in the network.

CCS CONCEPTS

• **Networks** → **Network protocols**; **Network simulations**; **Network resources allocation**;

ACM Reference Format:
Susmit Shannigrahi, Chengyu Fan, and Christos Papadopoulos. 2018. SCARI: A Strategic Caching and Reservation Protocol for ICN. In *AINTEC '18: ASIAN INTERNET ENGINEERING CONFERENCE (AINTEC '18), November 12–14, 2018, Bangkok, Thailand.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3289166.3289167

* The first two authors contributed equally to this work.

AINTEC '18, November 12–14, 2018, Bangkok, Thailand

ACM ISBN 978-1-4503-6131-6/18/11…$15.00
https://doi.org/10.1145/3289166.3289167

1 INTRODUCTION

Despite the Internet's huge success, the underlying Internet Protocol's (IP) best effort network service model does not match the requirements of several special classes of applications. Scientific applications, streaming video applications and CDNs require various guarantees from the network such as dedicated bandwidth, a limited amount of jitter or packet loss cannot rely on IP's best-effort service model can not easily provide. For example, scientific communities routinely transfer data ranging from multiple Terabytes to Petabytes [13]. Using TCP/IP for such large data transfers can dramatically reduce transfer performance [2] leading to missed transfer deadlines, wasted resources, and possible retries from the clients. Due to the ever-changing nature of the large networks and contention for resources, large-science communities, mobile networks and VoIP applications often use bandwidth reservation to ensure lossless data transfers and seamless performance.

In IP networks, end-to-end channels with reserved bandwidth are usually created using protocols built on top of RSVP[17]. However, the current model of resource reservation can be inefficient. First, reservations are often made by the users in an ad-hoc manner. A user trying to reserve bandwidth requires to know the data source and the destination, request a reservation, and transfer data within the reservation's validity period. There are several inherent problems associated with this approach. The users need to make sure the chosen source and the destination are optimal, need to know the operational details of the network and its capacity. From the network's point-of-view, the whole affair can be highly inefficient; if a user creates a reservation but only uses a small portion of the available bandwidth, rest of the reserved bandwidth is wasted. Content from end-to-end flows are not reusable even if multiple reservations share the same underlying path and retrieves the same content.

In this work, we present a new protocol for **S**trategic **C**aching **A**nd **R**eservation in **I**CN (SCARI). In a fundamental shift from current reservation protocols, our protocol places the burden of creating, maintaining, and optimizing reservations on the network, not the end user. Similar to RSVP [17], SCARI is a signaling mechanism that sets up in-network states for upcoming data transfers. SCARI operates on name-prefixes and takes into consideration ICN's in-network caching capabilities. Using Named Data Networking (NDN) for our prototype implementation, we show how SCARI enables hop-by-hop reservation of in-network resources.

To the best of our knowledge, this is the first work that explores resource reservation in the context of ICN. The motivation for our work came from deadline based large scientific

data transfers[6] [7] [3] and as a result, the scope of this particular study is confined to large scientific data transfers over single-domain science networks such as Internet2 or ESNet. For now, we do not consider inter-domain reservations which can introduce a large number of operational challenges such as traffic engineering policies, peering, and economic incentives [9]. We should clarify that we do not propose alternate QoS mechanisms for NDN networks equivalent to IntServ or DiffServ in IP. Instead, much like RSVP, SCARI is intended to act as the building block which such future services will utilize. We should also clarify that we only consider large scientific data transfers for this work. Other types of traffic such as VoIP and real-time video streaming have slightly different reservation requirements, and we plan to investigate them in a separate work.

Rest of this article is organized as follows: we first discuss the motivation and related work. We then introduce SCARI and discuss the design details. In the following sections, we discuss our simulation setup using ndnSIM[5] and evaluate it using a real data access log. Finally, we present our conclusions and future work.

2 MOTIVATION & BACKGROUND

Proliferation of sophisticated scientific instruments, observational and simulation capabilities continue to increase the volume of valuable science data. The datasets are so big, e.g., the Large Hadron Collider (LHC) alone generates approximately 50PB data per year [1], no central facility can provide storage and computational capacities required to support these communities. Scientific communities, therefore, must transfer raw or derivative datasets to local or institutional computation and storage facilities within a deadline. These data transfers are currently accomplished by manually scheduled transfers, orchestrated high-speed paths, and ad-hoc end-to-end bandwidth reservations that often lead to congestion and sub-optimal resource utilization.

Our work draws inspiration from the RSVP protocol[17] in IP that provides receiver-initiated reservations. Since NDN natively supports multicast, in-network intelligent forwarding strategy, and caching, we feel that an efficient reservation protocol is easier to implement with NDN. NDN based protocols can support multicast in a way that is much more difficult in an RSVP scheme. Caching data strategically in an NDN network reduces hop counts for the clients and improve network utilization. Arguments exist that scientific data in the network might be unique and therefore caching them in the network might not be very useful. Nevertheless, previous work has discussed in details [13] that locality of access for scientific data is pretty common, especially towards the edge. In these scenarios, multicast and in-network caching of data can be useful. Besides, as we will describe later, our protocol does not cache everything but only the popular content.

In addition, two crucial differences exist between our reservation protocol and RSVP: (a) Our reservation is per name prefix. Per-prefix reservation means transfers can share the reservation as long as they share some of the network paths, and (b) we argue that the network, and not the user, should be in charge of reservations. While the end-user (or the applications) uses a reservation, it is not in charge of creating or maintaining it. As we will discuss later, the network, specifically the End-node reservation managers(ERMs) decides if a reservation is needed, establishes and maintains it. The second point is an important distinction. In IP network, the end client determines what resources it needs and requests a reservation. In our protocol, the end client expresses the requirements to the network that then decides if a reservation is necessary.

Several previous works have built solutions on top of RSVP that can reserve bandwidth for end users or clients. Such works range from Energy efficient bandwidth reservation [8] to layer 3 protocols such as ESNet's OSCARS[4], from cooperative bandwidth scheduling [11] to time-shifted advanced bandwidth reservations[10]. However, all these protocols are either theoretical in nature since they are hard to implement and deploy in IP networks, or very complex since they need to implement end-user authentication, point-to-point reservation, and data transfer scheduling. As we show in later sections, it can be much simpler to create and support reservations in NDN networks and at the same time, reduce network load, and more optimally use the available resources.

3 PROTOCOL DESIGN

In this section, we discuss the protocol details of SCARI. Two crucial differences exist between SCARI and RSVP: (a) Our reservation is per name prefix. Per-prefix reservation means transfers can share reservations as long as they share some of the network paths, and (b) we argue that the network, and not the user, should be in charge of reservations. While the end-user (or the applications) uses a reservation, it is not in charge of creating or maintaining it, simplifying the applications.

Figure. 1 provides a high-level overview of our protocol. In our design, each router has a reservation manager (RM). In this work, we use two types of reservation managers; (a) reservation managers located on end nodes (ERM), and (b) reservation managers located on router nodes (RRM). The RRMs track available resources, reserve resources for future reservations, aggregate reservations if they are temporally close, and strategically cache content if the requests are not temporally close (what qualifies as "temporally close" depends on the local policy). ERMs perform all services that RRMs perform and also act as a liaison between the network and the applications. In addition to reserving resources, it translates client requests to reservation Interests that it then forwards upstream. For example, an ERM can translate a request from the client indicating <data size, start time, deadline >into a <data size, requested bandwidth, start time, deadline >tuple understood by RRMs. Besides, the ERMs can act as policy modules enforcing quotas and interact with the clients asking them to resubmit requests when requests fail.

Table 1: Reservation Table in SCARI

Req Num	Prefix	Face	Data Size	Avail. Bandwidth	Resv. Request	Avail. Cache	Used. Cache	Start Time (s)	Deadline (s)
1	/xrootd/data1	31	1GB	10Gbps	1Gbps	1TB	0TB	1	10
2	/xrootd/data1	32	1GB	10Gbps	1Gbps	1TB	0TB	2	100
3	/xrootd/data3	54	1GB	10Gbps	1Gbps	1TB	0TB	1	10
4	/xrootd/data3	59	1GB	10Gbps	1Gbps	1TB	0TB	10	20

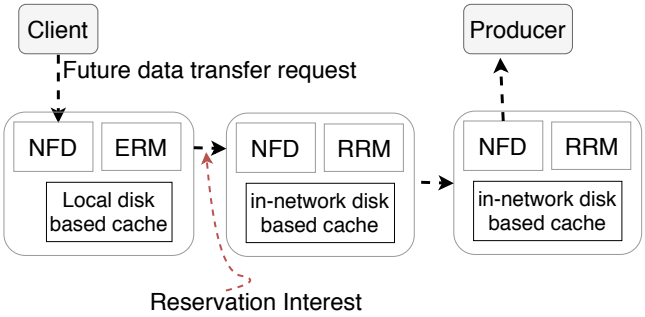

Figure 1: Reservation Communication Overview.

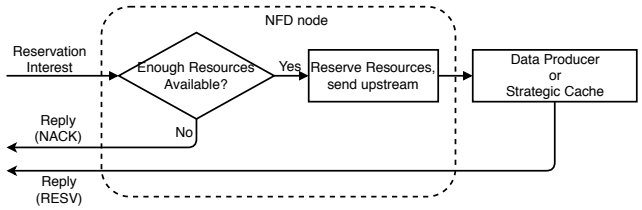

Figure 2: Reservation Protocol Details.

In our protocol, an ERM needs to know how much bandwidth the client would need to reserve. For scientific data transfers, the clients can usually query a catalog that holds this information and find out the sizes of the datasets it wants to retrieve. Once the client sends the data size and deadline information to the ERM [14], the ERM (and not the clients) can estimate how much bandwidth it needs to reserve for this particular request. However, note that resource reservation in NDN is one of the many ways ERM can try to satisfy the data request - for example, it can get data from a local cache, try multi-source retrieval, and other strategies.

For a more detailed description of intelligent retrieval strategies, we encourage the reader to look at our previous works[13][14]. For more general purpose requests such as video streaming, the client and the ERM must know the estimated required bandwidth. The specifics of this estimation will depend on the particular application, and we do not try to address it in this paper.

In SCARI, the Client expresses its requirements to the network through an ERM. Our protocol does not define these requirements, and these can be specific requirements for bandwidth, delay, cache space, or something else. Appropriate strategies are required to interpret the requests and reserve these resources. For this study, we use bandwidth and in-network storage as the reservable resources. We also assume an NDN only network for our study and do not consider dual-stack routers handling both NDN and IP traffic. We should make it clear SCARI should be used only when guaranteed resources are required.

Fig. 2 shows a high-level overview of the reservation protocol inside an RRM. To set up a reservation an NDN node sends a special reservation Interest that is forwarded hop-by-hop upstream. If enough resources are available, a router reserves these resources, and forward the Interest upstream. If enough resources are not available, the network returns a NACK along with the reason for failure. Depending on the network and local policies, there might be several reasons why a router may refuse to reserve resources. For example, a router can refuse reservation if a request exceeds allotted quota, resources are not available, or other higher priority requests must be satisfied first. On receiving the NACK, the downstream node explores different paths, if possible. If the downstream node cannot find enough resources, it sends the NACK further downstream. In the worst case, the NACK travels all the way to the client node, and the application, together with the ERM decides what to do next. The ERM might try other strategies, combine multiple paths, or work with the client to reduce the amounts of requested resources.

3.1 The Reservation Table

For our implementation, all RMs maintain a reservation table at the forwarding strategy layer. This table's functionality is similar to PIT, but instead of keeping forwarding information, it kepdf states about reservations. The table contains the prefix, the size of the reserved strategic cache, the reserved bandwidth, available bandwidth, and the start time and the deadline for the reservation. The reservations are based on name prefixes so that they can utilize NDN's request deduplication by caching and PIT and its name based forwarding. The table is extensible and can hold any number of parameters dictated by various use cases.

3.2 Reservation Prefix Granularity

The granularity of reservation is very important for our study and should be carefully considered. If a client makes a reservation under a top-level prefix, all other requests under the prefix will share that reservation. For example, a reservation for "/xrootd" would lead to millions of datasets under this name prefix to share in-network resources. On the other

hand, creating individual file level reservations will lead to an inflated reservation table. However, both of these scenarios we described can be perfectly fine depending on specific use cases. For example, if a large volume of scientific data is being replicated between two LHC site, as it often happens in practice[12], a reservation for an all-encompassing namespace (e.g., /xtootd) might be desirable.

Besides, reservation granularity can create interesting problems. Consider the following problem: two namespace level reservations with different resource requirements. The first one requires fewer resources (e.g., bandwidth and storage for $/xrootd/file_{1..n}$) and the second one needs more resources(e.g., bandwidth and storage for $/xrootd/file_{1..m}$), where $m > n$. Both reservations are successful until the intersection node where they are combined and forwarded upstream. Now, if the upstream node rejects the request because it can only support "x" requests, where $n < x < m$, both reservations are potentially canceled. Unfortunately, this action unjustifiably punishes the first requester that could be supported but was declined because it was combined with a larger request. While the granularity of reservation will vary depending on the use-case, we use file-level reservations for this study, i.e., the clients request reservations for each file they want to retrieve.

3.3 Reservation Interest

The NDN reservation request is an Interest that carries a tuple in following format **<data name, data size, requested bandwidth, start_time, deadline >**. In our implementation we send a reservation Interest using a special namespace, **</namespace/reservation/>**; so a reservation Interest might look like **</xrootd/ reservation/</xrootd/data1/data_size/start_time/ deadline/bandwidth>>** for requesting reservations for content under /xrootd.

On receiving the reservation Interest the forwarding strategy checks if the request can be merged with another existing request. If it can be aggregated, the strategy sends an ACK indicating this. For example, if a strategy sees two overlapping requests, such as request 1 and 2 in Table. 1, it combines them since the same data flow can satisfy both. In this scenario, the data retrieval starts at $start\_time_1$, takes approximately a little less than 1 second to complete. This data is then cached until 100 Second. Request 2 starts retrieval at $start\_time_5$ and retrieves data from the cache instead of the data source. Once the data retrieval is completed, the node may clean up the cache space, either lazily or immediately depending on its configuration.

Our protocol uses three types of messages to communicate with the client. For our protocol, the strategy sends out a "Duplicate Reservation" ACK to the client when reservations are combined, and the client then proceeds to request data at the start time. If a reservation request reaches the producer node, it simply answers with a "RESV" message that signifies a successful reservation up to the data producer, and the client starts the retrieval at the start time. Note that for this study, these two different messages are only for informational and the client behavior does not depend on the message type.

When two requests are merged, the start time for each reservation remains the same, and the clients are expected to start fetching data at the requested start time. An extension of our protocol might ask a client to begin retrieving at a specific time or specify a deadline. However, we do not consider this in our work. If enough resources are not available at a node, it sends a NACK back. A NACK clears the pending reservations at all downstream hops. We discuss more in Section. 3.6.

3.4 Bandwidth Reservation

On receiving a reservation Interest, a node reserves the appropriate amount of downstream bandwidth for future incoming Data Packets. We assume Interest packets are small and the network can support them without requiring a bandwidth reservation. On successful reservation, the node forwards the reservation Interest upstream. If the node is the publisher, it returns a RESV message.

In our protocol, we enforce a bandwidth quota for each prefix by controlling the Interest forwarding rate. We assume the NFD can forward a finite number of packets per seconds and divide this amount appropriately among multiple reservations; since Interest forwarding rate dictates data rate, simply controlling the Interest forwarding rate should be sufficient for our study.

However, NDN Data packet sizes are variable and hard to predict. In the cases where Data sizes are known, such as for science data, forwarding rates are easy to calculate. When returning data sizes are unknown, the forwarding strategy can predict how much bandwidth is being used based on observed Data packet sizes. When return data rate goes above the quota, the strategy asks the client to slow down Interest sending rate, either using a NACK or other congestion control mechanisms.

3.5 Cache Reservation and Strategic Caching

Unlike IP networks where bandwidth is the only reservable resource, NDN nodes can also reserve in-network caches and reduce bandwidth consumption at the expense of storage. In our protocol, contents are strategically cached on a long-term, disk-based, in-network storage. We envision the caches to be disk-based and significantly larger than the in-memory CS, in the order of several TB or more. Which content should be strategically cached for future requests requires foreknowledge and therefore, is not practical. However, for scientific workflows,it is typical for end users to submit their requests before the actual data transfers[6], allowing the network to optimize the transfers.

In our work, we reserve cache space along with bandwidth for the reservations. If another request for the same data with an extended deadline comes in, our strategy is to simply cache the content until the new deadline. Request 3 and 4 in Table. 1 demonstrates this; with our bandwidth reservation

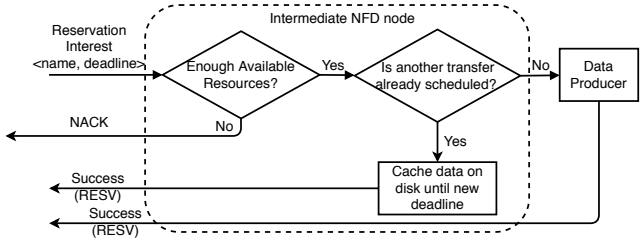

Figure 3: Reservation with strategic caching.

protocol without strategic caching, we will need to create two separate reservations, one from time 1-10 Sec and another from time 10-20 Sec. However, with the help of strategic caching, the network can reserve bandwidth from t_1 to t_{10} and cache the content until t_{20} at the intermediate node freeing up upstream bandwidth from the intermediate cache to the content producer.

Note that our protocol effectively extends the concepts of a vanilla NDN solution. In a normal NDN solution, caching exists with a traditional (e.g., LRU or LFU) replacement model and without any specific admission policies. While many of the caching benefits of our work can also come from vanilla NDN, there are two important distinctions between SCARI and vanilla NDN caching; first, in SCARI the temporal closeness is more explicitly defined for the selected objects. Second, unlike default NDN caching that caches all objects that pass through a particular node, SCARI selectively caches objects for a longer term. Since cache eviction in SCARI is independent of the normal traffic passing through the node, caching benefits are more stable and predictable.

SCARI can be useful in trading bandwidth for storage and bandwidth. In our protocol, RRMs cache contents depending on the observed set of duplicate requests. Note that our protocol does not explicitly allow users to request a specific amount of cache in the requests. There are various reasons for not allowing users to specify caching capacity but primarily because it might be difficult for the clients to know if their requests will overlap with other requests in the network. Additionally, allowing clients to request cache space might be beneficial for the requesting client but may not necessarily optimize the resource usage in the whole network.

Strategic caching also can automatically optimize content placement through the network. For example, a node may decide to cache data for **</xrootd>** until $t_{deadline}$ if it receives n requests through m different faces. This simple strategy places the content only at the nodes (e.g., Node 18 in Fig. 4) that lie at the intersection of future content paths. Since scientific datasets show a high degree of temporal and spatial locality [13], we anticipate that in-network strategic caching will be helpful for these data flows. In addition, this strategy ensures that not all content is strategically cached everywhere.

3.6 Path Teardown and Cleaning Up Defunct Reservations

If an upstream node can no longer support a reservation after initially confirming it, the node sends a downstream NACK that clears reservation states in downstream routers. The downstream routers, however, may choose to initiate another reservation if it has another alternate path to the same content. If the reservation on this other path is successful, the router simply discards the NACK, and the downstream nodes do not know about the path change. Otherwise, the router transmits the NACK downstream, and this process is repeated until either a new path is found, or the NACK reaches the client. If a new reservation is created, ICN network kepdf the network change transparent to the downstream nodes. However, in the worst case, the client and the ERM need to find an alternate.

Similarly, a client can send a cancellation Interest upstream in the following format: **</xrootd/reservation /cancel/</xrootd/data1/data_size/start_time/ deadline/bandwidth>>**. Similar to the reservation request, this Interest is forwarded upstream clearing reservation states at each upstream router. The Interest is forwarded until it reaches an intersection that has more than one reservation request for the same content. This intersection node deletes the reservation state *only* for the incoming face and replies with an ACK. However, it does not forward the cancellation request upstream. This operation has the effect of pruning a branch of the multicast tree; it does not affect any clients other than the one that is requesting reservation cancellation. If the multicast tree has only one branch, the cancellation request eventually reaches the data producer.

If a client fails without sending a cancellation request, the upstream nodes cancel the reservation when Interests for retrieving data do not arrive. The RRMs might also send periodic downstream messages similar to heartbeat messages for checking if the client is still alive and interested in the reservation.

3.7 Priority of Reservations and Identification of Endpoints

One of the problems of creating and maintaining reservations in IP networks is identifying and maintaining the priority of reservations. In this work, we make the simplifying assumptions that we have sufficient amounts of caches in the network that can accommodate all necessary reservations and all reservations have the same priority.

Another critical problem for creating reservations is identifying the endpoints so that they can be assigned appropriate priority. As we mentioned earlier, we do not propose a protocol for DiffServ or IntServ in this work but propose a protocol that we can use for building such services. These are the areas we plan to investigate in the future.

4 SIMULATION SETUP

In this work, we use an ndnSIM based simulation to evaluate our protocol from the perspective of the user, the network,

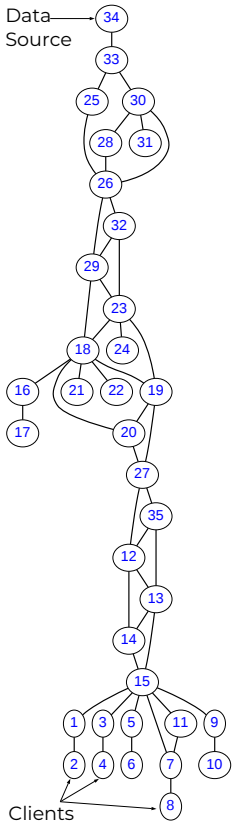

Figure 4: Evaluation topology for SCARI

and the data producer. To show the benefits of our protocol, we use a topology derived from a real data access log recorded at a single server[13]. Figure 4 shows the topology that we used for our study. Since the original topology contained a large number of nodes, such large topology slows the simulation down, and we pruned the excessive nodes on the paths of the topology without changing the actual connectivity. The new topology contains 35 nodes, one server/data producer, and nine clients.

In our topology, node 34 is the data producer, intermediate nodes are routers with RRMs, and the leaves are the clients with ERMs. For this study, we assume the data source (node 34) and the consumers (leaf nodes in the graph) as part of the same network and we use shortest path routing for our simulation. Using this topology, we conduct our simulations for the three approaches: IP-reservation, NDN-without-caching, and NDN-with-caching. To simplify our simulation, we set the strategic caching size to unlimited, though we found the actual caches could be much smaller, around 500GB.

We pick the top 999 requests from each client, a total of 8991 requests. 7965 of these requests contain the unique data names, and the rest are duplicates. The clients create and submit the reservation requests to their local ERMs that forward them upstream. For creating duplicate requests, we

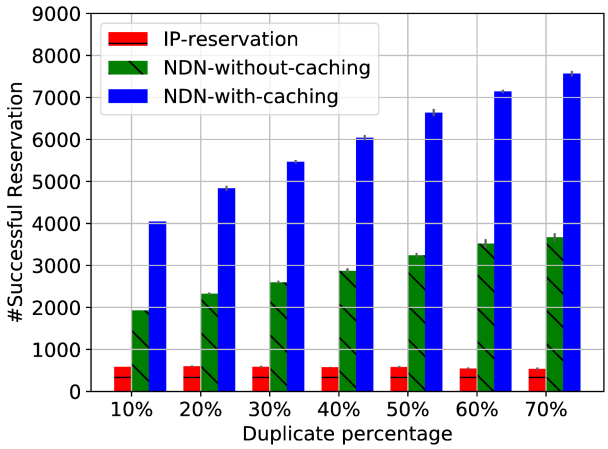

Figure 5: Successful reservation requests

choose one client and keep its access log intact, but substitute some of the other clients' requests using randomly chosen requests from the first client's request set. Each client then sends the reservation requests using the times in the original log, with data sizes ranging between 1 MB and 49.4 GB.

In our scenario, all links had 1Gbps bandwidth. To allow for realistic deadlines, we calculate the deadlines based on TCP transfer times over a 100Mbps link. Before the actual data retrieval happens, the client must successfully reserve the bandwidth on the path towards the server. The actual data can be cached but the ACK or RESV messages are uncacheable.

5 EVALUATION

In this section, we evaluate our protocol from the perspective of the end user, the network, and the data producer. We start with the scenario where all reservations are stand-alone, as is the case in today's IP networks, and no aggregation is possible. However, if the reservation requests are temporally close, our protocol can merge them in an effective multicast group for the subsequent data transfer, ensuring only one request reaches the server. While a vanilla NDN solution can also provide aggregation and caching benefits automatically, the temporal closeness is explicitly defined in SCARI, allowing the network more control over how and where the data is placed.

Figure 5 compares the number of successful reservation requests between IP and NDN. Since IP is unable to aggregate duplicate reservation requests, each request must reserve bandwidth separately on all on-path nodes up to the server. We found that in our simulation, the number of successful reservations over IP remains constant around 600 even when many of these reservation requests are duplicates. This happens since node 26 is the bottleneck in our topology. As the reservations on IP are separate and impossible to aggregate, they quickly consume the available bandwidth on node 26.

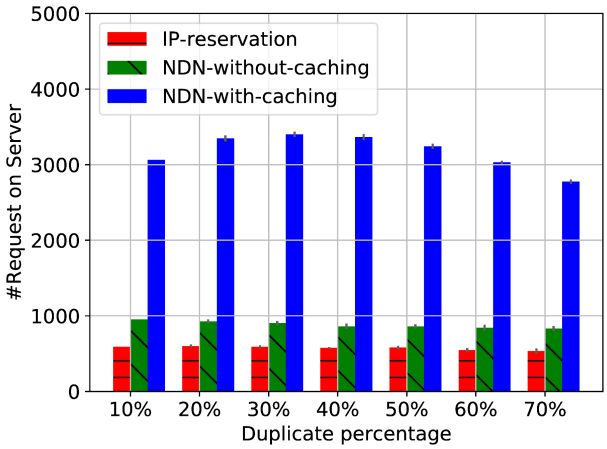

Figure 6: Reservation requests at data producer

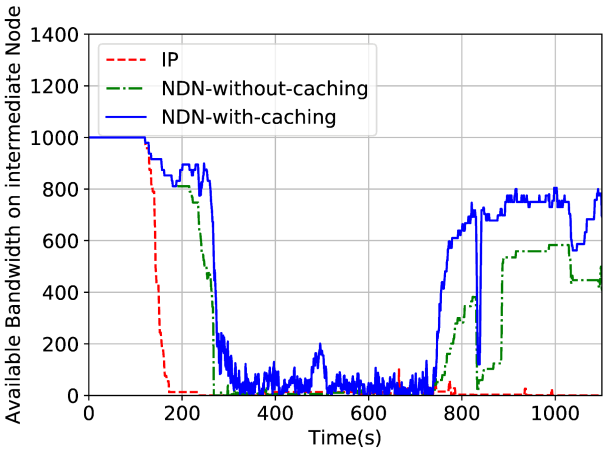

Figure 7: Free bandwidth at node 33

In contrast, with the same amount of resources, SCARI can support more reservations just by aggregating temporally close reservation requests. Figure 5 shows that with the original access log with approximately 11% duplicate requests, number of successful reservations with NDN is about 1200 more than IP. The number of successful reservation grows as the number of duplicate requests increases as we introduce more synthetic duplicates in the simulation. With 70 percent duplicates, the number of successful reservations grows to 3500 (out of 8991) with NDN-without-caching, a six-fold increase over IP.

With strategic caching, the results are even better. We find that with strategic caching, 7500 out of 8991 reservation requests could be satisfied, effectively increasing the number of reservations that the network can support with the same bandwidth. This improvement is the direct result of strategic caching where the intermediate nodes cache data locally for future requests, freeing up upstream bandwidth. We found that for our particular data access log, 500GB caches were sufficient to maximize this benefit. Note that as the network grows in size and complexity, we expect our protocol will be able to support even more reservations with the same amount of bandwidth. As more nodes create strategic caches at different points in the network and free up upstream bandwidth, the number of accommodated reservations should increase proportionally.

SCARI not only benefits the clients but also benefits data producers. In Figure 6 we plot the number of reservation requests that reach the data producer. We find that while the number of reservations that NDN can accommodate is much larger than the IP, the number of requests that reach the data producer is not proportionally more. This is an interesting observation; in IP, the intermediate router does not have enough bandwidth and therefore rejects most of the reservation requests even before they can reach the data producer. In NDN-without-caching, the number of requests that reach the producer is slightly reduced as the duplicate reservation requests increase. This decrease happens as in-network aggregation reuses most of the reservations. As we discussed before, when the duplicate percentage is low, in-network strategic caches serve some requests, allowing more unique requests to go through. However, as duplicate reservation requests increase, the number of requests that reach the data producer decreases linearly for NDN-with-caching scenario since the in-network strategic cache serves many of these requests, freeing up available bandwidth at the data producer.

Besides accommodating an increased number of reservations with the same amount of resources and freeing up bandwidth at the data producer, our protocol also benefits the network by reducing bandwidth consumption at the intermediate nodes. Figure 7 shows the amount of available bandwidth over time at the bottleneck (node 33) with 70 percent duplicate reservation requests. We find that both the NDN scenarios, with and without strategic caching, use less bandwidth compared to the IP scenario. Due to the interest of brevity, we do not show the number of reservations the intermediate node was able to support over time. However, the graph looks similar to Figure 7 where NDN scenarios were able to support a more substantial number of reservations compared to the IP scenario. Less resource usage at the network is good news as the network can support more data flows using the same amount of bandwidth.

In this section, we show that SCARI can reduce resource consumption for all stakeholder, the users, data producers, and the network. More sophisticated cache optimization, data transfer scheduling, and advanced strategies such as multi-path reservation may be able to reduce resource consumption even further. Though we have not investigated these aspects in this particular work, we plan to explore these in our future work.

6 CONCLUSIONS

In this work, we propose SCARI, a strategic caching and reservation protocol for ICN. In a first work that investigates resource reservation in NDN, we show that NDN based reservations can be more sophisticated and beneficial to all the stakeholders. Our protocol aggregates requests and efficiently frees up available bandwidth at the expense of in-network storage. While we have implemented and tested the basics of our protocol, we plan to evaluate how available caching capacity and heterogeneity in cache sizes affect our protocol performance. Finally, We are currently working on deploying our protocol with a data management framework where it can reserve resources for actual transfers. We plan to evaluate our protocol with the data we gather from this actual deployment.

The motivation for our work came from deadline based large scientific data transfers [7] [15][3] and as a result, the scope of this particular study is confined to large scientific data transfers over single-domain science networks such as Internet2 or ESNet. Several aspects of our protocol need to be extended. For example, we do not consider inter-domain reservations in our work. With interdomain routing, we will need to study the economics of reservations and dishonest behaviors. For example, a dishonest upstream might NACK a reservation request and force a downstream to use another, more expensive path. Similarly, a node might stop caching for a specific group of content, forcing retrieval from source. Besides, we need to investigate the business model of reservations and how the cost can be divided between multiple clients retrieving content using a shared reservation.

REFERENCES

[1] LHC Data Volume. https://home.cern/about/updates/2017/07/cern-data-centre-passes-200-petabyte-milestone.

[2] Brian Tierney and Joe Metzger, ESnet. High Performance Bulk Data Transfer. https://fasterdata.es.net/assets/fasterdata/JT-201010.pdf.

[3] Fan, C., Shannigrahi, S., DiBenedetto, S., Olschanowsky, C., Papadopoulos, C., and Newman, H. Managing scientific data with named data networking. In Proceedings of the Fifth International Workshop on Network-Aware Data Management (2015), ACM, p. 1.

[4] Guok, C., Engineer, E. N., and Robertson, D. Esnet on-demand secure circuits and advance reservation system (oscars). Internet2 Joint (2006).

[5] Mastorakis, S., Afanasyev, A., Moiseenko, I., and Zhang, L. ndnsim 2.0: A new version of the ndn simulator for ns-3. NDN, Technical Report NDN-0028 (2015).

[6] Newman, H., Mughal, A., Kcira, D., Legrand, I., Voicu, R., and Bunn, J. High speed scientific data transfers using software defined networking. In Proceedings of the Second Workshop on Innovating the Network for Data-Intensive Science (2015), ACM, p. 2.

[7] Olschanowsky, C., Shannigrahi, S., and Papadopoulos, C. Supporting climate research using named data networking. In Local & Metropolitan Area Networks (LANMAN), 2014 IEEE 20th International Workshop on (2014), IEEE, pp. 1–6.

[8] Orgerie, A.-C., Lefèvre, L., and Guérin-Lassous, I. Energy-efficient bandwidth reservation for bulk data transfers in dedicated wired networks. The Journal of Supercomputing 62, 3 (2012), 1139–1166.

[9] Pan, P. P., Hahne, E. L., and Schulzrinne, H. Bgrp: Sink-tree-based aggregation for inter-domain reservations. Journal of Communications and Networks 2, 2 (2000), 157–167.

[10] Patel, A., and Jue, J. Routing and scheduling for variable bandwidth advance reservation. IEEE/OSA Journal of Optical Communications and Networking 3, 12 (2011), 912–923.

[11] Rajah, K., Ranka, S., and Xia, Y. Advance reservations and scheduling for bulk transfers in research networks. IEEE Transactions on Parallel and Distributed Systems 20, 11 (2009), 1682–1697.

[12] Rehn, J., Barrass, T., Bonacorsi, D., Hernandez, J., Semeniouk, I., Tuura, L., and Wu, Y. Phedex high-throughput data transfer management system. Computing in High Energy and Nuclear Physics (CHEP) 2006 (2006).

[13] Shannigrahi, S., Fan, C., and Papadopoulos, C. Request aggregation, caching, and forwarding strategies for improving large climate data distribution with ndn: a case study. In Proceedings of the 4th ACM Conference on Information-Centric Networking (2017), ACM, pp. 54–65.

[14] Shannigrahi, S., Fan, C., and Papadopoulos, C. Named data networking strategies for improving large scientific data transfers. In 2018 IEEE International Conference on Communications Workshops (ICC Workshops): Information Centric Networking Solutions for Real World Applications (ICN-SRA) (ICC 2018 Workshop - ICN-SRA) (Kansas City, USA, May 2018).

[15] Shannigrahi, S., Papadopoulos, C., Yeh, E., Newman, H., Barczyk, A. J., Liu, R., Sim, A., Mughal, A., Monga, I., Vlimant, J.-R., et al. Named data networking in climate research and hep applications. In Journal of Physics: Conference Series (2015), vol. 664, IOP Publishing, p. 052033.

[16] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Crowley, P., Papadopoulos, C., Wang, L., Zhang, B., et al. Named data networking. ACM SIGCOMM Computer Communication Review 44, 3 (2014), 66–73.

[17] Zhang, L., Deering, S., Estrin, D., Shenker, S., and Zappala, D. Rsvp: A new resource reservation protocol. Network, IEEE 7, 5 (1993), 8–18.