Session Support for SCN

Mikael Gasparyan, Guillaume Corsini, Torsten Braun, Eryk Schiller, Jonnahtan Saltarin
Institute of Computer Science
University of Bern
Bern, Switzerland
{gasparyan,braun,schiller,saltarin}@inf.unibe.ch, guillaume.corsini@students.unibe.ch

***Abstract*—Service-Centric Networking (SCN) is a concept derived from Information-Centric Networks (ICN). In SCN, the service is in the center of the architectural design. Current efforts of SCN focus on design enhancements and the derivation of new components providing new functionality. In this article, we present session support for services in SCN. Our design makes use of existing hierarchical naming schemes to specify sessions using unique session identifiers. Sessions are established through a two-way handshake, which allows both the service consumer and provider to exchange their generated unique session identifiers. We have implemented and evaluated our SCN service session support mechanism, which provides promising results.**

Keywords—Service-Centric Networking; SCN; Service; Session; Named Data Networking; NDN; Information-Centric Networks; ICN; Content-Centric Networking; CCN; Service Session Support; Internet architecture

I. INTRODUCTION

The current Internet architecture is based on a host-to-host communication model, in which a requester has to possess an address of the provider to access content. In the current Internet model, information is exchanged through the TCP/IP protocol suite, which allows end-to-end communication in a multi-hop environment. In the last decades, the Internet traffic has grown dramatically; hence new requirements emerged, such as security, mobility, and service support. Limitations of the current Internet prompted researchers to develop fundamentally different architectures for future networking. As an example, Information-Centric Networking (ICN) is one of the most promising proposals for the next generation Internet architecture.

ICN proposes a fundamental paradigm shift in today's Internet architecture. It aims to change the current host-to-host communication model to a content-oriented one, in which network organization is centered around content. Content-Centric Networking (CCN) [1,2] provides a specification of ICN and an open source implementation is available online [3]. There is a significant number of other ICN architectures derived from it (e.g., Named-Data Networking (NDN) [4]). In CCN content is directly addressable and routable using unique content identifiers that replace the host identifier model of the current Internet. The content identifier naming scheme of CCN is based on hierarchical naming, in which the name is composed of a collection of strings of an arbitrary length. Apart from the content name, the identifiers contain information about version and segment of the content.

In CCN, a consumer sends Interest messages carrying the name of the desired content object. Then, the content provider replies with a Data message carrying the corresponding content object, i.e., the content object, whose name corresponds to the name of the Interest. The CCN architecture has been designed to handle content requests. However, in order to support the increasing number of services on the Internet, it is also essential to provide a next generation Internet architecture with integrated service support.

Service-Centric Networking (SCN) [5] enhances the CCN architecture by integrating support for service-oriented requests. SCN does not modify existing CCN functions. Instead, it extends CCN with addressing and routing capabilities that enable service support. Similarly to CCN, where content is offered by content providers to content consumer, services in SCN are offered by service providers towards service consumers through customized interfaces. Services give access to a set of specific software capabilities provided by service providers. Moreover, SCN takes advantage of existing capabilities offered by CCN such as caching.

An important component of services are sessions, which allow communicating parties to establish a semi-permanent message exchange. This paper provides the first, to the best of our knowledge, session support for SCN. Sessions are important because they allow communicating parties to create a context and perform a series of operations in the established context. For example, consider a cloud computing application, where the service provider should instantiate a virtual machine before it can process the incoming request. Without session support, requests requiring the same virtual machine might reach different service providers that should all instantiate the required virtual machine. Since instantiating a virtual machine is a time-consuming operation, this behavior is not desired. By using sessions, the requests requiring a single virtual machine will be routed to the same service provider. The selected service provider instantiates the virtual machine when the first session request arrives. Then, further requests concerning the created session can use the same virtual machine instance. Another example of applications that benefit from sessions are security related applications, e.g. encryption/decryption services. To process incoming data, these services require a key exchange, i.e. to create a context, which allows for efficient and secure processing. Sessions are beneficial for processing of continuous service requests requiring an execution context.

ISBN 978-3-901882-94-4

This paper is organized in the following manner. Related work is discussed in Section 2. Section 3 describes the newly proposed session support mechanism, by presenting its strength and weaknesses. Section 4 illustrates the implementation and evaluation of the proposed service session support. Finally, we conclude and present future work in Section 5.

II. Related Work

Ahlgren et al. [6] point out the missing support for sessions in ICN protocols, such as DONA [7], CCN [1], and PSIRP [8].

SCN, which builds upon extensions to CCN, provides support for services. Braun et al. [5] elaborate on still unsolved challenges of SCN, which require research attention. To the best of our knowledge, there is no existing solution integrating session support into SCN. This section gives an overview of SCN architectures and outlines their ability to deal with sessions.

NextServe [9] is an extension of the CCN architecture with support for services. NextServe allows service composition and uses a human-readable naming scheme. The object-oriented programming language inspires its naming scheme. NextServe does not integrate session capabilities.

SoCCeR [10] extends CCN to support services. SoCCeR integrates Ant Colony Optimization (ACO) to define the best forwarding face in the nodes for service requests. Service requests are then forwarded to the best available service provider based on the forwarding face ranking provided by the ACO algorithm. SoCCeR does not support sessions.

NFN [11] is another CCN extension to support services. It incorporates a lambda-expression resolution framework into CCN. NFN service requests contain a data block and a set of functions that have to be executed on the data block. NFN does not provide session support.

Serval [12] provides a modified TCP/IP protocol stack to allow applications in the network to exchange messages using service identifiers and provides mechanisms for load balancing. The newly added service layer is called SAL (service access layer), which is located between the transport and the network layer. SAL possesses service and flow tables. SAL uses these two tables to construct a mapping between service identifiers, service providers, flow identifiers, and interfaces. Serval uses special routers called service routers to support functionalities such as service discovery and load balancing. The Serval architectural approach can support function continuity for service requests by extending its designed flow identifier mechanism. However, a session support mechanism has currently not been implemented. Serval has been mainly designed for data centers, and its architectural design dramatically changes the TCP/IP stack, which makes integration in the current Internet architecture challenging. In contrast to Service-Centric Networking architectures based on the CCN concept, Serval does not support caching and has a single point of failure, which originates from the centralized aspect of its special service routers.

SOFIA [13] is a service-oriented ICN architecture. SOFIA decouples communication into two layers: the network and service layer. The network layer is responsible for data transmission using network addresses (e.g., IP) and the service layer provides flexible service processing by using service sessions. The service layer indicates the service name and the instance, to which a given request is addressed. The architectural design of SOFIA is similar to Serval; it differs, however, in the way service requests are forwarded. As previously stated, in SOFIA, service requests are identified by sessions. SOFIA uses its added service layer to set up a session. However, using the sessions created relies on the network layer. Consequently, after sessions are created, SOFIA forwards packets using destination addresses. This forwarding scheme goes against the fundamental principle of SCN, which is a belief that the future Internet requires a shift from the host (TCP/IP) to the service abstraction level that forwards data based on service identifiers and does not rely on the host-to-host communication model.

There are other Service-Centric Networking architectures [14]-[16]. However, none of them possesses an implemented session support or brings substantial session elements. The aim of our research is to fill the research gap in this area of SCN by designing and evaluating the first session support.

III. Design

In our novel mechanism for SCN session support, a service requester can establish a session with a service provider and use the created session during a time period. For our purposes, we modified the NDN framework to support services and sessions. NDN is a prominent implementation of the CCN architecture. The existing NDN implementation remains unchanged, but gets extended with service and session support. The following paragraphs will describe in detail our session support concept.

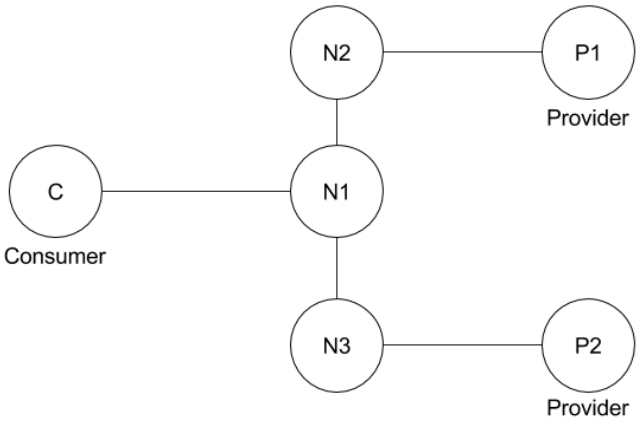

Fig. 1. Topology with a service consumer, a service provider, and three intermediate nodes

A. Preliminaries

The basic idea of a session is that two parties share a unique session identifier, which allows both sides to send and identify requests for a given session. In our design, both the service consumer and the service provider generate a unique identifier. The identifiers are then concatenated to provide a unique

session identifier, which is used by intermediate routers to forward the Interest and Data messages for a given session. The session route, as well as the pending session requests, are stored in the traditional Forwarding Information Base (FIB) and Pending Interest Table (PIT) tables. FIB is an NDN table, which saves forwarding faces for different prefix names, while PIT keeps track of pending Interest requests. Please note that session security aspects are out of the scope of this paper. However, the presented work can be extended to integrate session security support.

Fig. 1 illustrates a network topology with a service consumer C, two service providers P1 and P2, and three intermediate nodes n1, n2, and n3. To handle service requests, we use the traditional NDN content naming and forwarding scheme. Consider a request for service *getWeather*. The service name always starts with a prefix *service* that indicates a request for a service, allowing nodes to distinguish between content and service requests and preventing the service provider from additional matching operations. Incoming service requests can be sorted and sent faster to the underlying service for processing. The service name follows the prefix. In this example, the request asks for *service/getWeather*, where '/' indicates a separator.

Our aim is to concentrate on session support. Therefore, we do not consider other architectural aspects related to services such as service parameters or load balancing, which are out of the scope of this paper. In our example topology (c.f., Fig. 1), consumer C establishing a service session has to create and send a specific Interest message towards a service provider. The Interest message sent is provided to an arbitrary face leading towards the service name using the underlying forwarding strategy of NDN. The Interest message is then traditionally forwarded as NDN traffic by the intermediate routers to the provider, which in turn replies to the Interest with a Data message. This two-way handshake allows exchanging unique identifiers among the provider and consumer. The created session is stored by the intermediate nodes for future forwarding decisions. The session information is saved in existing NDN data structures. The session can be deleted by a timeout or a specific delete request. It is important to note that we only extend NDN without modifying its underlying architecture. In the following, we first extensively elaborate on session establishment. Second, we show how to use the established session, and finally, we illustrate session termination.

B. Session Establishment

First, a service consumer establishes a session. The session creation between two communicating entities is initiated through a Session Start Interest (SSI) request sent on the client side. The SSI carries a name, which follows a specific naming convention. The name is divided into three parts. The first part contains two elements: the prefix *service* indicating a service request followed by the service name.

As an example, */service/getWeather/* could be used, where *getWeather* is a specific service name. The second part provides the session establishment keyword *session/request*. Effectively, */service/getWeather/session/request* is generated by concatenating the two parts. The third remaining part carries the unique session identifier (generated on the consumer side) that is attached to the end of the request. The complete name containing the three elements could look like */service/getWeather/session/request/kdi32jd329j92rgq*, where *kdi32jd329j92rgq* is a unique session identifier.

Such a request initiates a two-way handshake establishing a new session. SCN forwards the SSI message as a traditional Interest message as illustrated in Fig. 2. Service consumer C requests the establishment of a session with any provider handling the required service (e.g., getWeather). First, consumer C sends an SSI message through an appropriate face leading to the corresponding service providers. The decision on the face selection depends on the underlying forwarding strategy of CCN. Second, intermediate node N1 forwards the request to node N2 or N3, depending again on the NDN forwarding strategy. In our example, the requested service is provided by service providers P1 and P2. Assuming that the underlying forwarding strategy sends the request along the path C – N1 – N3 – P2, service provider P2 will handle the session request.

A session identifier is a concatenation of two unique identifiers generated by both the consumer and provider. A provider, who receives an SSI message from a consumer with the consumer's unique session identifier, has in turn to generate its own provider unique session identifier. The provider's identifier is sent backward to the consumer. To accomplish this goal, the provider replies to the SSI message with a Service Start Data (SSD) message that contains the provider's unique identifier. The SSD message is forwarded as a usual NDN Data message through intermediate nodes N3 and N1 to reach consumer C. Table 1 shows the FIB and PIT tables of the provider node, after P2 received the SSI message. The PIT contains one entry created when the SSI message arrives. Like traditional NDN traffic, this PIT entry will be removed upon a data response. In our example, the SSD message corresponding to this PIT entry has been already sent.

The FIB table (c.f., Table II) of service provider P2 displays two entries pointing towards P2 itself. The first entry indicates that service requests towards *getWeather* should be handled locally and do not need to be forwarded any further. The second entry holds the session name, which is created upon the SSD reply. The session name contains the service name and a concatenation of the unique identifiers produced by the consumer and the provider. It indicates that provider P2 should locally handle messages on a per session basis.

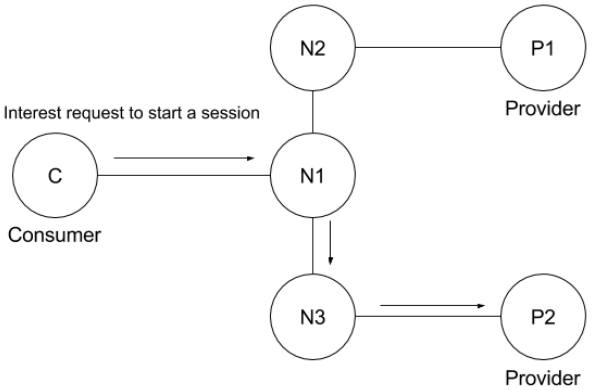

Fig. 2. Initiation of a new service session by consumer C

TABLE I. PIT TABLE OF PROVIDER NODE 2

Pending Interest Table	
Node	***Name***
N3	service/getWeather/session/request/kdi32jd329j92rgq

TABLE II. FIB TABLE OF PROVIDER NODE 2

Forwarding Information Base	
Node	***Name***
P2	service/getWeather
P2	service/getWeather/session/kdi32jd329j92rgqlkoq8i47ehobajua

The SSD message is forwarded through intermediate nodes on the route back towards the consumer. The standard routing mechanism of SCN is left unchanged; we take advantage of the traditional routing mechanism. Therefore, both ordinary content and services can be routed as usual. Forwarding of SSD messages in the backward direction is based upon the PIT entries on the corresponding routers. In our example, SSD is forwarded by intermediate nodes N1 and N2. Upon every forward, we create a new FIB entry on a given node to handle future forwarding decisions. The name of the newly created entry is composed of the service request name and concatenated unique session identifiers (from both consumer and provider). Table III illustrates the FIB entry of the intermediate node N1 when the SSD message arrived. The PIT entry (c.f., Table IV) reflects pending SSI requests. The pending entry will be removed once a corresponding SSD message goes through. We illustrate a newly created entry in the FIB table (c.f., Table III); it has the same name as the FIB entry of P2 and specifies the route for incoming interests in the newly created session.

TABLE III. FIB TABLE OF NODE N1

Forwarding Information Base	
Node	***Name***
N2, N3	service/getWeather
N3	service/getWeather/session/kdi32jd329j92rgqlkoq8i47ehobajua

TABLE IV. PIT TABLE OF NODE N1

Pending Interest Table	
Node	***Name***
C	service/getWeather/session/request/kdi32jd329j92rgq

The FIB entry is created when the SSD message as a reply for a corresponding SSI message arrives. The name is composed of the Interest Name, which contains information about the service name and the consumer session identifier. To construct the full entry name, the provider session identifier taken from the provider's SSD reply message is concatenated with the consumer's session identifier. This FIB entry sets the face, through which future requests belonging to a given session should be forwarded such that they reach the correct provider for the given session. In this example, N1 forwards to N3, which in turn forwards to P2.

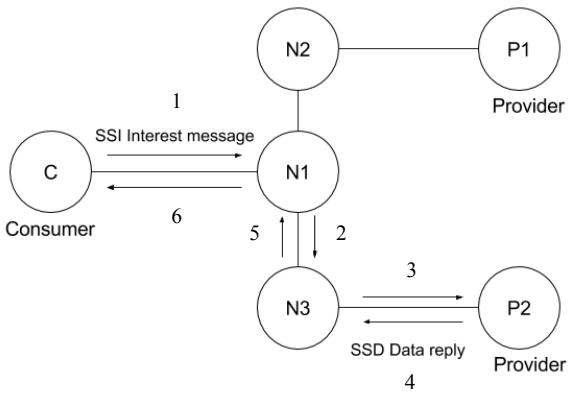

Fig. 3. The complete process of the session establishment

The complete process of the session creation between service consumer C and service provider P2 is illustrated in Fig. 3. To conclude, in a session initialization, a service consumer sends an SSI message, which specifies the name of the service requested in the name field, *session/request* indicating the request for the session start, and the consumer's generated session identifier. The forwarding routers usually forward the request to a service provider and populate their PIT table. Upon receiving the SSI message, the service provider generates another (provider's) session identifier and replies to the SSI with an SSD message containing the provider session identifier. Intermediate forwarding nodes save the session name in their FIB tables for future forwarding decisions. Upon receiving the SSD message, the service requester can start sending service requests using the newly created session name. In the following, we describe the use and termination of a session.

C. Session usage

To use the newly created session, the consumer sends its Interest request with the following naming convention, i.e., *service/[service-identifier]/session/[session-identifier]*. It contains two keywords: *service* and *session*, i.e. the service name, and the session identifier. This allows the requester to send Interests in the session, which are appropriately forwarded by the intermediate routers based on the standard forwarding and naming system of NDN, i.e., the use of FIB entries.

D. Session termination

To terminate a session, the consumer sends an Interest for the given session equipped with the keyword *terminate*. When the provider receives such an Interest, it replies with the confirmation of the session end. When the intermediate routers receive the provider confirmation, they delete the corresponding session FIB entry preventing from future use of a given session.

E. Node failure recovery

So far, session support for SCN was presented. Sessions allow for a dynamic establishment of paths between two communicating entities, i.e., the service provider and consumer. This path is used to exchange service requests in an arbitrary session. This subsection introduces a mechanism that allows our service session support to be network failure resistant. When a node goes down, the corresponding path failure can be detected by the service requester. A path is being considered as down, if after a specified number of Interest attempts the service requester (consumer) does not receive any Data reply messages. When the service requester detects a node failure, it has to find an alternative path to reach the service provider for a given session. Therefore, it starts sending a multicast Interest message to outgoing faces (i.e., through which the given service is accessible). The Interest message sent is called Service Node Discovery Interest (SNDI). The SNDI message content holds the keyword *service*, the service name, and the session identifier. The Interest is forwarded among the nodes towards the service providers. A service provider considers a given SNDI, if the session identifier matches one of the local sessions maintained. It then replies

with a Service Node Discovery Data (SNDD) message, with the session identifier in the payload. The SNDD message follows the path of the SNDI message in the opposite direction reaching the requesting consumer node. All the nodes along the SNDD path create an appropriate entry in their FIB tables for future forwarding decision for this session. This recovery mechanism also makes use of the existing SCN architecture.

IV. EVALUATION AND RESULTS

This section presents a preliminary evaluation of our service session support mechanism. We compared our service session support mechanism against the three existing strategies in NDN [4], namely, the Random, Multicast, and Best Route strategies [17]. The choice of these methods for comparison was motivated by the fact that there is no session support available in NDN. The Random strategy forwards the incoming request to a randomly selected face, which leads to the required service. The Multicast strategy as its name suggests broadcasts the request to all faces leading to the service required. The Best Route strategy uses network metrics to classify faces and forwards an incoming request to the face with the lowest cost.

The implementation and evaluation of our session support have been performed in the ndnSIM simulator [18]. NDNsim is currently the most advanced simulator for NDN; it is based on the ns-3 simulation framework [19]. Our implementation does not alter the NDN primitives, but extends it by integrating our session concept in the existing forwarding scheme and data structures. This is important, because it allows traditional NDN traffic to be handled as usual. It is worth noting that our implementation requires only a few changes to run on a real system.

We have evaluated our implementation in a testing scenario composed of 50 nodes (c.f., Fig. 4). We chose a graph that resembles, on a small-scale, the topology of the Internet. In our evaluation, we focus on the evaluation of the service delivery time with and without the integrated session support. The service delivery time is defined as elapsed time between the request sent and the response received by the consumer. Our evaluation topology contains 4 provider nodes and 8 consumer nodes, which are randomly selected leaf nodes in the network graph. The service consumers send service requests approximatively every second, by using a random variable of exponential distribution with the mean value equal to 1 second. The service provider nodes process each incoming service request with a processing time equal to a uniformly distributed random variable. We have performed two evaluations, in which processing time varies between 1500 ms and 2000 ms, and between 2500 ms and 3000 ms. All provider nodes run three different services that can be requested by service consumers.

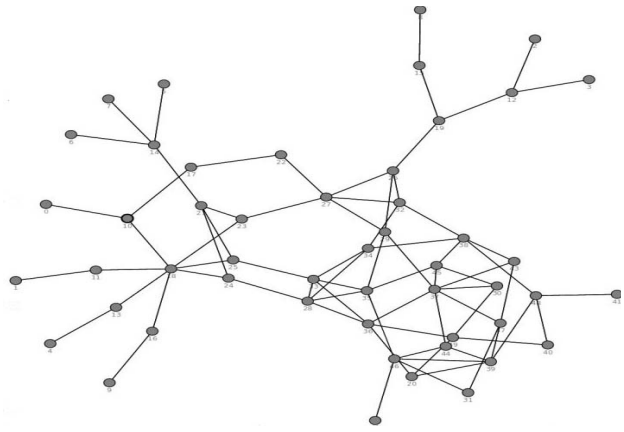

Fig. 4. Our evaluation topology composed of 50 nodes

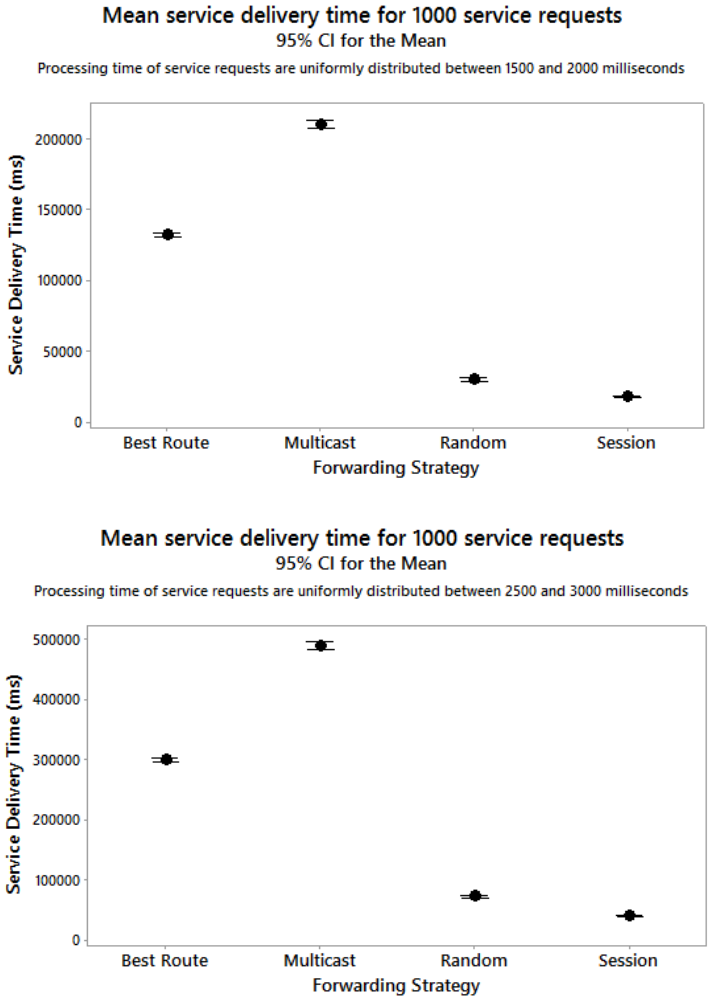

Fig. 5. Mean service delivery time and 95% confidence intervals for the mean concerning 1000 requests. The two charts compare our Session support mechanism with the three existing forwarding strategies in ndnSIM. Each chart shows the evaluation results for a different random uniformly distributed processing time.

We have executed this scenario comparing our session support mechanism against the three existing forwarding strategies of NDN (i.e., Best Route, Multicast, Random). When the session mechanism is used, a node establishes a session, uses the session twice sending a service request, and closes the session.

For each strategy, every service consumer sends 1000 service requests towards service providers. The service name of the Interest request is randomly selected from one of the three services that are installed on the providers. We ran two different simulations by varying the service request processing time. Each simulation was repeated 10 times, and finally, we used the mean service delivery time and confidence intervals for the mean to compare the strategies considered.

The charts in Fig. 5 show the mean service delivery time for the different strategies with different request processing time distributions. Each chart in Fig. 5 shows the mean service delivery time (circle) with their respective 95% confidence

intervals (horizontal lines) for the average of the 1000 service requests for the given request processing time distribution. In both charts, the confidence intervals for the four strategies do not overlap; this suggests a significant difference between them in terms of mean service delivery time for 1000 requests. Moreover, the mean service delivery time for our session support is significantly lower than that of the existing strategies in NDN.

The very high mean value for the Multicast strategy is due to missing load-balancing for service requests. The Multicast strategy forwards all requests to multiple providers and consequently, its requests are handled multiple times in the network. The relatively high service delivery time of the Best Route strategy is due to inefficient load-balancing among processing nodes. Out of the three strategies available in NDN, the Random strategy was the most efficient. The Random approach performance is an effect of forwarding requests randomly to different service providers. The charts show that a change of the processing time distribution does not affect the performance rank of the strategies considered. Therefore, our evaluation shows that our session support mechanism outperforms existing strategies in ndnSIM.

V. CONCLUSION AND FUTURE WORK

To the best of our knowledge, we have designed the first service session support scheme in SCN. We extended NDN to support services and to prototype our session support. Our session support mechanism is lightweight, and it makes use of the NDN architecture with respect to the forwarding mechanism and storage. This indicates that only a few modifications were required to integrate our solution into NDN. The NDN concept was left unchanged, and a few NDN extensions were provided. Therefore, content and service traffic can be forwarded in a traditional fashion.

We have integrated our service session support mechanism in the ndnSIM simulator and evaluated its performance with common forwarding strategies in a preliminary evaluation. Our session support mechanism outperforms the existing strategies in the evaluation conducted.

This paper provides a foundation for session support in SCN and ndnSIM. We will continue along this line i) to evaluate our network failure resistance mechanism and ii) to integrate new requirements for sessions. Due to the lightweight design of our session support, it will be easily extensible with relevant session requirements such as security or session information propagation.

REFERENCES

[1] Jacobson, V., Mosko, M., Smetters, D., & Garcia-Luna-Aceves, J. J. Content-centric networking. whitepaper (2009)

[2] Jacobson, V., “A New Way to Look at Networking”, Google Tech Talk, August 2006.

[3] CCNx | PARC's implementation of content-centric networking", Ccnx.org, 2016. [Online]. Available: http://www.ccnx.org.

[4] "Named Data Networking (NDN) - A Future Internet Architecture", Named Data Networking (NDN), 2017. [Online]. Available: http://named-data.net.

[5] Braun, T., Hilt, V., Hofmann, M., Rimac, I., Steiner, M., & Varvello, M. (2011, June). Service-centric networking. In Communications Workshops (ICC), 2011 IEEE International Conference on (pp. 1-6). IEEE.

[6] Ahlgren, B., Dannewitz, C., Kutscher, D., Ohlman, B., July 2012. A survey of information-centric networking. In IEEE Communication Magazine vol. 50, pp. 26-36. IEEE.

[7] Koponen, T., Chawla, M., Chun, B. G., Ermolinskiy, A., Kim, K. H., Shenker, S., & Stoica, I. (2007, August). A data-oriented (and beyond) network architecture. In ACM SIGCOMM Computer Communication Review (Vol. 37, No. 4, pp. 181-192). ACM.

[8] Tarkoma, S., Ain, M., & Visala, K. (2009, April). The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture. In Future Internet Assembly (pp. 102-111).

[9] Mansour, D., Braun, T., & Anastasiades, C. (2014, May). NextServe Framework: supporting services over content-centric networking. In International Conference on Wired/Wireless Internet Communications (pp. 189-199). Springer International Publishing.

[10] Shanbhag, S., Schwan, N., Rimac, I., & Varvello, M. (2011, August). SoCCeR: Services over content-centric routing. In Proceedings of the ACM SIGCOMM workshop on Information-centric networking (pp. 62-67). ACM.

[11] C. Tschudin and M. Sifalakis, "Named Functions for Media Delivery Orchestration", 2013 20th International Packet Video Workshop, 2013

[12] Nordström, E., Shue, D., Gopalan, P., Kiefer, R., Arye, M., Ko, S. Y., ... & Freedman, M. J. (2012, April). Serval: An end-host stack for service-centric networking. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 7-7). USENIX Association.

[13] Wu, Q., Li, Z., Zhou, J., Jiang, H., Hu, Z., Liu, Y., & Xie, G. (2014). SOFIA: toward service-oriented information centric networking. IEEE Network, 28(3), 12-18.

[14] Srinivasan, S., Singh, A., Batni, D., Lee, J. W., Schulzrinne, H., Hilt, V., & Kunzmann, G. (2012, June). Ccnxserv: Dynamic service scalability in information-centric networks. In Communications (ICC), 2012 IEEE International Conference on (pp. 2617-2622). IEEE.

[15] S. Srinivasan, J. Lee, E. Liu, M. Kester, H. Schulzrinne, V. Hilt, S. Seetharaman and A. Khan, "NetServ", Proceedings of the 2009 workshop on Re-architecting the internet - ReArch '09, 2009.

[16] Gasparyan, M., Braun, T., & Schiller, E. J. L-SCN: layered SCN architecture with Supernodes and Bloom Filters. In: CCNC 2017. IEEE

[17] "Forwarding Strategies," - NdnSIM Documentation, 2016. [Online]. Available: http://named-data.net/2.1/fw.html.

[18] Mastorakis, S., Afanasyev, A., Moiseenko, I., & Zhang, L. (2015). ndnSIM 2.0: A new version of the NDN simulator for NS-3. NDN, Technical Report NDN-0028.

[19] "ns-3 Simulator" - ns-3 Project, 2016. [Online]. Available: https://www.nsnam.org.