Fault-Tolerant Session Support for Service-Centric Networking

Mikael Gasparyan, Ali Marandi, Eryk Schiller, Torsten Braun
University of Bern, Switzerland
{gasparyan,marandi,schiller,braun}@inf.unibe.ch

***Abstract*—Service-Centric Networking (SCN) is a Future Internet architecture extending ICN with support for services, in which addressing and routing centers around services. SCN provides session support, however, recovery mechanisms do not exist in SCN session management. Therefore, SCN sessions suffer from link and node failures. In this paper, we present the design, implementation, and evaluation of three link failure recovery techniques for sessions in SCN. The first mechanism is based upon propagating session identifiers within the network using Bloom filters, the second design is based upon the propagation of service provider identifiers, and the third design uses piggybacking for the propagation of service provider identifiers.**

***Index Terms*—Service-Centric Networking; SCN; Service Session Support; Fault-tolerance; Service; Session Management; Named Data Networking; Information-Centric Networking; Future Internet architecture;**

I. Introduction

The current Internet is based upon the host-to-host communication principle. A content requester establishes a connection to the content provider before the requested content is retrieved. To establish a connection from one host to another, the current Internet uses the TCP/IP protocol architecture. The Internet development has expanded in multiple dimensions, pushing researchers to investigate solutions that satisfy emerging requirements such as mobility and enhanced support for services. The evolution of the Internet has also prompted researchers to focus on the development of fundamentally new Future Internet architectures such as Information-Centric Networking. ICN shifts the current host-centric Internet paradigm towards a Future Internet architecture that forwards packets based on the content name instead of the content location. ICN aims to enable content addressing through content identifiers. A content requester sends an Interest, which carries a unique content identifier. The content identifier is used by intermediate nodes to forward the request from the content requester towards any content replica in the network. One of the most prominent implementations of the ICN concept is Named Data Networking (NDN) [1].

Service-Centric Networking (SCN) [2] introduces service support within ICN. SCN does not alter ICN primitives, but rather extends them with service support capabilities and benefits from the existing ICN architecture. In SCN, service providers offer services in the network. Services, being functions implemented in software, can be consumed by service consumers, which send Interest requests for desired services. Sessions, which allow us to establish a semi-permanent communication channel between a service consumer and a service provider, are a substantial element of service delivery. Moreover, sessions are important, because they allow us to establish an execution context between a service provider and a service consumer to process a series of operations in the created context. For example, sessions can be beneficial in the cloud for applications that require the instantiation of virtual machines. Without sessions, the underlying forwarding mechanism of ICN could redirect service queries towards different service providers, hence, requesting the instantiation of a virtual machine and context among multiple entities. Moreover, context-related information that has been already transmitted towards one provider cannot be recognized at another service location. The context is, therefore, considered lost and requires retransmissions.

978-3-903176-15-7

In our previous research [3], we presented and evaluated the first session support for SCN. Our session mechanism allows a service consumer to establish a session with a service provider through a two-way handshake. When the handshake is completed, the communicating parties can exchange data through the newly created session, which is identified by a unique session identifier. A session is only recognized at intermediate routers along a single path through which the session has been established. Alternative paths from the service provider to the service consumer are not available, meaning that the session breaks down when a link failure on the path occurs. Alternative paths from the service provider to the service consumer offer recovery from link and node failures as well as load-balancing mechanisms in the case of overload. The aim of this work is to integrate routing over alternative paths into our previously designed SCN session mechanism. Alternative paths provide fault-tolerance and improve other important network metrics such as network utilization.

In this paper, we present fault-tolerant session management using three different fault-tolerance mechanisms. The first mechanism uses Bloom filters [4] to propagate service session identifiers. The second mechanism propagates service provider identifiers to the network. The third mechanism uses piggybacking for propagating service provider identifiers, which significantly reduces the traffic overhead.

This paper is structured in the following way. Section II elaborates on the related work. Our three fault-tolerance mechanisms are presented in Section III. In Section IV, we evaluate these mechanisms. Finally, we conclude in Section V.

II. Related Work

ICN has three main components: Forwarding Information Base (FIB), Pending Interest Table (PIT), and Content Store (CS). The FIB is a lookup table containing a match between outgoing faces (i.e., interfaces) and prefixes; it enables us to select an appropriate outgoing face towards the object name of Interest. The PIT stores pending Interests that have been forwarded further but have not yet been satisfied. CS is a locally disposed cache that temporarily stores incoming Data packets. CS allows incoming Interest requests to be satisfied directly from the local cache without forwarding the Interest request further on [1].

SCN is an extension of ICN. In our previous work, we have extended SCN with the first session management mechanism. To our knowledge, currently, there is no other session management mechanism available in SCN. However, in case of link or node failures, the SCN session management mechanism does not provide alternative paths. In this section, we present an overview of existing ICN architectures providing service support. We focus on important aspects related to session management.

SoCCeR [5] combines ICN and Ant Colony Optimization (ACO) [6] to upgrade ICN with service request forwarding. The ACO population-based meta-heuristic mechanism is used to broadcast special Interest messages for randomly selected services. The information maintained in Interest messages is used by intermediate nodes to classify outgoing forwarding faces. Incoming service requests are forwarded by using appropriate faces for a given service. SoCCeR does not provide session support.

NextServe [7] extends ICN with service support. NextServe uses an object-oriented programming language style as the naming scheme. NextServe enables service composition by allowing a chain of services to be executed one after another. NextServe does not integrate session and fault-tolerance mechanisms.

Serval [8] extends the TCP/IP protocol stack with a service layer that enables applications to communicate using service identifiers. The newly added layer lies above the network layer. It stores mapping information on service names, flow identifiers, network addresses, and outgoing faces, which allows service flow-based routing. Serval modifies the TCP/IP protocol stack and uses special routers used for service related actions such as service discovery. The flow identifier mechanism can be extended to integrate fault-tolerant session support. However, Serval was designed for data centers and is based on a centralized router organization that introduces a central point of failure.

SOFIA [9] is a service-oriented ICN architecture similar to Serval. It integrates session-based service routing. However, its implementation does not rely on ICN primitives for service session forwarding, it forwards session messages using the TCP/IP protocol suite. SOFIA dramatically modifies the TCP/IP stack. It, therefore, creates deplorability problems and compatibility issues.

NFN [10] is an ICN architecture that extends ICN with service support capabilities. NFN uses a λ-expression language-like naming scheme. An NFN service request contains data and a set of functions that have to be executed over this data. Sessions are not supported by NFN.

Other architectures exist [11], [12], but none of them provides session support.

In our previous work [3], we have implemented and evaluated the first SCN session support mechanism. Our designed session mechanism uses a two-way handshake to establish a session between a service provider and a service consumer. The session is established through the exchange of session identifiers generated by a service consumer and a service provider. Our design has three phases: session establishment, session use, and session termination. A session is only known by intermediate routers along the single path, over which the session was established. A session is lost when a node or link failure occurs, while there is no alternative path for an already established session. Therefore, our previous solution enables session support. However, it is not fault-tolerant. In this work, we enhance our previously designed SCN session management with alternative paths.

III. Fault-Tolerant Session Support

In this section, we present our three newly designed mechanisms that enable fault-tolerant session support for SCN. The first mechanism consists of Bloom filter-based propagation of session identifiers, the second proposal is based on dissemination of service provider identifiers, and the third approach uses piggybacking for the exchange of service provider identifiers. Each mechanism has its advantages and disadvantages, which will be elaborated in detail in the remaining part of this section. Before starting the description of our mechanisms, in the following subsection, we explain the essential components of the previously designed session support [3].

A. Session Support

Sessions allow two communicating parties to establish a semi-permanent communication channel for message exchange. A session is identified by a session identifier. Our designed session support mechanism for SCN uses the session identifier, which is a concatenation of two identifiers generated by the service consumer and the service provider. The session mechanism has three phases: session establishment, session use, and session termination. This section will describe these parts and emphasize important elements in fault-tolerant session management.

Session establishment: To use a session, we need to first establish it between two communicating parties. Our session mechanism uses a two-way handshake to exchange and establish a unique session identifier (ID) for a given session. The mechanism works as follows: the service consumer sends a Session Start Interest (SSI) request to the network. SSI is a regular ICN Interest message, which follows a specific naming convention of SCN. The SSI name is composed of three parts. The first part such as "/service/getServiceA" indicates an Interest for a service through the "/service" prefix, which

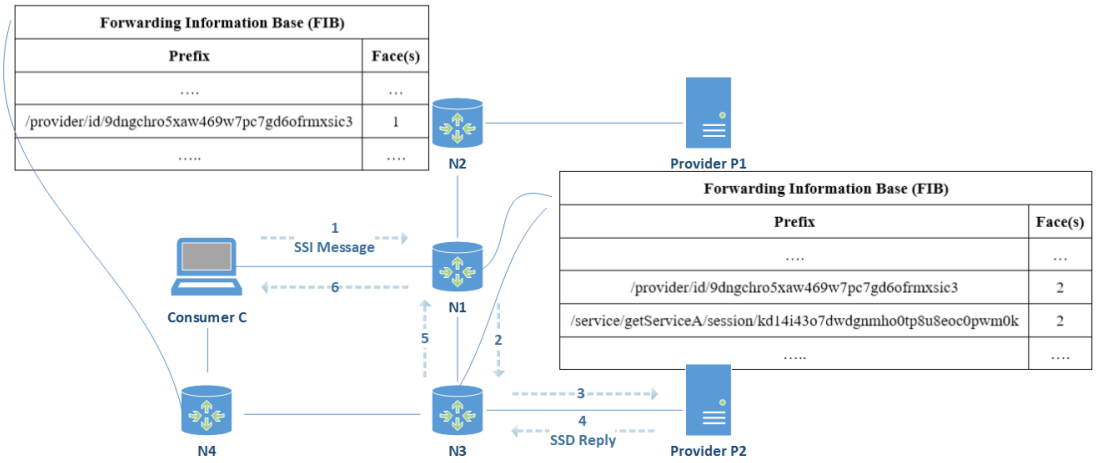

Figure 1: Session Establishment

is followed by the service name, e.g., "getServiceA". The second part contains two consecutive components: "session" and "request" keywords: "/session/request". They specify that the SSI starts a new session. The last component of the SSI name holds the unique session identifier generated by the service consumer. It is a randomly generated identifier of size equal to 16 characters. The service identifier is composed of 16 characters, e.g., "kd14i43o7dwdgnmh". An example SSI request name for the "getServiceA" service has the following structure: "/service/getServiceA/session/request/kd14i43o7dwdgnmh".

The SSI message is forwarded by the underlying forwarding scheme of the ICN network as a regular ICN Interest request. Fig. 1 illustrates an example topology composed of two service providers P1 and P2, one service consumer C, and three intermediate routers N1, N2, and N3. Three dashed arrows (i.e., 1-3) show the path traveled by the SSI message from consumer C to a provider that possesses the requested service. In this example, consumer C sends the example SSI message described in the previous paragraph. The SSI message arrives at node N1, which forwards it to N3 using ICN Interest forwarding. In turn, N3 forwards the SSI to the service provider P3 that offers the requested service.

Upon receiving the SSI message (c.f., Fig. 1), P2 replies with a Service Start Data (SSD) message. SSD is a regular ICN Data reply issued in response to an ordinary Interest request received. SSD contains a randomly generated unique identifier, e.g., "o0tp8u8eoc0pwm0k", which is generated by the provider for a given session. The reply is forwarded backwards towards the consumer C (Fig. 1, dashed arrows 4-6) through the intermediate nodes previously forwarding the corresponding SSI. Prior to the forwarding of the SSD reply, the intermediate nodes have to populate their FIB tables to enable further forwarding of session Interest messages towards a given service provider (P2 in our example). The intermediate nodes add a FIB entry (i.e., in the FIB table), which contains both unique identifiers generated by the consumer and the service provider. The session identifier of the service consumer is gathered from the SSD response name prefix, while the identifier of the service provider is collected upon receiving the SSD reply. The concatenation of the two unique identifiers derives the final session identifier.

Fig. 1 depicts the two-way handshake realized through the exchange of SSI and SSD messages. Let us continue the previous example. The SSI message carries "/service/getServiceA/session/request/kd14i43o7dwdgnmh" in the service name field and the service provider stores a unique ID "o0tp8u8eoc0pwm0k" in the SSD reply. During the forwarding phase of the SSD reply, intermediate nodes (Fig. 1, dashed arrows 4-6) insert "/service/getServiceA/session/kd14i43o7dwdgnmho0tp8u8eoc0pwm0k" pointing towards the faces through which the service provider P2 is reachable into their FIB tables, where the session identifier is the concatenation of the requester's unique ID "kd14i43o7dwdgnmh" and the provider's unique ID "o0tp8u8eoc0pwm0k". Please notice that the Interest name and newly created FIB entries contain a keyword "service" indicating a service request, followed by the service name, the keyword "session", which indicates a session request/entry, and the unique session identifier. Consumer C can start sending session requests upon receiving the SSD reply from provider P2. The next paragraph briefly explains the use and termination of a session.

Session use and termination: A consumer can start using a session when the session is established meaning that the service consumer successfully received the SSD reply to its initial SSI message. To use a session, the consumer creates Interest requests that use the same naming convention as the FIB entries previously described in the previous paragraph. In our example, a request for the "getServiceA" service using the previously instantiated session of ID "kd14i43o7dwdgnmho0tp8u8eoc0pwm0k" has the following structure: "/service/getServiceA/session/kd14i43o7dwdgnmho0tp8u8eoc0pwm0k", where the first component is the keyword "service", the second component is the name of the requested service, the third component indicates a session request, and the last component is the session identifier. The intermediate nodes can properly forward this name prefix, while the two-way handshake appropriately

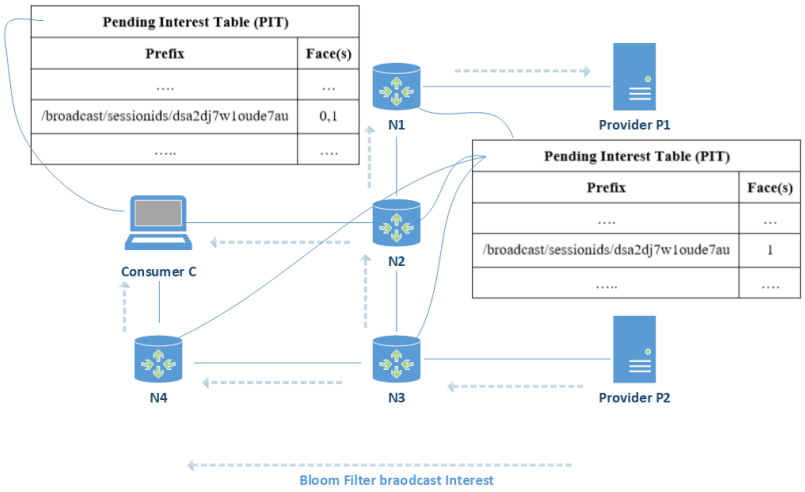

Figure 2: Bloom filter broadcast.

configured the FIB entries among the forwarding routers for a given session, as explained in the previous paragraph. When the session is not needed, the service consumer terminates it by sending another specific Interest. The Interest's name is a regular session Interest with the keyword terminate provided at the end of the Interest name. The service provider replies to this Interest message with an empty Data message. The reply traverses through the intermediate nodes towards the service consumer. Upon forwarding the reply, the intermediate nodes remove the FIB entries related to the terminated session. In our example, the Interest sent by the service consumer to terminate the example session instantiated in previous paragraphs has the following structure: "/service/getServiceA/session/kd14i43o7dwdgnmho0tp8u8eoc0pwm0k/terminate". For further details concerning our session management, please refer to our previous work [3].

This session support for SCN is not fault-tolerant; it supports only one path, through which the communication between a service consumer and a service provider is provided. This is due to the fact that the session identifier of a session is only known by the intermediate routers, through which the session was established.

B. Fault-Tolerant Session Support Mechanisms

This section presents three fault-tolerance approaches for SCN session management. The central goal of the three strategies is to provide alternative paths to sessions. Alternative paths, which connect a service provider and a service consumer, allow for fault-tolerance. When links or routers fail along the currently used path, we can use an alternative path to uninterruptedly forward messages between communicating entities of a session. Our session management is based on information propagation in the network allowing for the recognition of currently maintained paths by other routers that do not forward messages for a given session. The first mechanism is based upon propagation of session identifiers using Bloom filters, the second design is based on the propagation of service provider identifiers, and the third design uses piggybacking for the dissemination of service provider identifiers.

1) Session Information Propagation Using Bloom Filters: Bloom filters [4] are a useful tool for representing sets in a compact way. A Bloom filter consists of a bit vector and hash functions. If a Bloom filter is used, false positive errors of the search operation might occur. However, false negatives are not possible. If N is the number of mapped elements onto a Bloom filter, M is the bit vector size, K is the number of hash functions, and P is the desired probability of false positive errors, the trade-off between the bit vector size and the false positive rate is given by [4]:

$$M = -\frac{N \ln(P)}{\ln^2(2)}$$

Bloom filters provide efficient search operations in a set of elements. If set S is represented using a regular array, the computational complexity of finding an element is equal to $O(N)$, where N is the cardinality of S. However, if S is represented using a Bloom filter, the complexity of the search operation is $O(K)$. If S is a big set and $K << N$, then $O(K) << O(N)$. In the literature, Bloom filters have been already proposed for efficient routing protocols in ICN [12]–[15].

The first fault-tolerant session management uses Bloom filters to propagate a set of session identifiers in the network (i.e., the element set consists of session identifiers). Bloom filters disseminate session identifiers currently maintained in the network. The propagation of Bloom filters is realized through the broadcast of regular ICN Interests (i.e., Broadcast Interests) having a name, in which the prefix "/broadcast/sessionids/" is followed by a randomly generated identifier. In our example illustrated in Fig. 2, P2 can use the following naming structure: "/broadcast/sessionids/dsa2dj7w1oude7au", where "dsa2dj7w1oude7au" is P2's unique randomly generated identifier. The first two name components indicate a broadcast Interest containing provider's session identifiers. The third Interest name component ensures that Broadcast Interests from distinct providers cannot be discarded by ICN as redundant forwarding due to unique name components (i.e., different random identifiers of different providers). Furthermore, loop-

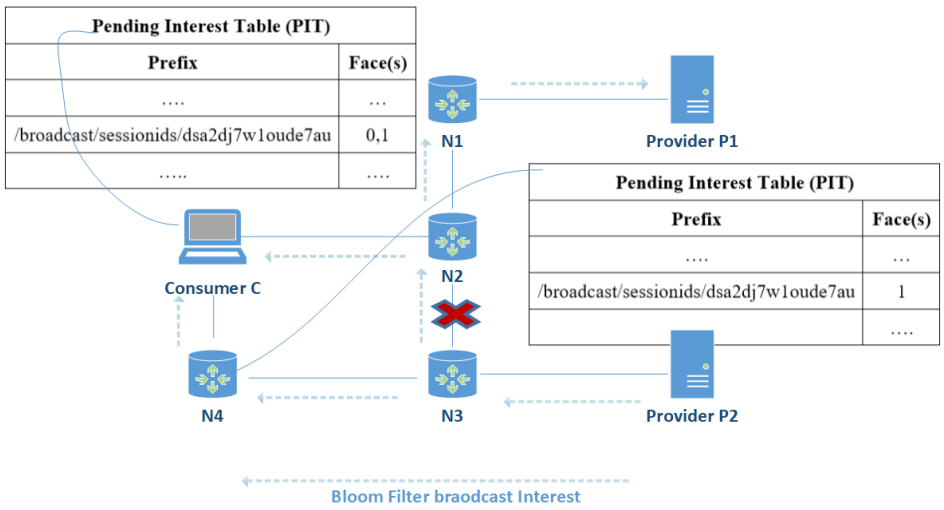

Figure 3: Link failure between N2 and N3.

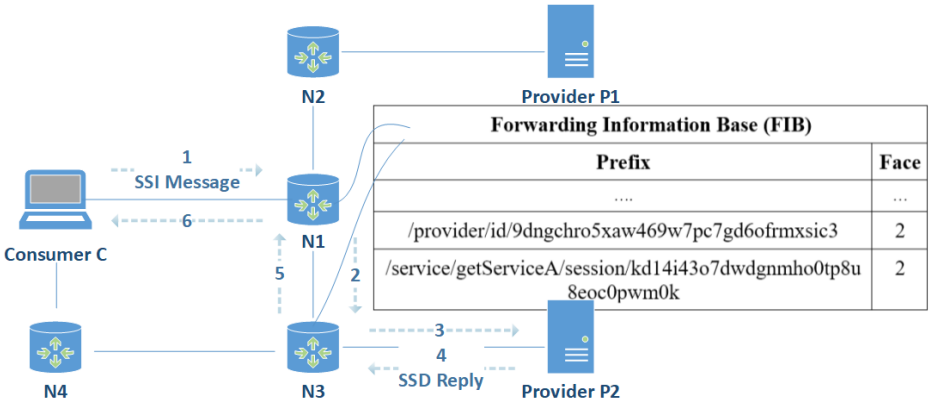

Figure 4: Session Establishment.

free Interest propagation is guaranteed by the Interest's Nonce field [16].

When a node receives Broadcast Interests, it forwards them over all faces except the Interest's face of arrival (i.e., incoming face). Similar to regular Interest messages, Broadcast Interest messages are stored in PIT tables. As regular PIT entries they will be removed from the PIT table when their timeout elapses. When a node receives two distinct Broadcast Interests originated from different service providers, it adds two separate entries in its PIT table, i.e., one for each distinct Interest. Fig. 2 shows an example propagation of session identifiers using Bloom filters. In this example, provider P2 propagates a Bloom filter containing session identifiers. The identifiers are propagated in the network (dashed arrows), because routers forward Interests containing Bloom filters further also adding entries among PIT tables.

Fig. 2 shows two PIT tables belonging to consumer C and intermediate node N2. Both tables contain the broadcast entry added upon receiving an Interest request. The PIT table of consumer C has two entries in its face column, because it received Interests from two different faces. In our example, consumer C can reach provider P2 through N2 and N4, which can enhance session management with fault-tolerance. In Fig. 3 consumer C and provider P2 have established a session through nodes N2 and N3. All sessions are then advertised by provider P2, which periodically broadcasts its session identifiers (c.f., Fig. 2). In the case of the link failure between N2 and N3, consumer C can use the alternative path via N4. When a pre-defined number of session Interest retransmissions fails along a given path, consumer C and the intermediate router N4 find an alternative path by checking the session identifier in the received Bloom filters stored in PIT tables. Intermediate router N3 knows about the session, because the session was set up over N3.

Bloom filters do not have false negative errors, however, false positive errors may occur. In the case of false positive matches, the Interest will be forwarded through one or multiple wrong faces, then the Interest will be discarded by recipient nodes. Please notice that Bloom filters drastically compress the size of information broadcast in the network and, therefore, increase the scalability of the protocol with an increasing number of nodes in the network.

2) Service Provider Identifier Propagation: Our first fault-tolerant session management is based upon the propagation of session identifiers using Bloom filters. There is, however, a session identifier for each created session. Therefore, each provider has to propagate multiple session identifiers in the network.

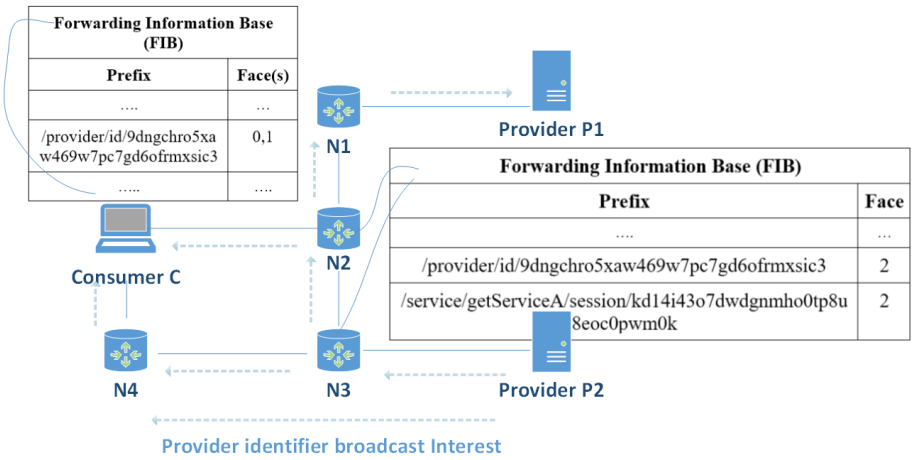

Figure 5: Provider identifier broadcast.

The second derived mechanism uses provider identifiers instead of session identifiers. Propagating provider identifiers, instead of session identifiers (i.e., as in the first strategy), has two main advantages. The first advantage is that there is only one identifier for a given provider. This means that only one identifier per provider needs to be propagated in the network. We equip each service provider with a randomly generated unique identifier composed of 32 characters. This identifier is propagated through the network by Interest messages. Broadcasting provider identifiers instead of session identifiers requires, however, the adaptation of the initial session management, while the request forwarding is based upon session identifiers. It requires us to perform changes in the presented session support mechanism integrating provider identifiers into the forwarding scheme.

Fig. 4 illustrates a regular SSI/SSD handshake between consumer C and provider P2 (c.f., Section III-A). Typically, the SSD reply possesses a session identifier. In addition to previous fields, we include the provider identifier within the SSD reply. Upon SSD forwarding, the intermediate nodes add two FIB entries in their FIB tables as shown in Fig. 5. The first entry contains the provider identifier. It starts with the prefix "/provider/id" which holds the keywords "provider" and "id", it is then followed by the provider identifier extracted from the SSD content. The second entry is the session identifier extracted from SSD.

Once a session has been established, consumers can use the session by sending service requests using the following Interest naming convention /service/[service-identifier]/session/[provider-identifier]/[session-identifier].
In our above example the Interest name used by the service consumer C to utilize the newly created session would look as follows: "/service/getSearviceA/session/9dngchro5xaw469w7pc7gd6ofrmxsic3/kd14i43o7dwdgnmho0tp8u8eoc0pwm0k".

The name contains two keywords "service" and "session", The "service" keyword is followed by the service name, while the "session" keyword is followed by the session and provider identifiers. This information enables us to forward the request to the appropriate service provider by using both session or provider identifiers. The session identifier is only known by the intermediate nodes, through which the session was established. The provider identifier is broadcast with Interest messages to the entire network using a Provider Identifier Broadcast Interest (PIBI) message. Therefore, the provider identifier is recognized by all nodes in the network.

Fig. 5 shows FIB entries in consumer C and intermediate nodes N2 and N3 after broadcasting PIBI messages. We continue using the example from the previous paragraph. Intermediate nodes N2 and N3 have two FIB entries created upon the forwarding of the SSD reply. The first entry enables the request to be forwarded using the provider identifier. The second entry allows for regular session forwarding (c.f., Section III-A). Consumer C, as well as intermediate nodes N4 and N1, maintain the identifier prefix added to their FIB tables, which was created upon receiving the PIBI message.

This strategy extends the initial session support mechanism, explained in Section III-A. We integrate service provider identifiers (SPI). The SPI is broadcast with PIBI messages to the whole network and is used for FIB population by the nodes in the network. Additionally, compared to the regular session support mechanism [3], the service requests do not only contain the session identifier, but also the provider identifier. This offers alternative paths for service session request forwarding. For example, in Fig. 5, when node N2 is down, the request from consumer C can be forwarded using the service provider identifier through node N4. The introduction of service provider identifiers lowers the message overhead dramatically, since a provider identifier is rarely changed.

3) Provider Identifiers Piggybacking: The third strategy is an alternative specification of the second strategy. We do not use PIBI messages to broadcast SPI. Instead, we only rely on piggybacked SPI using SSD messages. This strategy does not create any message overhead, because it piggybacks information. Table I compares the three proposed mechanisms.

IV. Evaluation and Results

A. Evaluation

This section presents an evaluation of our three fault-tolerant strategies for SCN session support. We implement

Table I: Comparison of three fault-tolerant mechanisms.

	Bloom Filter Broadcast of Session Identifiers	**Provider Identifier Broadcast**	**Provider Identifier Piggybacking**
Fault-tolerance	Yes	Yes	Yes
Based-on	Session Identifiers	Session and Provider Identifiers	Session and Provider Identifiers
Broadcasted information	Bloom filter containing session identifiers	Provider identifiers	-
Scalability	Low	High	Very High
Stability	High	Very High	Low
Advantages	Broadcasting by using Bloom filter, do not use provider identifiers	Provider identifier information has only to be broadcasted in long periods	No message overhead
Disadvantages	Message overhead	The initial session support mechanism has to be extended and provider identifiers introduced	Session information propagation may be slow. The initial session support mechanism has to be extended and provider identifiers introduced

Figure 6: Evaluation topology.

and compare them in ndnSIM [17]. ndnSIM is an ns-3 based simulator for the NDN [1] implementation.

The three strategies were evaluated in a network topology composed of 100 nodes. The created topology is shown in Fig. 6. It resembles the topology of the Internet at a small scale with high connectivity hub nodes and single connectivity leaf nodes.

We have selected three leaf nodes as service consumers and one leaf node as a service provider. The service consumer nodes establish consecutive sessions with the service provider. Each session is used by sending 48 service Interest requests. Finally, the session is terminated. In total, 50 Interest messages are sent for each session: 1 Interest to establish a session,

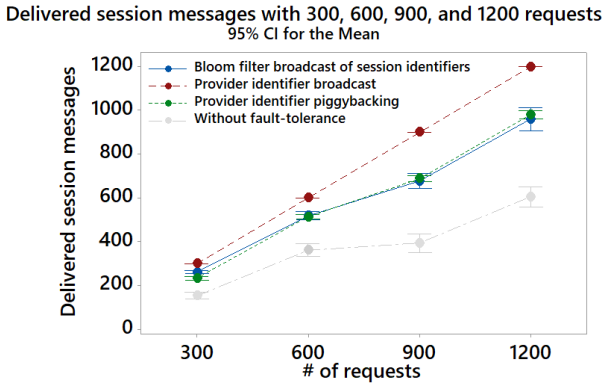

Figure 7: Delivered session messages for the three fault-tolerant mechanisms and the session mechanism without fault-tolerance for the four evaluation setups.

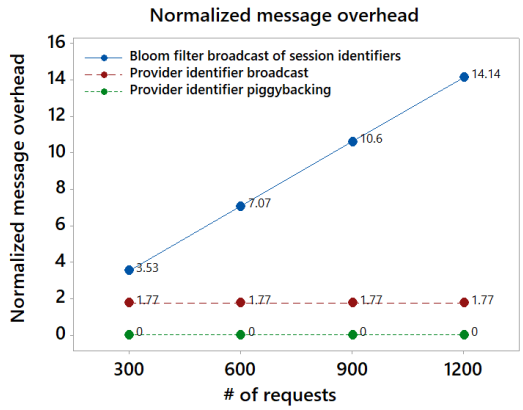

Figure 8: Normalized message overhead for the three fault-tolerant mechanisms and the four evaluation setups.

48 Interests that use the session, and 1 Interest to terminate the session. We have set four evaluation setups where each service consumer sends 100, 200, 300, and 400 service Interest messages that need to be satisfied with the responses of the service provider. We have simulated link failures by randomly selecting links located in between the service provider and service consumers. The link failure time is randomly selected using a uniform distribution in the range [1 s, T], where T is the simulation execution time. The chosen link location is randomly selected among links connecting service consumers and the service provider. We do not, however, select links that divide the network into two disconnected components. The request frequency is set to one Interest per second. In the case of the first mechanism, the Bloom filters of size 1450 bits are broadcast every 10 seconds, but only if new sessions are created. In the second mechanism, the identifier information is broadcast at the request time of the instantiation of the first session. The third mechanism uses piggybacking with a piggybacked payload of size equal to 16 bytes, hence, there is no broadcast. For the four evaluation setups, in total, there are 300, 600, 900, and 1200 messages sent in the network to create, use, and terminate sessions. Each experiment is repeated ten times. The mechanisms are compared on a statistical basis.

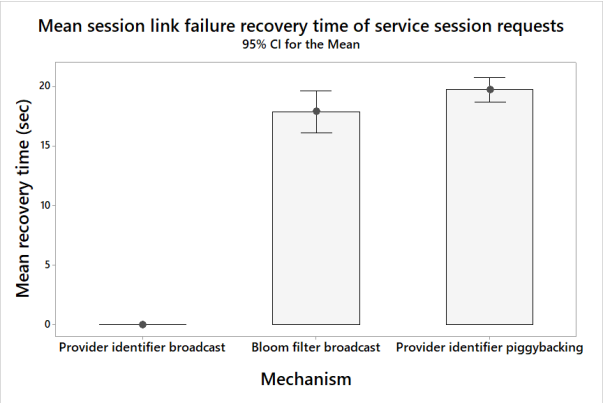

Figure 9: Mean session link failure recovery time of all service session requests for the three mechanisms.

B. Results

Fig. 7 compares the three mechanisms and the state-of-the-art session mechanism [3] that does not support session recovery. The mechanisms are evaluated in terms of successfully transmitted session Interests against the total number of session Interests sent in the network. Please notice that Interest messages are used to start, use, and terminate sessions. Circles in Fig. 7 display the mean number of successfully exchanged session messages. The circles are accompanied with respective 95% confidence intervals for different numbers of Interest messages sent (c.f., x-axis). The provider identifier mechanism delivers all service session messages because the provider identifier is broadcast at the beginning of the evaluation execution. Network participants can save and use the provider identifier for future forwarding decisions. The Bloom filter broadcast and provider identifier piggybacking strategies do not deliver all session messages. This is due to the fact that the information about alternative paths is not immediately available after a link failure. The confidence intervals for Bloom filter and provider identifier broadcast mechanisms overlap for different numbers of session messages sent. This suggests that there is no significant difference between the mean delivered session messages for these two mechanisms. The overlap can be explained by the delayed propagation of routing information in the network. Moreover, all our newly developed strategies clearly surpass the state-of-the-art mechanism [3], which is not equipped with any session recovery mechanism. The previously introduced session support [3] satisfies around 50% of Interest requests (c.f., the ratio between the number of messages delivered and messages sent for the state-of-the-art strategy [3] in Fig. 7). The newly delivered strategies display much better values with the provider identifier broadcast mechanism reaching 100% for all experiments performed.

Fig. 8 shows the normalized message overhead (NMO) for the three fault-tolerant mechanisms. The message overhead consists of messages sent by the service provider to inform the network about the existing session or provider identifiers, which enable us to use alternative paths. To draw a fair comparison, we compute the normalized message overhead (NMO) by using the formula NMO = A/B, where A is the number of transmissions concerning the propagation of session and provider identifier information for a given evaluation setup, and B is the number of distinct nodes that have received the message. Therefore, the NMO cost is expressed in overhead messages per node. Fig. 8 shows the NMO for the three mechanisms for the different number of service Interests sent. The provider identifier broadcast mechanism has a normalized message overhead of 1.77 and offers a successful delivery of all session messages. The identifier is broadcast at the beginning. Later on, when a link failure occurs, all the messages are redirected successfully by using the provider identifier. The Bloom filter broadcast and piggybacking strategies deliver successfully a similar number of service messages; some messages are lost since there is a delay between the session establishment and the session id or provider identifier propagation. The Bloom filter strategy has the highest overhead, because it periodically broadcasts Bloom filters in the case a new session has been created since the last broadcast. The message overhead of the Bloom filter strategy grows linearly in relation to the number of requests, because of its periodically broadcasting nature. The provider identifier piggybacking mechanism has no broadcast and reaches a similar ratio of successfully delivered session messages as the Bloom filter-based mechanism. Fig. 9 shows the mean session recovery time with respective confidence intervals for our three mechanisms. Broadcasting of provider identifiers delivers the best recovery time, as in the case of a failure, the nodes can immediately switch to an alternative path. The remaining mechanisms suffer from an increased recovery time caused by periodical dissemination of session identifiers in broadcasting of session identifiers and poor network penetration of SSD messages piggybacked on regular interest messages in the piggybacking method.

V. Conclusions

According to our knowledge, we present the first fault-tolerant session management for SCN. The first of the three mechanisms uses Bloom filters for the propagation of session identifiers, the second mechanism relies on the propagation of provider identifiers, and the third strategy uses piggybacking propagating provider identifiers. We have evaluated the presented mechanisms in terms of successful session message forwarding and message overhead in an environment with failing links. However, the solution could also be applicable for load balancing in the SCN network. The Bloom filter forwarding strategy does not rely on provider identifiers, which is an advantage if one does not want to depend on address-based forwarding. However, it displays the highest message overhead among the three mechanisms. The second strategy introduces provider identifier broadcast and has the best results in terms of successful forwarding. The third strategy relies on piggybacking; it does not have any message overhead, but it suffers from a high number of lost messages compared to the provider identifier broadcast mechanism.

REFERENCES

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[2] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, "Service-centric networking," in *2011 IEEE International Conference on Communications Workshops (ICC)*, June 2011, pp. 1–6.

[3] M. Gasparian, G. Corsini, T. Braun, E. Schiller, and J. Saltarin, "Session support for SCN," in *2017 IFIP Networking Conference, IFIP Networking 2017 and Workshops, Stockholm, Sweden, June 12-16, 2017*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.23919/IFIPNetworking.2017.8264879

[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[5] S. Shanbhag, N. Schwan, I. Rimac, and M. Varvello, "Soccer: Services over content-centric routing," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 62–67. [Online]. Available: http://doi.acm.org/10.1145/2018584.2018600

[6] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Scituate, MA, USA: Bradford Company, 2004.

[7] D. Mansour, T. Braun, and C. Anastasiades, "Nextserve framework: Supporting services over content-centric networking," in *Wired/Wireless Internet Communications*, A. Mellouk, S. Fowler, S. Hoceini, and B. Daachi, Eds. Cham: Springer International Publishing, 2014, pp. 189–199.

[8] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, "Serval: An end-host stack for service-centric networking," in *NSDI*. USENIX Association, 2012, pp. 85–98.

[9] Q. Wu, Z. Li, J. Zhou, H. Jiang, Z. Hu, Y. Liu, and G. Xie, "SOFIA: toward service-oriented information centric networking," *IEEE Network*, vol. 28, no. 3, pp. 12–18, 2014.

[10] C. F. Tschudin and M. Sifalakis, "Named functions for media delivery orchestration," in *20th International Packet Video Workshop, PV 2013, San Jose, CA, USA, December 12-13, 2013*, 2013, pp. 1–8. [Online]. Available: https://doi.org/10.1109/PV.2013.6691449

[11] S. Srinivasan, A. Singh, D. Batni, J. W. Lee, H. Schulzrinne, V. Hilt, and G. Kunzmann, "Ccnxserv: Dynamic service scalability in information-centric networks," in *ICC*. IEEE, 2012, pp. 2617–2622.

[12] M. Gasparian, T. Braun, and E. Schiller, "L-SCN: layered SCN architecture with supernodes and bloom filters," in *CCNC*. IEEE, 2017, pp. 899–904.

[13] A. Marandi, T. Braun, K. Salamatian, and N. Thomos, "BFR: A bloom filter-based routing approach for information-centric networks," in *2017 IFIP Networking Conference, IFIP Networking 2017 and Workshops, Stockholm, Sweden, June 12-16, 2017*, 2017, pp. 1–9. [Online]. Available: https://doi.org/10.23919/IFIPNetworking.2017.8264842

[14] ——, "A comparative analysis of bloom filter-based routing protocols for information-centric networks," in *2018 IEEE Symposium on Computers and Communications, ISCC 2018, Natal, Brazil, June 25-28, 2018*, 2018, pp. 255–261. [Online]. Available: https://doi.org/10.1109/ISCC.2018.8538676

[15] ——, "Pull-based bloom filter-based routing for information-centric networks," in *2018 IEEE Consumer Communications and Networking Conference, CCNC 2019, Las Vegas, USA, 11-14 January, 2019*, 2019, pp. 1–6.

[16] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN '14. New York, NY, USA: ACM, 2014, pp. 27–36. [Online]. Available: http://doi.acm.org/10.1145/2660129.2660140

[17] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," Tech. Rep., Jan. 2015.