Service-Centric Networking Extensions

Torsten Braun
Universität Bern
IAM, Neubrückstrasse 10
3012 Bern, Switzerland
+41 31 631 4994
braun@iam.unibe.ch

Andreas Mauthe
Lancaster University
InfoLab21
Lancaster, LA1 4WA, UK
+44 1524 510485
andreas@comp.lancs.ac.uk

Vasilios Siris
Athens U of Economics and Business
Department of Informatics
Patission 76, 104 34 Athens, Greece
+30 210 8203 581
vsiris@aueb.gr

ABSTRACT
This paper discusses several issues of Service-Centric Networking (SCN) as an extension of the Information-Centric Networking (ICN) paradigm. SCN allows extended caching, where not exactly the same content as requested can be read from caches, but similar content can be used to produce the content requested, e.g., by filtering or transcoding. We discuss the issue of naming and routing for general dynamic services for both tightly coupled and decoupled ICN approaches. Challenges and solutions for service management are identified, in particular for composed services, which allow distributed in-network processing of service requests. We introduce the term Software-Defined Service-Centric Networking as an extension of Software-Defined Networking. A prototype implementation for SCN proofs its validity and feasibility and underlines its potential benefits.

Categories and Subject Descriptors
C.2.1 Network Architecture and Design C.2.2 Network Protocols C.2.6 Internetworking

General Terms
Management, Performance, Design, Reliability, Experimentation.

Keywords
Future Internet, Information-Centric Networking, Services

1. INTRODUCTION
Information-Centric Networking (ICN, [11]) such as Content-Centric Networking (CCN, [4]) has been discussed intensively as a novel paradigm for the Future Internet. ICN has strong focus on supporting efficient content transfer in the Internet by using novel addressing and forwarding schemes. We argue that such novel paradigm must also consider how to support services in general, where service requests are issued by (mobile) users and trigger processing in service elements as well as responses to those requests. In particular, with the recent advent and growth of cloud computing it is necessary to support dynamic services in any Future Internet paradigm. We propose the paradigm of Service-Centric Networking (SCN), which makes use of and extends ICN forwarding concepts. We argue that extended caching mechanisms are needed for multimedia services and we discuss the requirements for management of services in a SCN environment. Further issues that need special attention are naming and routing as well as service management. Finally, we discuss possible implementation architectures for the service-centric networking approach.

1.1 Service-Centric Networking
SCN proposes to extend ICN by effectively supporting service requests in addition to content requests. Service and content requests are handled in a similar manner, i.e., a service or content lookup is integrated with packet forwarding to a certain degree. The idea is to avoid the traditional strong separation of service lookup/discovery and subsequently forwarding service requests to a specific server that has been identified a priori. With SCN a service user just sends a service request with an identifier describing the service in the address of the packet to the network. The service request will then be delivered to an entity that is able to optimally serve the service request.

Services are software elements located throughout the networked infrastructure and are hosted on dedicated hardware placed alongside the routing infrastructure. They are provided by active components, called *service elements*. This reflects the increasing importance of cloud computing but it has an even wider scope. We propose to have processing entities inside the network to efficiently support advanced services in the Internet. Figure 1 shows a service-centric network architecture with SCN routers (squares) and service elements (circles). Service elements are providing the service and can be hosted by an SCN router, by servers co-located to SCN routers, or by end systems attached to the network.

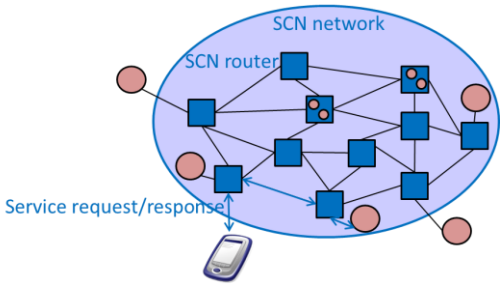

Figure 1: Service-Centric Network Architecture

1.2 Target Services
The services within an SCN can be of different types, depending on their scope and level within the system architecture. This paper is motivated by services related to multimedia distribution. How-

ever, this does not restrict the generality of the basic principle. Other services can be grouped as follows:

Infrastructure services provide service users with computing, storage, and/or communication resources to deploy and run own software and services. The concepts described as Infrastructure / Platform as a Service in cloud computing belong to this category [5]. DC-ICN proposes using ICN in data centres [16].

Client-oriented services provide application level services. Requests sent to servers are served directly with an immediate, single response. Web services and Software as a Service as defined in the cloud computing terminology are such examples. Services are deployed by service providers such as network operators, third-party providers, or end users.

Continuous content retrieval and streaming services are used to optimise content distribution. Here, a single request triggers continuous subsequent content transmission such as stored/live audio/video streaming. This also includes the use of in-network services such as transcoding, mixing, filtering, etc. Filtering and aggregation services deployed inside the network can reduce data traffic for continuous sensor data streams.

Event services are based on the detection of unusual events such as exceeding sensor data thresholds or exceeding stock exchange prices. In case of exceeding threshold, an Interest message can describe the interest of a user in becoming notified.

SCN aims to optimize the performance (in particular delay) of future network applications and services and utilize network resources more efficiently, e.g., by reservation of resources and deployment of services close to (mobile) users. This might be important in a mobile network environment with an increasing number of context-dependent (e.g., location) and personalized services. However, SCN leaves open several research issues. These are discussed in the following sections.

After discussing related work to SCN in Section 2, we introduce the concept of extended caching, which is enabled by SCN, in Section 3. Service naming and routing are discussed in Section 4. Service management, i.e. creation, adaptation, and termination of distributed in-network services, is considered in Section 5. Section 6 describes a general SCN implementation architecture as well as a prototype implementation used for preliminary SCN evaluations. Section 7 concludes the paper.

2. RELATED WORK

2.1 Web Services

Web Services provide programming interfaces, which are accessed via the HyperText Transfer Protocol (HTTP) and executed on a remote system hosting the requested services. Web Services are described using the Web Services Description Language. In addition to in-bound services, out-bound services allow the service to send the first message to a client. Web Services focus on providing operations using Simple Object Access Protocol (SOAP) as a protocol for exchanging structured information. SOAP relies on the eXtensible Markup Language for its message format, and usually relies on other application layer protocols such as Remote Procedure Call and HTTP for message negotiation and transmission. Representational State Transfer (REST) is a style of software architecture for web services based on HTTP. In contrast to SOAP, no additional server-side operations are introduced, but only HTTP methods (e.g., POST, GET, PUT or DELETE) are used. REST is completely stateless at the server and supports caching. Certain multimedia services are rather difficult to be implemented inside the network using REST.

2.2 Content Distribution Networks

The Internet nowadays relies on the usage of caching mechanisms or content distribution servers. Content Distribution Networks (CDNs) support several well-known services on the Internet. Users connect their end systems to residential gateways or directly to Internet Service Provider routers, relying on pre-existing configurations that allow access to static content available on caching servers or CDNs. As an alternative solution, peer-to-peer overlays are also becoming more widely available. However, these networks are particularly inefficient in situations, where content sharing nodes are isolated from each other. In a future Internet user experience could strongly benefit from changes in the paradigm of content distribution. Such changes could include caching contents closer to end users; dynamic content interactively or automatically adapted according to the needs of users and terminal nodes (not only considering the network structure); content aware routers and optimized routing paths (according to bandwidth, latency, etc.).

2.3 Active Networking

Active and programmable networks have been devised to provide means of flexible and dynamic data path processing capabilities that allow network und user side services inside the network [19]. The processing capabilities of active networking nodes can provide low-level packet processing and per user or application processing of data streams. Research in active networks has resulted in the (further) development of areas such as active node platforms [19][20], service composition models [21], mobile coding techniques [22], execution environments [23], etc.

To be able to provide such functionality network nodes (such as switches and routers) have to provide an advanced execution environment to turn network nodes in to active programmable elements. With the so-called *programmable switch* programs are downloaded without changing the packet format. Hence, the network providers can maintain control of the network operation by only allowing specific programs to be executed. A more radical concept is a *capsule* approach where "active" programs are part of individual packet and flows. These programs are then executed in the appropriate network elements along the data path.

Active networking has never been widely deployed. One major reason for that were security concerns and possibly extensive resource consumption dependent on program functionality.

2.4 Information-Centric Networking

Different ICN proposals involve a different degree of coupling between name resolution and routing/forwarding. At one extreme, the same nodes perform both functions in a ***tightly coupled*** manner. This is the approach followed by CCN/NDN [4]: Receivers express their request for content using Interest packets. Interest packets are routed based on the name of the requested content, using longest prefix matching, either to the node that contains a data packet with the requested name or to an intermediate node that has cached the requested content. Once the data packet is found, it is returned to the requester following the reverse path of the Interest packet.

At the other extreme, name resolution and routing/forwarding are implemented in different nodes and/or different modules (***decoupled*** approach). This is the approach followed by architectures such as PSIRP/PURSUIT's PSI (Publish-Subscribe Internet) [7] and 4ward/SAIL's NetInf architecture [11][8]. NetInf uses a two-phase name resolution approach based on registration of network locators at a name resolution service [6]. The name resolution system is independent of the routing/forwarding network that transports content from the publisher to the subscriber.

In PURSUIT/PSIRP, publishers advertise the content objects (publications) that they make available. A publication belongs to a particular named scope. Receivers can subscribe to content objects. Publications and subscriptions are matched by a rendezvous system. The subscription request specifies the scope identifier and the rendezvous identifier that together name the desired content object. PURSUIT enforces uniform naming by unique scope and rendezvous IDs (SID, RID). RIDs are application dependent identifiers unique within the scope of an SID.

Proposals such as DONA [9] and COMET [10] investigate overlay solutions running on top of an IP infrastructure, hence use IP routing and forwarding functionality inherently. Juno [12] is a content-centric middleware based on self-certifying content identifiers. It includes a content discovery service, by which content is resolved similarly as in NetInf and DONA.

2.5 Service-Centric Networking

In recent years, a few approaches targeting service support by the network infrastructure have been published.

SCAFFOLD [14] (Service-Centric Architecture For Flexible Object Localization and Distribution) is an architecture that provides a flow-based anycast service with possibly moving service instances. SCAFFOLD aims to manage on system churn, i.e., some change of physical system or network resources, e.g., from failures, planned maintenance, load balancing, workload migration, or physical mobility. SCAFFOLD depends on underlying virtualized and programmable network elements to support transparent migration of objects, services, or virtual machines across physical resources. SCAFFOLD directly addresses distributed or replicated objects or services, rather than hosts.

SERVAL [13] is a service access layer above an unmodified network layer enabling applications to communicate directly on service names. SERVAL separates service ID, flow ID (socket), and network address (interface). Service IDs consist of 256 bits to be administered by a central authority. All IDs are visible to service routers. Forwarding is based on rules defined in tables. Possible actions are forwarding, demultiplexing, delaying, dropping if packets. SERVAL provides an anycast service for service routing with late binding of connections and services. Multi-path communication can be used for striping of connections/flows.

3. EXTENDED CACHING FOR CONTINUOUS CONTENT RETRIEVAL AND STREAMING

Like content delivery networks, a major motivation for introducing ICN [4] has been the reduction of traffic and delays by exploiting caching. In contrast to CDNs, ICN combines location-independent content lookup with forwarding/routing and router (in-network) caches. Not only server caches as in CDN can be used: A user request for accessing content could be served by an ICN router cache instead of a server cache. Caching with services is more difficult and possibly somewhat less likely to occur, since service requests may be individual and context-dependent. Nevertheless, we think that in particular (personalized) multimedia streaming applications can significantly benefit from services deployed inside the network, in particular from so-called ***extended caching***. The underlying principle of extended caching is that of "content as an object", i.e., a content object is not identical with a specific copy but is a representation of a content element that can exist in different formats, copies and at different locations. Aligned with the idea of ICN it is assumed that the user is not interested in the exact copy of the content object but the content itself. In addition to this, the user is ultimately agnostic as far as delivery is concerned (which leads to delivery-centric content networking [17]). To optimise delivery with respect to network and device conditions the notion of content centricity and delivery centricity can, therefore, be extended to also provide adapted content according to device, network and user context.

Traditional caching is based on repeatedly requesting identical content and serving such requests from a cache. Extended caching does not necessarily deliver the original content, but the content might just be similar to the original one. As example, requests can be served by taking content objects from the cache, transcoding, filtering, aggregating or extracting the content to be delivered to the requesting users. In both cases, traditional and extended caching, cached objects are (re-)presented in the cache. Though, in extended caching, not only the exact copy of the object (i.e., the content object) can be delivered if requested, but it is also sufficient that if the content is available in a specific form to use this to generate the desired format.

Extended caching could be applied for streaming services, when segments of a requested video stream are available in the cache, but the request demands a different encoding than what is in the cache. The request can then be served after transcoding the available cached content into the requested format. Services that can benefit from extended caching are streaming applications tailored to certain device capabilities, interconnectivity conditions or user context information (e.g., location of a user close to multimedia output devices), news streams composed of contents elements based on personal interests and preferences, on-line games with real-time audio/video etc. As an example, different users might request a streamed video tailored to their viewing device capabilities. The request might meet a service element, where parts of the requested stream are stored, but not in the same format as requested. If the service element provides some transcoding facilities, the request can be served without forwarding it to the original server. A similar example is a situation where the requesting device is connected via a path with impaired conditions such as higher delay or loss or limited bandwidth. Redundant encoding schemes or multi-path transmission might provide the required robustness. This again can be implemented by supporting encoding and transmission services inside the network.

Caching can be implemented in a passive or active way. In case of passive caching, content forwarded along a path of routers can be cached in the router memories. In case of active caching, content can be proactively placed at routers (or attached servers) to better support the target users and to increase quality of availability (QoA) [18]. This placement has a geographical as well as a temporal aspect. Similarly as content, also service entities have to be placed within the network, cf. Section 5.1.

4. SERVICE NAMING AND ROUTING

SCN proposes to use both service and content names as identifiers for Future Internet routing. This creates several challenges to be investigated as discussed in the following subsections.

4.1 Service Naming

A first issue to be considered is name resolution for services. Different services should be distinguishable by different names. Identical services or service classes should have identical names to support location-based services close to a user. Several mechanisms for uniform content naming exist such as the Digital Object Identifiers for digitally available scientific articles or magnet links as used in peer-to-peer networks, which are built by hashing the content. While a completely flat name space does not guarantee uniform naming, hierarchical approaches similar to domain names require strong coordination similar to the domain name system (DNS), which is also not desirable for content producers. Therefore, schemes combining both approaches by splitting the content name into two parts <content_owner, content_name> might be an interesting alternative. This allows both uniform naming as well as flexible naming of contents. This could also work for naming of services, although it might be possible that users do not care about or do not know the service provider in advance. Therefore, wild cards, e.g., <*, service_name>, should be supported. Hierarchical service names could then guarantee uniformity of service names.

4.2 Service Name Resolution

Content / service names can either be directly used for routing (tightly coupled approach), e.g., as proposed in CCN, or the content / service name can be mapped to a locator identifier (e.g., using a hash function), which is then used for routing messages to the content / service source (decoupled approach). In case of a decoupled approach similar content / service names may be mapped to completely different locators, which would result in providing similar services at rather different locations in the network. This adds certain flexibility in selecting service entities for a requested service name.

Tightly coupled approaches only scale, if names can be aggregated. In that case, classes of services should have common prefixes to allow aggregation of services but also to simplify searching. This can be achieved by classification of services and building a hierarchy of services as proposed in [1]. The service name could then be built as a concatenation of service categories. A single-level web service directory is available at [2]. This would also allow having different service providers for the same service name. On the other hand, it requires strong coordination of the naming hierarchy, similar as for the domain name system (DNS).

Another option in tightly-coupled approaches such as CCN is to use an overlay of service nodes. Service nodes are interconnected by normal CCN routers. This can be achieved by adding a prefix to the service name containing the name of the service node that will next process the service request. In such an approach there are two types of nodes: normal CCN routers and service (SCN) nodes that do more processing of service requests than CCN routers do. This can be regarded as creating CCN tunnels between SCN nodes.

When implementing SCN above an ICN architecture that decouples name resolution and data transport (***decoupled approach***), advanced and more complex processing of service requests can be performed by rendezvous nodes. Implementing SCN over a decoupled approach such as PURSUIT has benefits such as separate control of the route for data transfer, different routes for control and data packets and more effective many-to-one communication. A decoupled approach has the advantage of incremental deployment in the Internet, but adds additional delay due to the two consecutive phases name resolution and routing.

In certain special cases, in particular when hashing is used for mapping names to locators, similar content or services, e.g., different versions, will be mapped to different locators. Thus, service requests might be forwarded along a completely different path, which can reduce benefits obtained from extended caching. This might also be an issue for video retrieval, when two files /owner/video_name.mpeg and /owner/video_name.avi are mapped to different locators. A user might first request the MPEG file and a second user, which is close to the first user, might then request the AVI file. While for two-phase name resolution, the two requests (including name resolution and forwarding of content/service requests) might travel along two different paths, in case of single phase name resolution, there is a high chance that the requests from both users travel along the same path. The request of the second user for the AVI file can then be served by a router that already has cached the MPEG file by transcoding the MPEG file into AVI format.

4.3 Service Parameter and Type Support

In addition to the service name, service parameters must be described for certain service requests. On the one hand, some services will require support for different input or output formats and hence will need only a limited, fixed set of parameters. As an example, different streaming services might exist for different screen sizes or different encoding formats. On the other hand there are services that need to support different numbers and types of parameters; e.g., a location-based service to find the best hotel could consider different combinations of parameters such as price, current distance to querying user, category etc. These examples show that routing a service request might not only depend on the service name but also on additional parameters describing the service request.

For a ***decoupled approach*** such as PURSUIT, rich service descriptions can be supported in PURSUIT rendezvous nodes, which are responsible for mapping service requests to service publishers. An appropriate forwarding path can then be set up between publisher and subscriber.

For ***coupled approaches*** such as CCN, we see two options for describing service parameters, either as an extension of the service name or as additional information in the body of the service request message.

1. Coding service parameters or parameter types as part of the service name require very efficient coding. A possible solution are type representations as in ASN.1 Basic Encoding Rules, but use short encodings for frequently used types, e.g., based on Huffman or arithmetic encoding. If types and parameters should be part of the service name, the encoding must be mapped to ASCII characters, e.g., using base64 encoding. This is mainly appropriate for services with a relatively limited, well defined and static set of parameters.
2. In cases where a service requires a parameter list that is arbitrarily long and nested and might need dynamic parame-

ters, it makes hardly sense to put anything else than the service name into the address of the service request. Parameter values and types should then rather be described in the body of a service request message, e.g., using XML. If parameter values or types need to be analysed for routing a service request, this would then require analysing the message body. Alternatively, if multiple services exist under the same name, but analysing different parameter sets and types and analysis of the message body is not feasible in routers, the SCN network should forward service requests to all potentially relevant service entities and let these finally check whether they can support the service request or not. Also, initially a "plain" service request might be forwarded in the network and through a handshake process the actual parameterisation is then negotiated.

4.4 Service Routing

By using name-based addresses CCN [4] routing can be considered a superset of IP routing. If a routing entry in the Forwarding Information Base (FIB) is present, e.g., for "/content-provider/path-name", there is a high chance but no 100 % reliability that the requested content will be reached via the interface specified in the routing table. This means that a content/service request (also known as Interest message) should be forwarded along multiple paths to increase the probability to find content. If no matching FIB entries for a requested content object can be found, then an Interest message might have to be flooded or distributed via several paths. Forwarding Interests over multiple paths can also be used to find the closest copy of the content object stored in a cache. Strategies to forward Interest messages are left open [4], but it has been proposed to populate FIBs by announcements of available content. Such announcements can avoid extensive flooding. Similar mechanisms, i.e., announcement of available services could be used by SCN. SCN service routing should make use of the significant amount of work on service discovery that has been developed in peer-to-peer and mobile ad-hoc networks. Techniques such as Distributed Hash Tables (DHT), random walks to avoid flooding, extended ring search, Bloom filters can be considered.

Routing metrics are another issue related to service routing. It might be reasonable to consider performance issues in addition to address information when deciding along which path a service request message should be forwarded. This is the case, when multiple service elements exist for the same service, possibly provided by different service providers. Another potential use case for multiple service elements is automatic load balancing.

Routing metrics can be supported by adding more columns for each address entry into the routing table. As example, a routing table column entry could contain the estimated response time for a service, i.e., the time between forwarding a service request packet and receiving the service response. Round-trip time measurements and exponential averaging techniques could be applied. The routing entry with the best value (e.g., lowest measured response time) can be used to forward a service request. However, it must be ensured that metrics for alternative routes are refreshed periodically, e.g., by periodically using alternative routes for service requests, or by generating explicit probing messages. Another option is probabilistic routing by defining a probability function dependent on the measured metric values. This approach can also nicely be applied to load balancing. Overloaded service elements with increased response time will then be less likely selected. SoCCeR [15] adds a control layer on top of CCN for the manipulation of the underlying CCN Forwarding Information Base. Routing of Interests is performed based on the measured load and expected delay based on ants mechanisms. SOCCER adds overhead by exchange of ants in addition to Interests and Data.

For decoupled ICN approaches, routing of search or subscription information for name resolution is decoupled from data transfer. DHTs can be used to find responsible rendezvous nodes for (SID, RID) pairs in PURSUIT. After finding (or even selecting from multiple options) a rendezvous node responsible for a service identified by a (SID, RID) pair, a router between the service provider (publisher) and service user (subscriber) can be set up. cf. Section 5.2. While metrics for routing can be determined in straight-forward way for coupled approaches like CCN, due to the decoupling of forwarding and name resolution this is rather an open issue for decoupled approaches. Metrics should be determined based on round-trip times during the data forwarding phase, but should be considered for name resolution when determining appropriate service entities.

5. SERVICE MANAGEMENT AND DELIVERY

5.1 Service Management

A major issue of SCN is where and when to deploy which services in the network. Supporting multimedia streams by servers at the edge of a network only is not most efficient. Even for personalized streams, extended in-network caching and transcoding should be utilized to provide a better service. In addition to deployment, services need to be adapted and removed when not needed anymore. Deployment, adaptation, and termination of services should be considered as parts of service management.

Service deployment can either be triggered by users, network operators or agents in network elements when through specific user content, user service, or content service interaction it is discovered that a service deployment for a specific case is beneficial for performance. This requires a measurement infrastructure to detect which data flows between service providers and service consumers exist. This knowledge can then be taken as input by a decision making entity to decide when and where to dynamically deploy new service elements. In addition to the temporal deployment decision of a new service element, the decision where to deploy is also important. For example, a transcoding service might be best deployed at a forking point where different networks (and specifically network types) converge. There are certain criteria and parameters that have to be taken into account. Service deployment must consider the hardware capabilities required to perform expensive calculations in order to implement the service. Service elements might also have to access remote data. The overhead for accessing remote data can be reduced by an appropriate choice for the location of the service element to be deployed. Optimal service deployment is challenging. Algorithms based on heuristics might be needed.

Since several parameters such as usage patterns, mobility of users, and network conditions are highly dynamic, parameters and location of service elements need to adapt to current conditions. This might result in modified/adapted functionality of service elements or in migration of service elements. **Service adaptation** again requires monitoring tools to record traffic and service usage. Ad-

aptation of services should be supported by self-adaptive service elements. However, self-adaptivity of services might be limited and for significant modifications, e.g., replacement of service elements, operator-assisted adaptation might be needed.

Services can have different lifetimes depending on their purpose, task and goals. Certain services can be permanently installed and make up part of the "service fabric". Others will only be required at a certain point in time to serve a specific service request. Services can also be required at a specific location for a predefined duration and then they should be removed. Hence, instantiated services should be removed when not needed anymore, e.g., in case the service task has been fulfilled or of detected low service usage (**service termination**).

An appropriate service management framework possibly based on policies and autonomic network management mechanisms must be identified. The framework must comprise mechanisms for monitoring, resource optimization, decision support, and control.

Service management in tightly coupled approaches such as CCN includes deployment of services at service entities such as routers or attached servers. Routing tables must be adapted accordingly. Service management using a decoupled approach such as PURSUIT requires announcement and termination of services at rendezvous nodes. Adaptation is a special case of terminating the previous service and announcing the adapted service.

5.2 Service Delivery

In coupled approaches as CCN, service delivery is based on the proposed Interest/Data message exchange, which requires having one Interest message for each Data message. Another option is to have a value, e.g., N, in each interest message indicating after how many Data messages an Interest can be discarded in a router. This would replace subsequent transmission of N Interest messages. Alternatively, it might also make sense to use Interest/Data message exchange only for service discovery and establish a fixed path between client and service, e.g., using OpenFlow. The path is then determined by the path taken by Interest/Date messages. This makes in particular sense for stateful services. In decoupled approaches service/data delivery is independent of name resolution. After name resolution any kind of data path can be selected or established for data/service delivery. This could be even multipath or multicast communications.

5.3 Software-Defined Service-Centric Networking

Consider that an extended service and object name in SCN can define multiple functions to be performed on multiple objects involving multiple service nodes. Then, a service requested by a user can be composed out of a set of basic services already deployed inside the network. Hence, a service request can be seen as a service "program", which defines service composition involving objects or methods from different servers. Indeed, this promotes the service-centric and location-independent trend: the service program does not indicate the specific servers that need to be involved, but rather only the service(s) the requester is interested in. It is up to the network to identify the specific servers that will provide the service. This might have similarities with active networks, but it is based on ideas such as location-independence of service and data and routing based on names and addresses.

A service mapper has to react on a service request for the composed service, identify the necessary basic services, call these by issuing single service requests, and combine the results into a single response for the composed service request. Service nodes with mapping functions should run at strategic points inside the network so that other basic services are reachable with low overhead in terms of delay and bandwidth usage. Otherwise, if the client knows the decomposition of a service, a list of services to be combined in a chain can be provided by the client by making use of a SCN routing header. To support service composition, a service program can be routed between service nodes using CCN named-based routing. Between two service nodes there can be normal CCN forwarding/routing nodes that forward based on names. Once the service program reaches a service node, the actions in the service program can involve sending the service request to a specific service node; the latter can be achieved by adding the specific service node address (locator). The next service node that follows can depend on the results of functions that are executed in the current node.

Service composition can be supported by software development tools such as G-Streamer [3], which is an open source multimedia framework for developing audio/video applications out of a set of already existing basic components. Service composition can be supported by in-network service nodes that receive composite service descriptions and break down / map the composite service descriptions into service components or basic service elements. As example, a multi-party conference service can be broken down into basic service elements such as encoding a participant's audio, transcoding it into a common encoding, mixing the single participant's audio streams, and transcoding the mixed audio stream to the encoding that can be decoded by each single participant. In addition to the multi-party conference service, possible other applications benefitting from service composition are distributed services such as top headline news from major news sites or distributed search. These example services involve functions from possibly multiple servers. However, the fact that the service is implemented by multiple servers is not exposed to the requester or indicated in the request itself, which is location independent.

It should also be emphasized that here is a certain analogy between the above software-defined view of SCN and Software-Defined Networking such as OpenFlow. OpenFlow separates control from forwarding functionality of a node. Forwarding nodes expose an API for controlling its operation. In a similar way, a service-centric networking architecture can be seen to provide an API to service developers (distributed application developers) to define services (which can involve combinations of multiple functions on multiple objects). Thus, the SCN approach can be considered and called Software-Defined Service-Centric Networking (SDSN). Similarly as SDN, SDSN can provide API to service developers (distributed application developers) to define services (which can involve combination of multiple functions on multiple objects). SDSN acts on the service level and programs the whole network to provide a service, whereas OpenFlow acts on the network level and controls operation of a forwarding node.

6. SCN PROTOTYPE IMPLEMENTATION

There are a variety of alternatives to implement Service-Centric Networking. Service elements could be implemented inside routers, similar to concepts proposed by Active Networking. Due to

the risks and concerns about Active Networking we rather see more promising options by deploying execution environments co-located with routers alongside the delivery network. This is in line with emerging cloud computing environments and data centres that are directly connected to routers or deployed somewhere else in the network. However, in contrast to cloud services, other SCN target services might be more dynamically located throughout the SCN infrastructure. It is also possible to have service elements deployed in end systems, i.e., servers of service providers or even end user devices.

6.1 Example: Image Conversion Service

As an example for demonstrating the benefits of SCN we take a still image conversion service. This service can be deployed at a CCN router. Assuming that a still image is available from a publisher as bit map (BMP) or JPEG file, a user might request a JPEG encoded file of the still image. If a CCN router in between the user and the publisher has already stored the bit map file in its content store (cache) or in its file repository, the bit map file can be converted into a JPEG file by the CCN router and delivered to the user. This saves bandwidth between the CCN router and the publisher, who might also have stored the JPEG file of the still image. However, this comes at the cost of additional processing in the CCN router for the image conversion. This adds additional delay, which might be higher than the end-to-end delay between user and publisher.

6.2 SCN Implementation

As a proof of concept, we implemented the still image conversion service on top of CCNx [24]. This allows several options for the implementation. One option would have been to extend the CCNx daemon called ccnd, which runs in Linux user space. This would allow implementing a conversion service transparent to the user. In this case, ccnd could intercept requests for a bit map file, convert the bit map file into a JPEG file and return the JPEG file to the user. This approach would require modifications of ccnd. Moreover, ccnd would have to decide whether and from which source file the conversion should be done. Due to these drawbacks we decided not to follow the above approach. Instead we implemented the image conversion service as a separate application running on a CCN router. In this case, the image conversion service provided has to be addressed explicitly by the user. As example, if a user wants to use the service an Interest message as request has to be sent to the CCN network:

`/unibe.ch/sc/JPEG/publisher.org/image.BMP`

The request asks a CCNx conversion service provided by University of Bern (unibe.ch) to retrieve the original image with the name `/publisher.org/image.BMP`, convert it into a JPEG file and return it to the user. This approach does not require changes to ccnd. The user can completely control which file should be converted into which format. The service is addressed explicitly. The prototype implementation has been built by using the CCNx repository at the publisher side and the ccngetfile program at the user side. These programs are provided by the CCNx implementation.

6.3 Service Usage Example

Figure 2 shows one example for service processing with the image conversion service. Other modes are possible as well.

1. User1 at client1 issues a request for a bit map file from the publisher.
2. The request, i.e. Interest message, will arrive at the ccnd of the CCN router (ccndr).
3. The Interest message is forwarded by ccndr to ccndp, the ccnd of the publisher.
4. The bit map file will be read.
5. The bit map file arrives at the ccnd of the publisher's server.
6. The bit map file is sent in data messages to the ccnd of the CCN router.
7. The bit map file is sent back to ccnd1, the ccnd of client 1.
8. The bit map file is delivered to user1
9. User2 issues a service request to convert the bit map file of the publisher into a JPEG file.
10. The service request is forwarded by ccnd2, the ccnd of client 2, to the CCN router.
11. ccndr, the ccnd running at the CCN forwards the service request to the service entity running on the CCN router.
12. The conversion service requests the bit map file as input for the conversion.
13. ccndr has the bit map file in its content store / cache and can deliver it to the conversion service for conversion into JPEG.
14. The JPEG file is delivered by the service entity to ccndr,
15. ccndr returns the JPEG file to ccnd2.
16. The JPEG file is delivered to user2.
17. User1 issues the same request as user2 before.
18. The service request arrives at ccndr and can be served from the content store / cache.
19. ccndr delivers the JPEG file to ccnd1.
20. The JPEG file is delivered to user1.

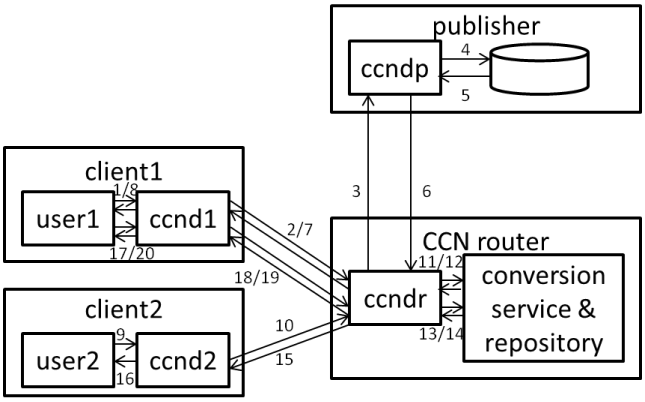

Figure 2: Service processing with Service-Centric Networking

6.4 Experimental Evaluation

The evaluation has been performed using 4 computers according to the example scenario depicted in Figure 2.

- First, we measured the time for the delivery of a bit map file from the publisher to user 1. This corresponds to steps 1-8. The time for the delivery of the 36 MB bit map file was 186 s. This time is needed for forwarding the Interest and Data messages between the involved CCN daemons ccnd1, ccndr, ccndp.
- Next, we measured the time for handling the service request of user2. This includes steps 9-16 and lasted 31 s. This mainly includes time for converting the bit map file into a

JPEG file (6 MB) by the CCN router as well as forwarding Interests and Data between ccnd2 and ccndr.

- Finally, we measured the time for the service request issued by user1. The service data is already in the CCN router's content store and the request can be completed after 4 s.

In the example used for performance evaluation, the amount of data exchanged over the network could be decreased significantly compared to a scenario without SCN support, where both clients would have requested a bit map file first and then converted it locally. Client 1 could have requested the bit map file from the publisher over two hops. Client 2 could then be served by the CCN router, but the full 36 MB file would have to be transferred between CCN router and client2. In the service-centric networking case, the 36 MB file was only transferred over one link, instead of three links in a pure CCN scenario without SCN support. However, the bandwidth savings come at the cost for additional delay caused by the conversion service. In our example the delay is rather significant due to the complexity of the service and the user space Java implementation of the service, which has not been optimized for performance. More appropriate programming languages, runtime environments, and better integration of service implementations and ICN routers can reduce this drawback.

7. CONCLUSIONS

This paper discussed certain issues of Service-Centric Networking such as naming and routing as well as management of services in more detail. In particular, it discussed differences of SCN when using tightly coupled or decoupled ICN approaches. As a proof-of-concept we have implemented a still image conversion service, which can significantly reduce network bandwidth.

8. REFERENCES

[1] K. Arabshian, H. Schulzrinne: An Ontology-based Hierarchical Peer-to-Peer Global Service Discovery System, Journal of Ubiquitous Computing and Intelligence, 2007

[2] http://www.programmableweb.com/apis/directory, accessed November 28, 2012

[3] http://www.gstreamer.net, accessed November 28, 2012

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, R. Braynard: Networking named content. Communications of the ACM, vol. 55, no. 1, 2012

[5] J. Walz, D. Grier:Time to Push the Cloud, IT Professional , vol.12, no.5, 2010

[6] B. Ahlgren, Ch. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman: A Survey of Information-Centric Networking, 2011, Dagstuhl Seminar Proceedings

[7] N. Fotiou, P. Nikander, D. Trossen, and G. Polyzos. Developing information networking further: From PSIRP to PURSUIT, 7th ICST Int'l Conf. on Broadband Commun., Networks, and Systems, 2010.

[8] B. Ahlgren: Content, Connectivity, and Cloud: Ingredients for the Network of the Future,IEEE Communications Magazine, vol. 49, no. 7, pp. 62–70, July 2011.

[9] B. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, I. Stoica: A data-oriented (and beyond) network architecture,"ACM SIGCOMM Computer Communication Review, vol. 37, no. 4, 2007.

[10] W. Koong Chai et al.: CURLING: Content-Ubiquitous Resolution and Delivery Infrastructure for Next-Generation Services, IEEE Communications Magazine, vol. 49, no. 3, 2011

[11] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman: A survey of information-centric networking, IEEE Communications Magazine, vol.50, no.7, 2012

[12] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, A. Mauthe: A survey of mobility in information-centric networks: challenges and research directions, 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications, 2012

[13] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Ko, J. Rexford, M. J. Freedman: Serval: An End-Host Stack for Service-Centric Networking, 9th USENIX Symposium on Networked Systems Design and Implementation, 2012

[14] M. J. Freedman, M. Arye, P. Gopalan, S. Y. Ko, E. Nordström, J. Rexford, D. Shue: Service-Centric Networking with SCAFFOLD, Princeton University, Computer Science, Technical Report TR-885-10, 2010.

[15] S. Shanbhag, N. Schwan, I. Rimac, M. Varvello: SoCCeR: services over content-centric routing, ACM SIGCOMM ICN workshop, 2011

[16] B. Jun Ko, V. Pappas, R. Raghavendra, Y. Song, R. B. Dilmaghani, K.W. Lee, D. Verma: An Information-Centric Architecture for Data Center Networks, ACM SIGCOMM ICN workshop, 2012

[17] G. Tyson, A. Mauthe, S. Kaune, P. Grace, A. Taweel and Th. Plagemann. Juno: A Middleware Platform for Supporting Delivery-Centric Applications. ACM Transactions on Internet Technology, 2012

[18] G. On, J. Schmitt, R. Steinmetz: Quality of Availability: Replica Placement for Widely Distributed Systems, in proceedings of the 11th international conference on Quality of service, IWQoS 2003

[19] M. Sifalakis, A. Mauthe, D. Hutchison: SAND: A Scalable, Distributed and Dynamic Active Network Directory Service, Proceedings of 7th Annual International Working Conference on Active and Programmable Networks, 2005

[20] Merugu, S., Bhattacharjee, S., Zegura, E., Calvert, K., Bowman: A Node OS for Active Networks, IEEE INFOCOMM, 2000.

[21] S. Schmid, J. Finney, A. Scott, W. Shepherd: Component-based Active Network Architecture, IEEE Symposium on Computers and Communications, July 2001.

[22] D. Wetherall, J. Guttag, T.Tennenhouse: ANTS: A toolkit for building and dynamically deploying network protocols, IEEE Openarch, April 1998.

[23] M. Hicks, P. Kaddar, J. Moore, C. Gunter, S. Nettles: PLAN: A Packet Language for Active Networks, In Proceedings of the 3rd ACM SIGPLAN International Conference on Functional Programming, pages 86-93, 1998.

[24] E. Cheriki: Design and Implementation of a Conversion Service for Content Centric Networks, Master thesis, FH Bern, 2012