Service-Centric Networking

Torsten Braun
Institut für Informatik und Angewandte Mathematik
Universität Bern
Bern, Switzerland
braun@iam.unibe.ch

Volker Hilt, Markus Hofmann, Ivica Rimac,
Moritz Steiner, Matteo Varvello
Bell Labs, Alcatel-Lucent
Holmdel, USA
firstname.lastname@alcatel-lucent.com

Abstract— **Content-centric networking is a novel paradigm for the Future Internet. This paper argues that content-centric networking should be generalized towards a service-centric networking scheme. We propose a service-centric networking design based on an object-oriented approach, in which content and services are considered objects. We show implementation architectures for example services and how these can benefit from service-centric networking.**

Keywords: Content-Centric Networking, Service-Centric Networking, Future Internet

I. INTRODUCTION

Content-centric networking (CCN) is a networking paradigm for the Future Internet, in which routing and forwarding are based on content identifiers rather than on host identifiers. The CCN approach provides several benefits over the existing network architecture: the network provides full content identifier lookup functionality; content can be cached anywhere along the delivery path allowing better network utilization; content itself can be secured instead of the end-to-end connection that carries it. While CCN strongly focuses on content retrieval, the Future Internet is expected to provide a more general support of services. Content delivery is merely one example of a service; other examples are content generation and manipulation as well as general processing services. This trend is also driven by cloud computing. In this paper, we propose a service-centric networking (SCN) scheme as an extension of CCN. In our design we consider both content and services as key design elements. SCN supports a variety of services including file storage and retrieval, audio/video streaming and recording, processing of stored images and video, on-line shopping, location-based services, cloud computing, and telecommunication services.

After discussing related work in Section II, Section III introduces the basic principle of SCN, illustrates its potential benefits and discusses some example applications. Section IV describes the object-oriented SCN rationale in detail. Section V describes how two example applications, namely video rendering and audio conferencing, can be implemented based on SCN. Finally, Section VI concludes the paper.

II. RELATED WORK

A. Content-Centric Networking

In their work on networking named content [1] the authors propose a content-centric architecture based on forwarding Interest messages towards content. Interest messages are disseminated along possible paths where the requested content might be located. When Interest messages reach a node holding the content, the latter is returned to the requestor along the reverse path of the Interest messages constructed via bread crumb routing. Since each Data message carries the name of the encapsulated object, content can be cached by any router on the delivery path and served from there upon the reception of an Interest message. CCNx™ is an open source project based on the above scheme and using the naming convention described in [2]. In the following, we use the term CCNx for the concept described in [1], since other content-centric networking approaches such as TRIAD [7] and DONA [6] exist. Consequently, we call an implementation of any CCN approach a CCN infrastructure.

In [2], the authors propose a scheme to encode markers to code functional components. These command markers with the value of 0xC1 followed by "." do usually not appear in names. Optionally, the operation name may be followed by arguments delimited by tilde characters "~". An operation name may be constructed using a reversed DNS name. As an example ***%C1.org.ccnx.frobnicate~1~37*** would be a command in the namespace org.ccnx, where the operation is frobnicate, which takes 1 and 37 as arguments [2]. CCNx deals commands as extensions to content names. Operations are always performed on addressed content. SCN aims to be more flexible and allow addressing service objects (functions), not only content objects (data) in object names.

B. Web Services

Web Services provide programming interfaces that are accessed via the Hypertext Transfer Protocol and executed on a remote system hosting the requested services. Web Services are described using 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 HyperText Transfer Protocol (HTTP) for message negotiation and transmission. Representational State Transfer (REST) is a style of a 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. A REST-based service architecture could easily be implemented using CCNx by encoding the content and the methods to be performed. However, the functionality of services is rather limited then and several complex services, in particular multimedia services, are rather difficult to be implemented inside the network.

III. SERVICE-CENTRIC NETWORKING (SCN)

A. Extending Context-Centric Networking for Services Using an Object-oriented Design Approach

We propose to enhance content-centric networking to support general services. Data is not just retrieved, but can be processed before being presented to the user. We propose to extend names not only for content but also for services to be invoked. We recognize that services and content processed by services usually are two distinct entities that can reside at different locations in the network. Therefore, Service-Centric Networking (SCN) provides explicit addressing for both entities. We achieve uniform naming of services and content by using an object-oriented approach that introduces object names for both services and content. SCN leverages the concept of Interest and Data messages as defined by an underlying CCN infrastructure such as CCNx; a client sends Interest messages for services to be invoked and the results from service execution are returned in Data messages. The underlying CCN infrastructure should support name-based routing of Interest messages as well as routing of Data.

B. Advantages and Motivation

The advantages and benefits of SCN are as follows:

No service lookup and service registry: In traditional (web) service scenarios, services must be registered by the web service provider at a registry and must be looked up by the client before the service is invoked. In SCN, service registration is replaced by announcing service availability in the underlying CCN infrastructure, i.e., in the CCN routing tables. This results in a lower delay and avoids relying on an additional registry component. When invoking a service, no specific server needs to be addressed, but rather the service specified by its name. The CCN infrastructure automatically routes the service request encoded in an Interest message to the closest server supporting this service.

Caching of service data: SCN leverages the features of the underlying CCN infrastructure. Thus, routers can cache data resulting from service calls and provide these data on subsequent requests. Although the benefits of caching are reduced for personalized services (e.g., commercial transactions), caching significantly reduces network traffic and response times for popular content and services. Caching is beneficial in case of mobility. Mobile users might request data or services when being mobile. This data might be stored in the network only, e.g., in case of cloud applications, and cached at network elements close to the user. After changing network access, users might request personal data again, with a high chance of being still cached close to the user.

***Location-based services*:** Traditional location-based services work as follows: 1) the client contacts a (central) server, 2) the position of the client is determined, 3) the closest server is detected, and 4) the service request is redirected to the closest server. In SCN, location-based services can be easily built by deploying service entities at various locations and populating routing entries appropriately. Then, service requests (mapped to Interest messages) are routed to the closest (typically the local) server for processing the request independent of user locations.

***Optimized service selection**:* Service requests are sent by the client to the network using an object name that identifies the service. With the help of the underlying CCN infrastructure, the request is routed to the most appropriate location. Optimizations can consider the distance between client and server or between server and data, etc. Therefore, a new instance of the service can be started on a resource close to the user that invokes the service or close to the data. Another option is to automatically deploy services on routers that previously received many service requests.

C. Example Applications

SCN could be useful for a variety of services such as file storage and retrieval, audio/video streaming and recording, processing of stored images and video, e-Commerce applications like ticket ordering, e-banking, on-line shopping, location-based services (gas, food, travel, weather, events etc.), cloud computing, e.g., to instantiate virtual machines and data bases as well as telecommunication services, e.g., signalling functions using distributed data bases.

IV. OBJECT-ORIENTED APPROACH FOR SCN

A. Objects for Integrating Services and Data

Services typically perform some processing of data. Therefore, services need processing entities, called *servers* in the following, as well as *data stores* for storing data. Services are represented as functions to be invoked by users, e.g., Web Services. This means that a service-centric networking scheme should support both data and functions. The object-orientated programming paradigm nicely integrates / encapsulates both functions and data into objects. Methods are called among objects in order to invoke functions. We propose to use object names for both services and content requested by clients. Server functionalities and data can be handled equally; the same naming scheme should be used for both, e.g., /youtube.com/rendering (function) or /unibe.ch/braun/lecture-04052010 (data). The underlying CCN infrastructure ensures that Interest messages be routed towards those objects using object names as addresses and that Data messages find the way back to the client.

The object-oriented approach has two advantages. First, object names provide a uniform naming scheme suitable for both services and content. Second, services can be implemented as a set of cooperating objects. Cooperating objects exchange method calls by addressing the target object and the method to be called. Method parameters must be described as well.

B. Object Types

In a mixed content- and service-centric network we can envision three types of objects (Figure 1):

Pure ***content objects*** representing data (images, audio/video files, etc.) only support read methods. Read methods are the default methods to be called, if no other method is given. Those objects are equivalent to on-line accessible content such as audio/video files, articles, etc.

Pure ***service objects*** without any stored data represent service functions that are not associated with particular data and can be invoked by a client for processing its individual data. The content or location of the data must be specified in the service invocation as additional parameters. Web services are an example use-case.

Combined content and service objects integrate both services and content data. A service request is sent using the object name as address and routed towards the object storing the content data. By providing the method to be performed on the object, content data can be directly processed on the node hosting the object. This can be seen as an extension of pure content objects, by not only allowing read methods but any other supported methods to be performed on the object.

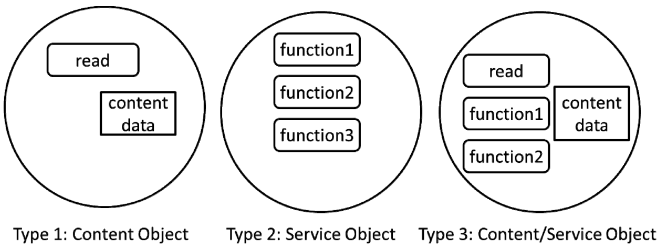

Figure 1. SCN Object Types

C. Addressing Multiple Objects

The object-oriented approach allows combining services by invoking them one after the other. We propose to make use of a routing header, cf. IPv6, in which all the objects to be visited by a service request are listed (Figure 2). The service request is first forwarded to Objectname1, subsequently to Objectname2, etc., until the service request reaches ObjectnameN. The parameters for the methods to be called are in the data part of the request, described by Parameters1-N. We propose a parameter stack, where the top elements of the stack are consumed by the addressed objects and replaced by resulting parameters. As example, Parameters1-2 can be consumed by Objectname1 and replaced by the resulting parameters as input for Objectname2.

Objects listed in the routing header might also be visited in parallel. This is possible, if corresponding functions do not depend on each other. This issue requires further investigation.

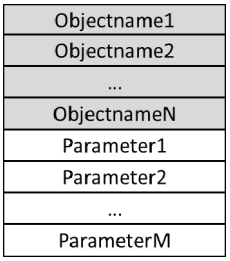

Figure 2. Combining services using routing header

D. Object Size

In the case of large data volumes such as for video files, chunking is generally used to distribute the content over several smaller pieces (e.g., in peer-to-peer and content distribution networks). If a complete video file is equal to an object, the benefits of parallel downloading of chunks cannot be exploited. However, content can usually be structured into smaller components. As an example, MPEG-4 has introduced an object-oriented approach. Therefore, we propose to divide large objects into smaller ones and use the smaller objects (elementary objects) as units for storage and processing. Elementary objects can then be used for larger composite objects, e.g., storing a complete video file.

E. Automatic Server Selection

An advantage of CCN is the possibility to support automatic server selection, e.g., the closest server for a (mobile) client. In SCN, not only the distance between client and server, but also the distance between server and required data determine the selection of the most appropriate server. Figure 3 shows a scenario, where a service request (Interest) sent from the client to the network arrives at two different possibly appropriate servers. Both servers extract the data to be processed from the service request and query the data. The response is then sent back to the client in a Data message that contains the result of service processing. Each server delays its response according to the distance between server and data resulting from querying the data objects. The first response arriving from a server at a CCN router will consume the pending Interest in the intermediate CCN router. Further Data messages from other later responding servers will be discarded.

A problem occurs when the underlying CCN infrastructure forwards the Interest message to several servers. In this case, several servers might start processing the service request in parallel. This could lead these servers to download the required data for service processing and to perform rather expensive processing in parallel. The fastest server will reply with a Data message to the request consuming the Interest, but all the processing and download of the other servers might then be useless. Mechanisms are needed to avoid excessive resource consumption. Possible approaches are described in the following:

- Use anycast to forward Interest messages, i.e., delivery of Interest message to only one server. If the nearest server according to a certain metric is selected, this requires knowledge, e.g., gained from a link-state routing protocol, at the forwarding CCN router about the distance to the appropriate servers and potential data sources.
- A server selection might precede service processing (cf. Section IV.F). The first Interest message from a client is then used for server selection. In this case, each server indicates by a Data message its willingness to process the upcoming service request. Data messages can then be intentionally delayed dependent on the costs to access the data objects by the server, e.g., by considering current load. This allows the client to select the best available server based on the incoming Data message. Subsequently, Interest and Data messages should only be exchanged

between client and selected server, preferably using a unicast session established previously between client and server during the server selection step. Identifying the best combination of servers and data replica, e.g., in terms of network delay, could be computed centrally, but might become very complex for large numbers of servers and data stores. The service-centric networking approach could be used to support optimized server and data selection in an automatic way by exchanging (and appropriately delaying) Interest and Data messages.

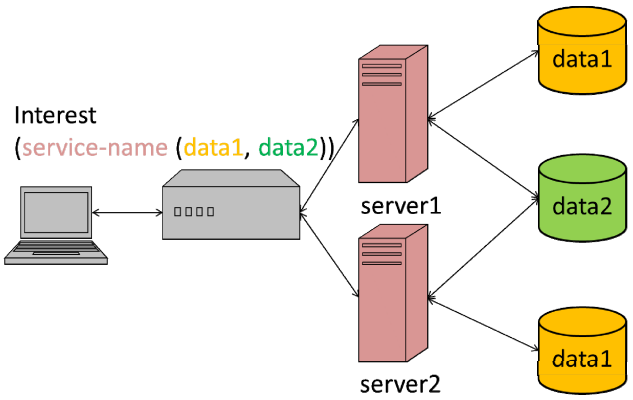

Figure 3. Automatic Server Selection

F. *Stateless and Stateful Services*

Many services require both client and server to share some client-specific context. These are called *stateful services*. Examples are shopping or e-banking applications as well as personalized services requiring to access personalized user profiles. A common requirement of stateful services is that clients need to talk to the same server, also due to security reasons. In this case, we propose to establish sessions between clients and servers to ensure that the client always talks to the same server.

A session_establishment method can be sent in an Interest message to a service object prior to exchange of service requests and results. The Data message on the reverse (or optimized) path should then establish a session within the CCN router. All service requests from the client to the server and the responses from the server to the client should be forwarded along the session.

If multiple clients connect to the same service, a session tree could be established. This could be useful in scenarios where a single server serves several clients sharing the same service requirements, e.g., for real-time video streaming.

The session concept requires modification of the underlying CCN infrastructure. In case of CCNx, an additional data structure (in addition to the forwarding information base, the content store, and the pending interest table) called *session table* in the following needs to be introduced. The session table should contain entries for all Interest and Data messages to be exchanged between a client and server. Each session table entry should at least include object name, session identifier, and a timeout when the session expires. The entries can be implemented as soft states and are somehow similar as the gradients in Directed Diffusion, a routing protocol based in Interest and Data messages for wireless sensor networks. Sessions should support both unicast and multicast distribution of data. As an option, sessions could be supported using OpenFlow [8] to bypass a CCN router for packets of an established session.

V. Service Implementations using SCN

This Section describes how two example services can be implemented based on the object-oriented SCN approach. The first example is a simple service processing possibly distributed data; the second example presents the implementation of service consisting of a chain of subservices.

A. *Video Rendering*

As a first example we consider the case of rendering one or more video files. There are two options for implementing such a service. In the first option, the video files and the rendering functions are separated objects (Figure 4). We further assume that the video file objects are located on an end user device, while the rendering function object is located on a server hosted by a video rendering service provider. The rendering service can be implemented by sending a method call from the end user device hosting the video file objects to the server hosting the rendering object (1). As parameters we deliver the video file object identifiers. We use an Interest message for transmitting the method call. This allows the underlying CCN infrastructure to localize the closest desired rendering function automatically. The rendering server can send a read method call (again using an Interest message) to the clients addressing the video file objects to be rendered and download the video files for further processing using Data messages (2, 3). Finally, the server creates and writes the resulting rendered video into a new object (4) and returns the result of the rendering, e.g., the location of the final rendered video, to the client (5).

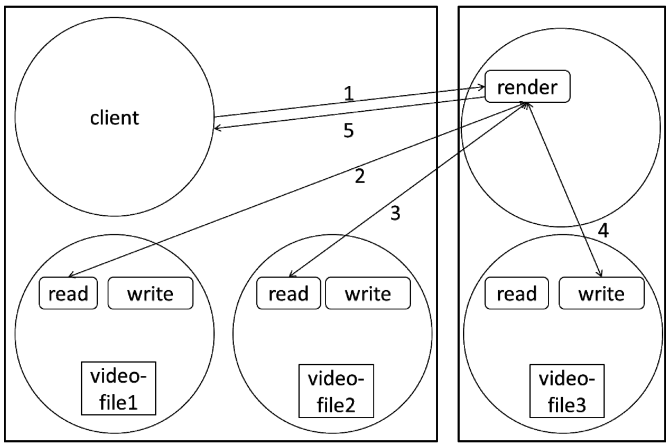

Figure 4. Video Rendering with Separate Objects for Data and Rendering (Option 1)

Another option is to have objects including both the rendering functionality and the video data (Figure 5). The client sends the service request (Interest) to one of the video file objects indicating that the addressed file and a second file need to be rendered together (1). The object receiving the request, reads the second file (2), performs rendering, creates a new object

and stores the result therein (3) and informs the client about the newly created object (4).

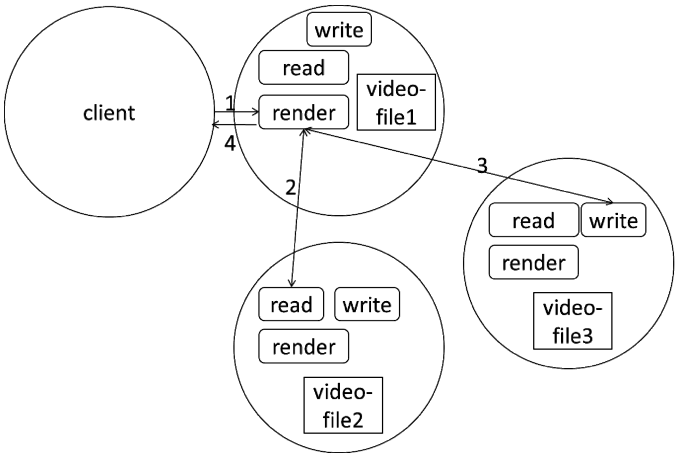

Figure 5. Video Rendering with Integrated Objects for Data and Rendering (Option 2)

B. Real-Time Audio Conferencing

In this example, we illustrate how a real-time audio conferencing service could be implemented using several functional service elements available inside the network or in a cloud computing environment. Please note that these services could be implemented in alternative ways, e.g., by putting more functionality into the end systems. Moreover, the service can be implemented using CCN principles, i.e., addressing only content and invoking operations on that. This example aims to show how functions/services available within the service-centric network can be addressed in order to provide services by network operators.

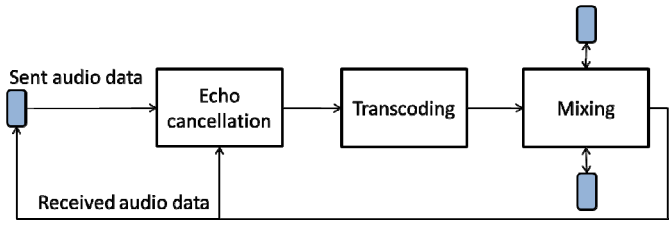

Figure 6. Real-time Audio Conferencing Scenario

In our service example, a number of audio conferencing participants can communicate with each other. A mixer receives the different audio streams, mixes them to a single output stream, and sends them back (individually or by multicast) to the individual participants. We assume here that the mixer produces a single output stream to be sent back to all participants. Alternatively, individual selections of participants to be mixed could be specified by each participant. We further assume that each participant is able to process the mixer's output. Otherwise, transcoders are required for the reverse direction, too. In addition to mixing, the service shall include an echo canceller and a transcoder for the stream from client to mixer in order adapt the individual audio formats to the format desired by the mixer. The real-time audio conferencing application scenario is depicted in Figure 6.

First, the client audio application transmits its generated audio stream to the echo canceller, which removes the audio of the reverse direction. This audio stream is either available from the mixer or from the audio client, which should have received it for play-out. Here, we assume that the echo canceller receives the mixed audio from the mixer. The output of the echo canceller is forwarded via the transcoder to the mixer. Other clients are served by the mixer as well.

Figure 7 shows the resulting packet flow to implement the real-time audio conferencing service in a service-centric network. First, an audio stream has to be generated by the user's audio tool running at the client (1). The audio data can be generated periodically and span over a certain time period in the range of a few ten milliseconds. This allows rather low algorithmic delays as well as low packet overhead.

The user's audio tool (e.g., /john/audiotool) sends an Interest message to the /echocancel object and describes the three service steps (echo cancellation, transcoding, and mixing) to be performed on the audio data by inserting the three names of the service objects in the routing header of the Interest message (2). The audio data parameters serving as input are described in the data part of the Interest message. The Interest message can be described by the notation (/echocancel, /transcode, /mix; /john/audiostream_original, /mix/todaysconf/audiostream).

The /echocancel object reads both the audio data generated by the client (e.g., /john/audiostream) and from the mixer (e.g., /mix/todaysconf/audiostream) (3) in order to remove echo signals. For retrieving the audio data Interest messages are sent to the content names, i.e., /john/audiostream_orignal and /mix/todaysconf/audiostream. As a result, the /echocancel object stores the echo cancelled audio by writing the results into the newly generated object /john/audiostream_ec (4). Next, the echo canceller forwards the Interest message from the user to the /transcode object (5). A pointer in the routing header depicts the location of the currently addressed service object. The parameters in the data section can be replaced for the next data object. As an example, the Interest message contains the name (/john/audiostream_ec) of the object to be used for transcoding.

Next, the /transcode object receives the Interest message from the /echocancel object, reads the audio data /john/audiostream_ec (6) and writes the result into the newly generated /john/audiostream_trans object (7). Procedures are similar as in the echo cancelling step.

The /mix/todaysconf service object has been created for a particular conference and performs mixing of several audio streams. When receiving the Interest message from our client via /echocancel and /transcode (8, with /john/audiostream_trans as parameter to indicate the object t be used for mixing), it reads the /john/audiostream_trans object (9) and updates the mixed audio data stream /mix/todaysconf/audiostream (10).

Interest message processing has been finished at this point of time. The Data message has to be returned along the return path from /mix/todaysconf via /transcode and /echocancel to the client /john/audiotool (11-13). The Data message indicates the result of Interest message processing. In particular, it indi-

cates the final result, i.e., /mix/todaysconf/audiostream. This object can then be used by the client for playout and by the echo canceller for mixing. In case of a failure during Interest message processing at one step of the chain, a Data message is returned earlier, indicating partial Interest message processing. The originator can then take appropriate actions, e.g., searching for alternative service elements.

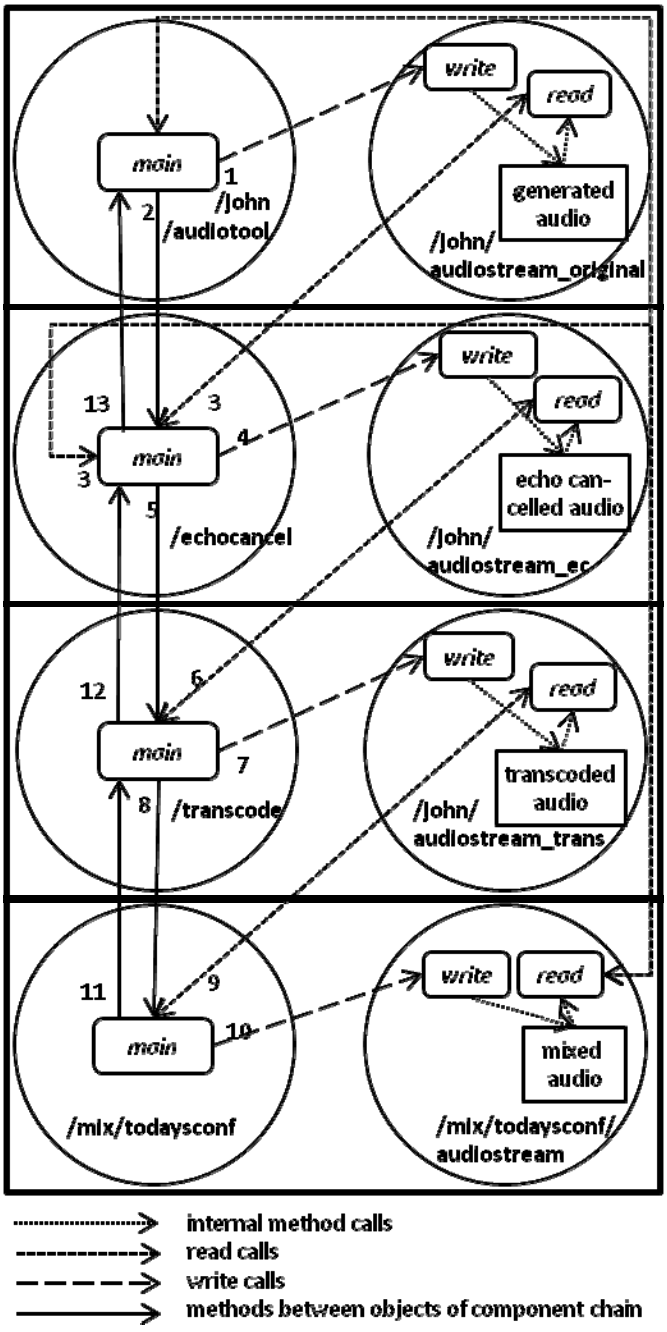

Figure 7. Implementation architecture of a distributed implementation of an real-time audio conferencing service.

The concept described has several advantages to be exploited. Data objects can be replicated, e.g., the client can create a copy of the /mix/todaysconf/audiostream object. This object can then be used by the echo canceller for its own purposes. SCN routing of the request to read the object /mix/todaysconf/audiostream is automatically forwarded to the closest source, either at the client or the mixer.

VI. Conclusions and Open Issues

This paper has motivated the need for service-centric networking (SCN) as an extension of content-centric networking. We propose to consider content and services as objects and use object names as addresses in a Future Internet routing infrastructure. We showed how composed services can be built and benefit from SCN. Besides implementing SCN, there are several open issues to be solved. First, a naming scheme for services must be investigated. One potential solution is to structure and classify services hierarchically, as proposed in [10]. This hierarchy could be adopted as the naming scheme for services. Second, our approach currently does not prevent services from being routed to a server that does not support the specified number or types of parameters specified in the body of the service request (Interest) message. Searching for services matching the needs of a client in terms of supported parameter types might be too complex to be solved on the routing level. Another issue is to define a routing function on two names for services and content, respectively. This could be done based on preferences, weights, closest first, etc.

References

[1] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. 2009. Networking named content. In *Proceedings of the 5th international Conference on Emerging Networking Experiments and Technologies* (Rome, Italy, December 01 - 04, 2009). CoNEXT '09. ACM, New York, NY, 1-12. DOI= http://doi.acm.org/10.1145/1658939.1658941

[2] http://www.ccnx.org/releases/latest/doc/technical/NameConventions.html, visited September 1, 2010

[3] http://www.ccnx.org/, visited September 1, 2010

[4] Scaffold, http://sns.cs.princeton.edu/projects/scaffold, visited September 1, 2010

[5] Friedmann, M.: Building a Service-Centric Network with SCAFFOLD, http://netseminar.stanford.edu/seminars/freedman-scaffold.pdf

[6] Koponen, T., Chawla, M., Chun, B., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. 2007. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications* (Kyoto, Japan, August 27 - 31, 2007). SIGCOMM '07. ACM, New York, NY, 181-192. DOI= http://doi.acm.org/10.1145/1282380.1282402

[7] Gritter, M. and Cheriton, D. R. 2001. An architecture for content routing support in the internet. In *Proceedings of the 3rd Conference on USENIX Symposium on internet Technologies and Systems - Volume 3* (San Francisco, California, March 26 - 28, 2001). USENIX Association, Berkeley, CA, 4-4

[8] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J.. OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, 38(2): 69–74, April 2008.

[9] Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F. 2003. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.* 11, 1 (Feb. 2003), 2-16. DOI= http://dx.doi.org/10.1109/TNET.2002.808417

[10] Arabshian, K., Schulzrinne, H: An Ontology-based Hierarchical Peer-to-Peer Global Service Discovery System, Journal of Ubiquitous Computing and Intelligence Volume 1, Number 2, pp 133-144, December 2007