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 that treats content as a first class citizen. 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, and discuss opportunities as well as open research issues.**

I. Introduction

Content-centric networking (CCN) is a networking paradigm for the Future Internet, in which routing and forwarding is 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 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. In our design we consider both content and services as key elements so that 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.

II. Related Work on 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 packet 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 for the implementation of NNC and uses a 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 [5] and DONA [4] 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 "~". A 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).

III. Service-Centric Networking

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, SCN provides explicit addressing for both entities. We achieve uniform naming of services and content by using an object-oriented approach, and introduce object names for both services and content. SCN uses the concept of Interest and Data messages as introduced 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. 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 table. 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. 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. Service requests (mapped to Interest messages) are routed to the local server for processing independent of user locations.

***Optimized service selection**:* Service requests are sent by the client to the network using an object name representing the service as an address. With the help of the underlying CCN infrastructure the request is routed to the most appropriate location. Optimization 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.

IV. Object-oriented Approach for Service-Centric Networking

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. While content-centric networking focuses mainly on data, we aim to support services as well. Services are represented by 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 functionality and data can be handled equally; for both the same naming concept should be used, e.g., /youtube.com/rendering or /unibe.ch/braun/lecture-04052010. 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. In a mixed content-centric and service-centric network we envision three types of objects:

Pure ***content objects*** representing data (images, audio/video files, etc.) only support read methods. 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.

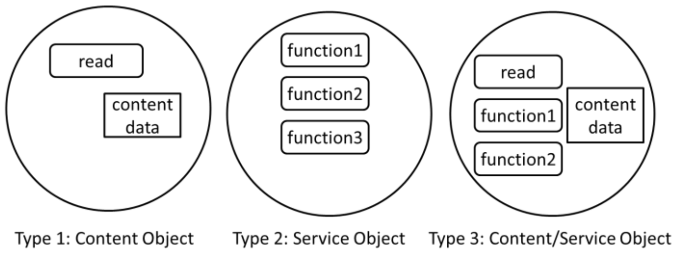

Figure 1. SCN Object Types

V. Open Issues

Besides implementing SCN, there are several open issues to be solved. First, a naming scheme for services must be investigated. [3] proposes a peer-to-peer based hierarchical service discovery. Services are structured and classified hierarchically. This hierarchy could be adopted as the naming scheme for services. Second, our approach currently does not avoid services to be 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 routing level. Another issue is to define a routing function for two (or more) names: service and content. 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.. Networking named content. 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT), pp. 1-12, Rome, Italy, December 1-4, 2009.

[2] www.ccnx.org/releases/latest/doc/technical/NameConventions.html

[3] Arabshian, K. and Schulzrinne, H. An Ontology-based Hierarchical Peer-to-Peer Global Service Discovery System, Journal of Ubiquitous Computing and Intelligence, Vol. 1, No. 2, pp. 133-144, December 2007

[4] Koponen, T., Chawla, M., Chun, B., Ermolinskiy, A., Kim, K. H., Shenker, S., and Stoica, I. A data-oriented (and beyond) network architecture. ACM SIGCOMM, Kyoto, Japan, August 27-31, 2007, pp. 181-192

[5] Gritter, M. and Cheriton, D. R.. An architecture for content routing support in the Internet. 3rd Conference on USENIX, Symposium on Internet Technologies and Systems, Vol. 3, San Francisco, March 26 - 28, 2001, p. 4