Dynamic Service Scalability in Information-Centric Networks

Suman Srinivasan[1], Dhruva Batni[1], Volker Hilt[2] and Henning Schulzrinne[1]
[1] *Columbia University,* [2] *Alcatel-Lucent*
sumans@cs.columbia.edu, dlb2155@columbia.edu, hgs@cs.columbia.edu, volker.hilt@alcatel-lucent.com

Abstract

Information- or content-centric networks have gotten a lot of interest recently, particularly due to the promise to address problems inherent in today's host-based networking architecture. But while information-centric networking aims to address the concerns of networking which is evolving in the direction of serving content, it does not inherently address the issue of services, particularly service scalability, which is also an important part of networking. We present our work on dynamic service scalability in information-centric networking, particularly our implementation on top of the CCNx framework.

1. Introduction

Information-centric networking aims to solve a trend in computer networking: today, content is one of the key players in networking. Information-centric networking, such as the CCNx project [1], aim to solve this problem by specifically addressing content, and rewriting the entire networking stack by focusing on content requests.

However, current information-centric networking implementations fail to address the issue of services, which are at least as important as raw content. The popularity of services such as Facebook and Google allude to the fact that users are also interested in services. Hence we believe that a complete networking paradigm will focus on not only information-centric networking, but will involve services as well.

In this poster, we present our architecture and current implementation of dynamic services and service scalability on top of an existing information-centric networking framework, CCNx. We present how we treat service modules as content, thus leveraging the existing information-centric features of CCNx, while naturally allowing for service scalability and mobility.

2. Scaling Services Dynamically in CCNx

There has been some recent work to address the concept of services in content networks. In this vein, "service-centric networking" [2] is one of the most relevant papers. SCN aims to address service invocation and service scalability and mobility in the network through building a "superset" of CCNx.

In our work, we attempt to implement service centric networking on top of CCNx, instead of building a new networking paradigm. Our implementation allows for a service name to be invoked alongside the content name. Thus, when a content name is seen by a content router, the content router is able to parse the name into the content name and the corresponding service that is to be invoked on it. If it does not have that service, it fetches the corresponding module via CCNx and invokes the service on the content, and puts the processed content back into the CCNx namespace so that future requests for the content fetch this processed content.

3. Architecture

In our current implementation, a "client" or a node makes a request for a content name and a service it wishes to invoke on the content (ccnx://video.mp4+service). This request is converted to a CCNx Interest packet and forward. The request is intercepted by any of the nodes that operate as a Content Router, and if the content does not exist, the Content Router looks for a service corresponding to the name. If it does not find a service, it fetches it by issuing a CCNx Interest packet for that service module. Once the service module is located and downloaded on the Content Router, it invokes the service module on the content, thus producing a processed version of the content.

5. Future Work

We are currently in the process of integrating our work more fully into the NetServ [3][4] service virtualization framework to get the best of two worlds – information-centric networking and in-network service virtualization.

6. Conclusion

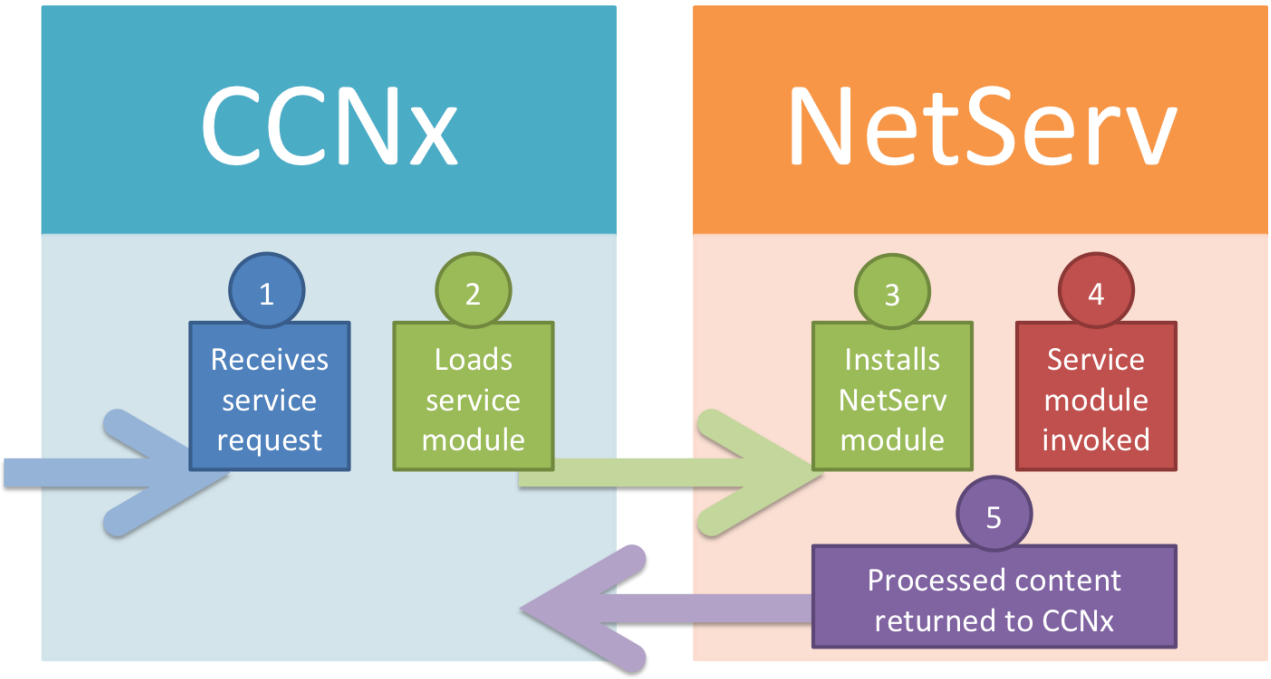
We believe that services are central to network operations. In our work, we aimed to show that it is possible to integrate service functionality – including dynamic service invocation, scalability and mobility – into the core of information-centric networking, thus extending its features. To this end, we have a working implementation of a services scaling architecture implemented on top of the CCNx protocol stack.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard (PARC) Networking Named Content, CoNEXT 2009, December 2009.
[2] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, M. Varvello: Service Centric Networking, FutureNet IV workshop, Kyoto, Japan, June 2011.
[3] Suman Srinivasan, Jae Woo Lee, Eric Liu, Mike Kester, Henning Schulzrinne, Volker Hilt, Srini Seetharaman, Ashiq Khan, "NetServ: Dynamically Deploying In-network Services", ACM ReArch '09 (CoNEXT workshop), December 2009.
[4] Jae Woo Lee, Roberto Francescangeli, Jan Janak, Suman Srinivasan, Salman Abdul Baset, Henning Schulzrinne, Zoran Despotovic, Wolfgang Kellerer, "NetServ: Active Networking 2.0", IEEE ICC 2011, Kyoto, Japan, June 2011.

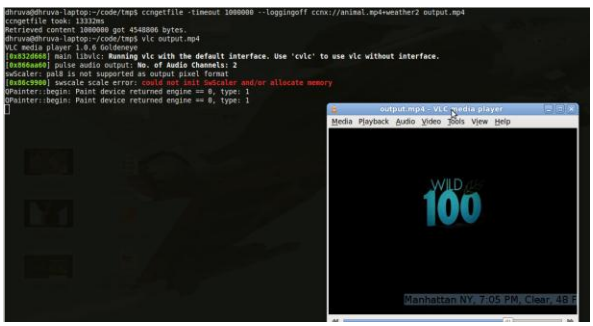
Dynamic Service Scalability in Information-Centric Networks

Suman Srinivasan, Dhruva Batni, Volker Hilt and Henning Schulzrinne
Columbia University, Alcatel-Lucent

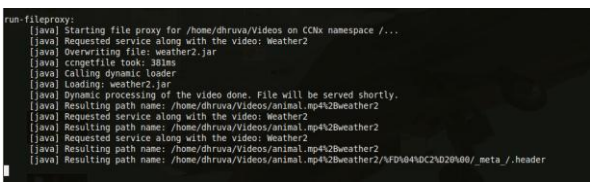


Details: How dynamic services in CCNx would integrate with service invocation in NetServ, a virtual services container.

Screenshots



A processed content is obtained through CCNx and played in VLC media player. There is a small watermark with weather information at the bottom right of the video, showing that this is processed video obtained from CCNx.



When a service request is made, the service module (in addition to the content) is downloaded dynamically from CCNx, and invoked on the content.

Pseudo-Code for Implementation

```
ccnName = "ccnx://test.mpg+weather";
list(service, file) = parse (ccnName);
download ("ccnx://netserv/" +
                    service + ".jar");
download ("ccnx://content/" + file);
loadLocalJAR (service + ".jar");
processedFile = loadClass("Process").
          getMethod ("run").invoke(file);
putFileIntoCCNx (processedFile);
```

How It Works

- Explanation

Future Work

- Integration with NetServ

Conclusion