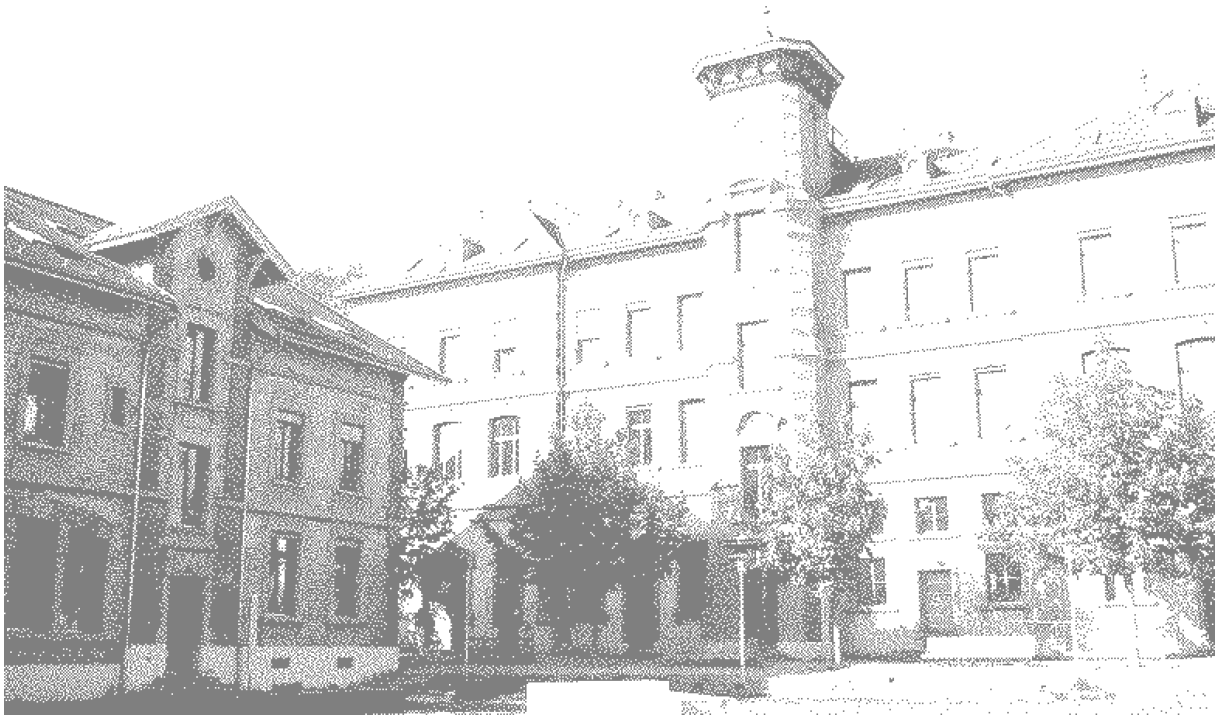
UNIVERSITÄT
BERN

Performance Evaluation of Service Support in Content-Centric Networking

D. Mansour, T. Braun

Technical Report IAM-14-003 , 24. November 2014

Institut für Informatik und angewandte Mathematik, www.iam.unibe.ch



Performance Evaluation of Service Support in Content-Centric Networking

Dima Mansour, Torsten Braun

Technical Report IAM-14-003 , 24. November 2014

CR Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design;
C.2.2 [Computer-Communication Networks]: Network Protocols;
C.2.3 [Computer-Communication Networks]: Network Operations;
C.2.4 [Computer-Communication Networks]: Distributed Systems;

General Terms:
Design, Management, Measurement, Performance

Additional Key Words:
Service-Centric Networking, Content-Centric Networking, Service Orchestration, Layered Architecture.

Institut für Informatik und angewandte Mathematik, Universität Bern

Abstract

NextServe is a Service-Centric Networking approach built on top of Content-Centric Networking (CCN). NextServe allows for the publishing and invocation of remote services on top of the name-based CCN routing. NextServe has a human-readable naming scheme that supports caching, parameter passing, and service aggregation.
In this report, we enhance the architecture and the naming scheme of NextServe to support heterogeneous applications, services, and protocols. We evaluate the performance of NextServe regarding service response time and caching. We show that NextServe has an insignificant overhead and supports CCN caching efficiently.

Contents

1 Introduction

Current Internet protocols use IP addresses for routing messages. Clients have to connect to hosts identified by host names and IP addresses to retrieve data. Information-Centric Networking (ICN), on the other hand, uses content names for addressing, *i.e.*, no location based identifiers. This approach is manifested in many research projects like Content-Centric Networking (CCN) [1], Data-Oriented Network Architecture
(DONA) [2], and Publish-Subscribe Internet Routing Paradigm (PSIRP) [3]. The main motivation behind these projects is that named content is a better abstraction for today's communication problems, like security and mobility issues, than named hosts.
Other projects like Service-Centric Networking (SCN) [4][5], Serval [6], CCNxServ [7], and Named Function Networking (NFN) [8] try to extend the idea of ICN towards providing in-network services making services at the core of the communication model. Each project of those has its own protocol, naming scheme, and architecture and cannot interact with the conventional architectures or protocols making it very hard to combine the current Internet architecture with future Internet architectures.
We believe that in the first phase of deploying any future Internet architecture there should be room for coexistence and cooperation between legacy and future Internet architectures. Otherwise, costs will make it unrealistic to deploy a new communication model.
In our previous research [9], we developed a framework called NextServe to support services over CCN. Parameters of published services were either local parameters sent directly from the client, published content, or other atomic services. In this paper, we enhance the naming scheme and extend the architecture of NextServe [9] to support services over CCN with minimal overhead, as well as to support other conventional communication protocols like FTP and HTTP for parameter retrieval. We also conduct experiments to measure the NextServe overhead and the service response time under various request rates and cache conditions.
The rest of the paper is organized as follows: We give a short technical background on CCN in Section 2. Then, we present the NextServe framework, its naming scheme, and its architecture in Section 3. In Section 4, we evaluate the performance and the cache usage in NextServe on a local testbed. Then, we discuss the related work in the field of Service-Centric Networking in Section 5. Finally, we conclude this paper in Section 6.

2 Content-Centric Networking

The Content-Centric Networking architecture [1] retrieves content objects by name and not by location. In CCN, the consumer sends an Interest message with the name of the requested content. The CCN router forwards the Interest over the network using name-based routing. When the content publisher receives the Interest, it sends back the corresponding content as a Data packet, which follows the bread crumb of the Interest packet.
When a CCN router receives an Interest packet, it follows the steps in Figure 1. It checks whether the requested content object is already processed and stored in the Content Store (CS). If it is, it sends it back to the consumer without forwarding the Interest to the content publisher. This is the caching mechanism in CCN. If there is no cached data in the CS for the requested named content, the router checks the Pending Interest Table (PIT) to determine whether another Interest for the same named content already exists. If there is a corresponding entry in the PIT the router just updates the PIT entry by adding the incoming face of the Interest. If there is no similar entry with the same requested named content in the PIT, the router registers the name in the PIT and forwards the Interest via the faces specified in the Forwarding Information Base (FIB).

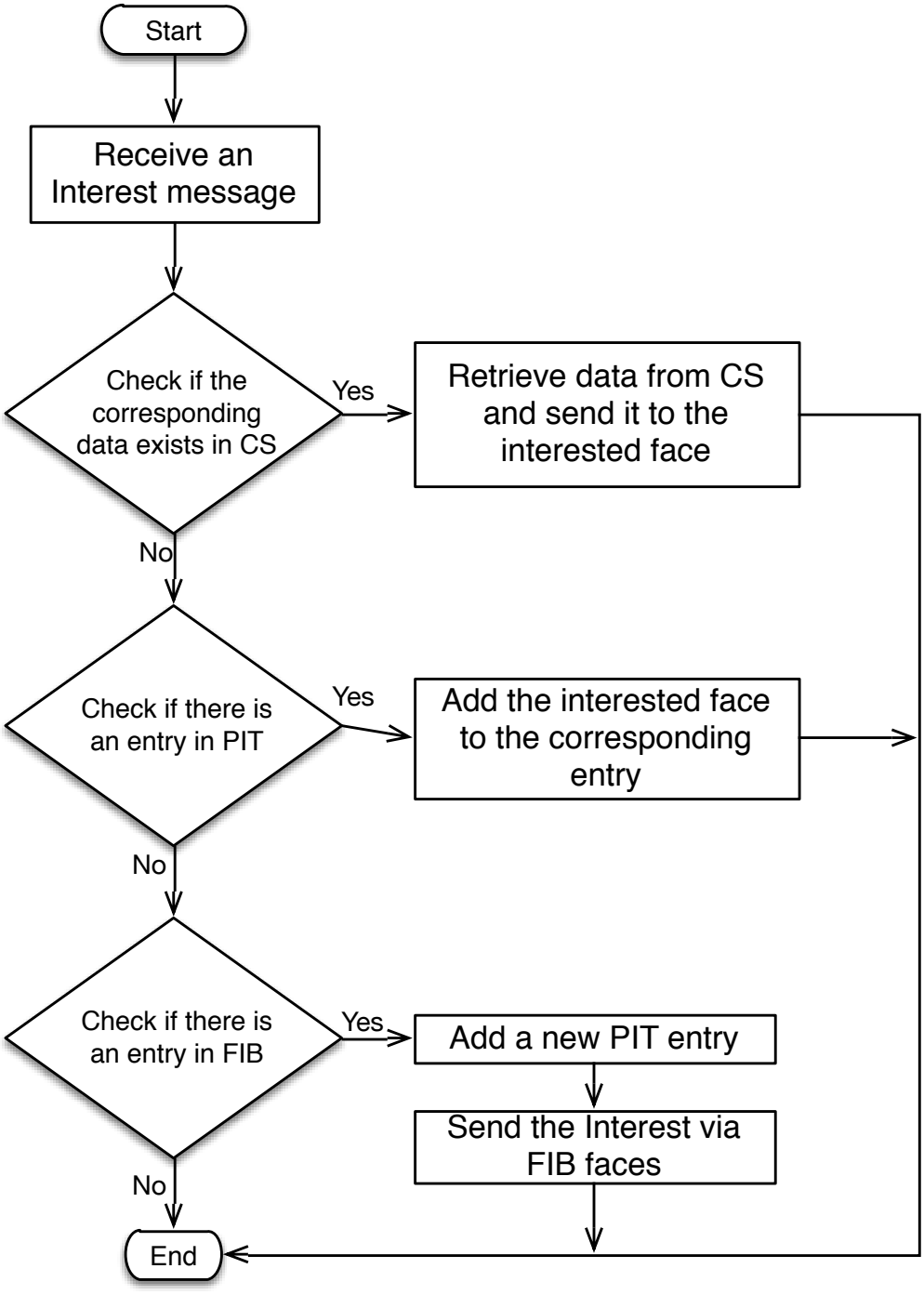


Figure 1: Processing an incoming Interest in the CCN Router

3 NextServe Framework

Previously [9], we introduced NextServe as a Service-Centric Networking approach built on top of CCN. It facilitates the request of services by name. We extend the architecture of NextServe and modified the naming scheme for services to support multiple protocols besides CCN.

3.1 NextServe Naming Scheme

```
/prefix1/prefix2/.../prefixN/!{“Param1” : “Value”,
                               “Param2” : “Value”,
                                        .
                                        .          }
```

Figure 2: The general structure of a service name in NextServe

The new naming scheme for services in NextServe, which is presented in this paper, is human-friendlier and more expressive. It consists of the service name, which is a hierarchical CCN name, followed by “!”, and followed by the service parameters in JSON [10] format (JavaScript Object Notation) as described in Figure 2.
For example, suppose we have a service that takes three parameters; the first parameter is an image file, the second parameter is an integer representing the width, and the third parameter is an integer representing the height. This service scales the specified image file to the specified dimensions and returns the scaled image file. The Interest name of this service can be as described in Figure 3.

```
/unibe/cds/scaleImage/!{ “file” : “ccnx:/dir/imgs/profile.bmp”,
                         “width”  : “300”,
                         “height” : “200” }
```

Figure 3: An example of a service with local parameters and published content parameters.

Also suppose that we have a service that takes an image file of any format and transforms it into JPEG. In NextServe, we can pass the result of the "scaling" service as a parameter to the JPEG conversion service. The Interest name of such request is as in Figure 4. Note that the name prefixes of the two services can be different without restricting the ability to combine them.

```
/scn/toJPEG/!{ "image" : "/unibe/cds/scaleImage/!{ "file" : "ccnx:/dir/imgs/profile.bmp",
                                                   "width"  : "300",
                                                   "height" : "200" }" }
```

Figure 4: An example of a service that takes the result of another service as a parameter.

This naming scheme is a combination of the hierarchical structure of CCN names [1] and the attribute-value naming scheme in the Combined Broadcast and Content-Based routing (CBCB) [11]. It also has some similarities with the current HTTP query format and RESTful web APIs. In that sense, NextServe is not a novel idea itself but rather a novel application of a novel combination of known techniques in a novel context.
This naming scheme has the following features:

- The service publisher publishes the service under the service name only without the parameter part. When a CCN router receives a service Interest, it takes only the service name into account without the service parameters for routing the Interest. This is necessary because the parameter values change from one Interest to another and they should not affect service routing. But still, the full name is taken into account when the CCN router checks the Content Store (CS) or the Pending Interest Table (PIT). In other words, routing is based only on the service name and caching is based on the service name with parameters.

- The service Interest name is user-friendly and expressive.

- NextServe allows for service overloading. Two services might have the same name but with different parameter names or different numbers of parameters as in method overloading in object-oriented programming languages.

- As explained in the example in Figure 4, service parameter values can be simple values, content names, JSON objects, or even service names. Also parameter values can be HTTP URIs, FTP URIs, or any other URI-based content names.

3.2 NextServe Architecture

The NextServe framework follows the layered architecture as shown in Figure 5. It consists of three layers: The Communication Protocols layer, the Service Publishing layer, and the Services layer.

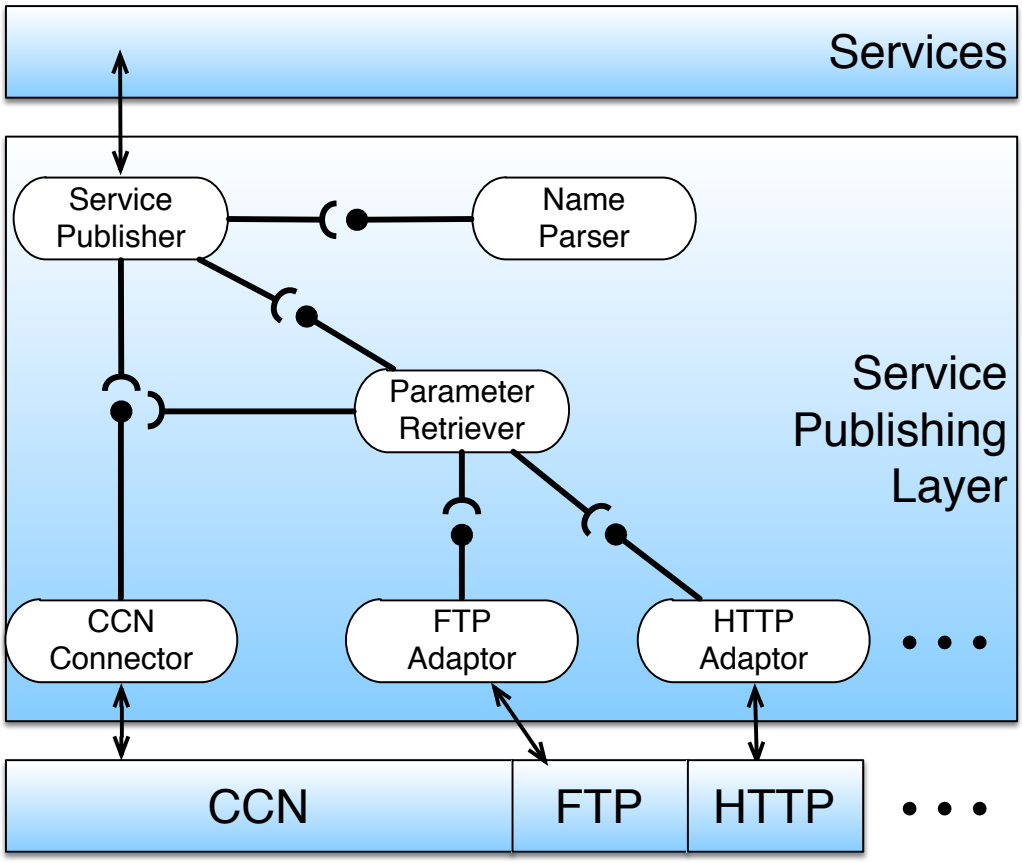


Figure 5: NextServe architecture

The Services layer (at the top) contains the concrete implementations of the published services. The Service Publishing Layer (in the middle) contains all the necessary components responsible for publishing services,

handling Interests and service parameters, in addition to invoking service implementations. The lowest layer contains the CCN core and other supported communication protocols.
The service publisher component is the only component that interacts with the service implementation. When an Interest is received by the CCN connector, it forwards it to the service publisher. The service publisher uses the name parser to extract information about the service implementation to invoke, as well as the values of the service parameters. If there is a parameter that needs to be fetched from remote nodes (content or service), the parameter retriever issues requests or Interests through the appropriate adapter and passes the fetched values to the service publisher to invoke the service implementation.
In CCN, each Data packet corresponds to one Interest message only. Content objects can be composed of multiple packets (chunks). In that case, the client sends an Interest for each chunk. In NextServe, the client sends only one Interest to request a service and the Service Publishing Layer establishes a stream over CCN to get all the chunks corresponding to the service reply transparently from the client to keep the packet segmentation implicit.
It is worth noting that services do not always produce the same results for the same parameter values. Some services depend on external factors outside the control of the client (time, database, *etc.*). For that reason, the client can explicitly direct the service request not to be retrieved from the cache but from the service provider itself. This is done by setting the ”AnswerOriginKind” flag in the Interest to 0 (do not use the cache) or to 1 (use the cache when possible).

4 Evaluation

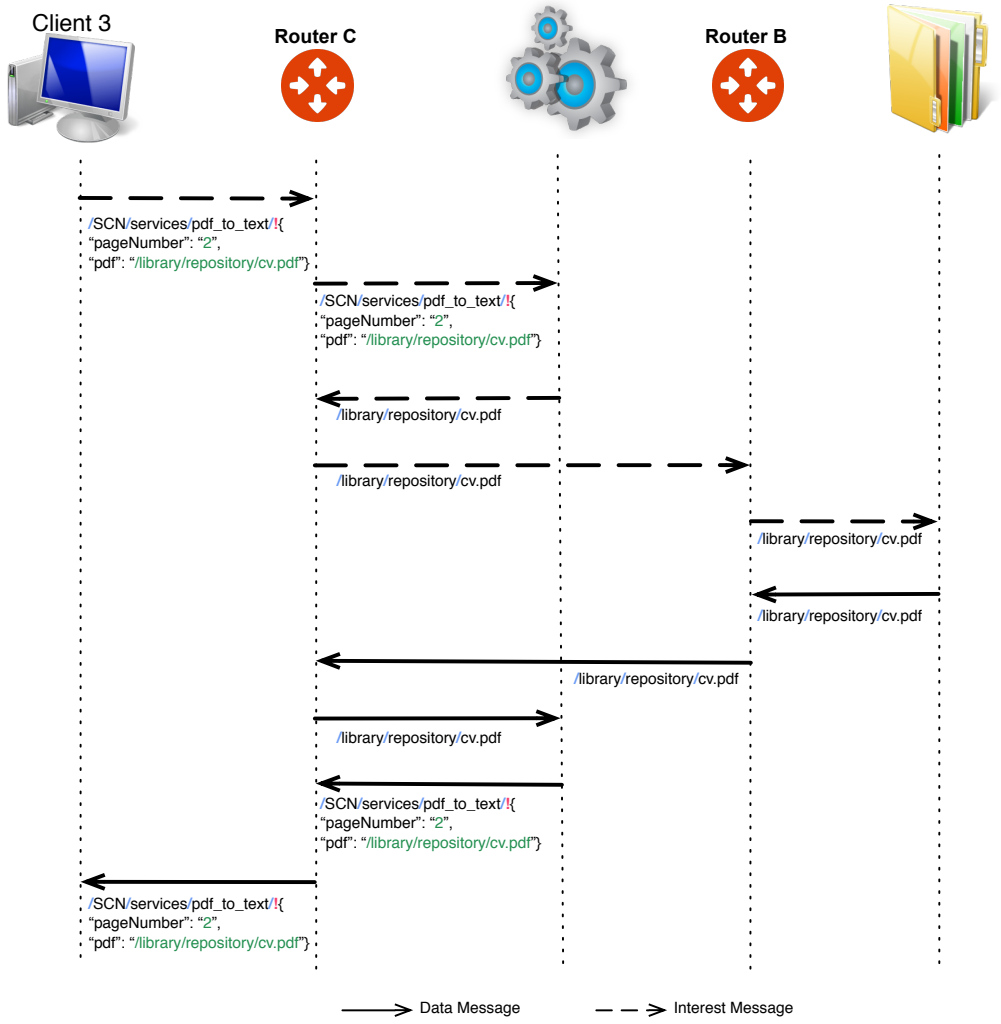


Figure 6: Processing of an example service request (Interest) in our testbed.

We conducted experiments using the NextServe framework on top of CCNx 0.8.0. Everything was deployed on a local testbed to evaluate the overall performance of SCN over CCN in terms of the following aspects:

- What is the overhead of NextServe regarding response time?
- How is the response time distributed among CCN, NextServe, and service functionality?

- How beneficial is including the parameters in the Interest of a service request considering cache utilization?

4.1 Testbed Specifications

We ran our evaluation experiment on three physical machines as shown in Figure 7. Each machine has an AMD Opteron 252 CPU (1-core, 2600MHz) and 8GB of RAM. Each machine runs Linux Debian 7.0 "*wheezy*" and Debian Xen Hypervisor for kernel-based virtualization.

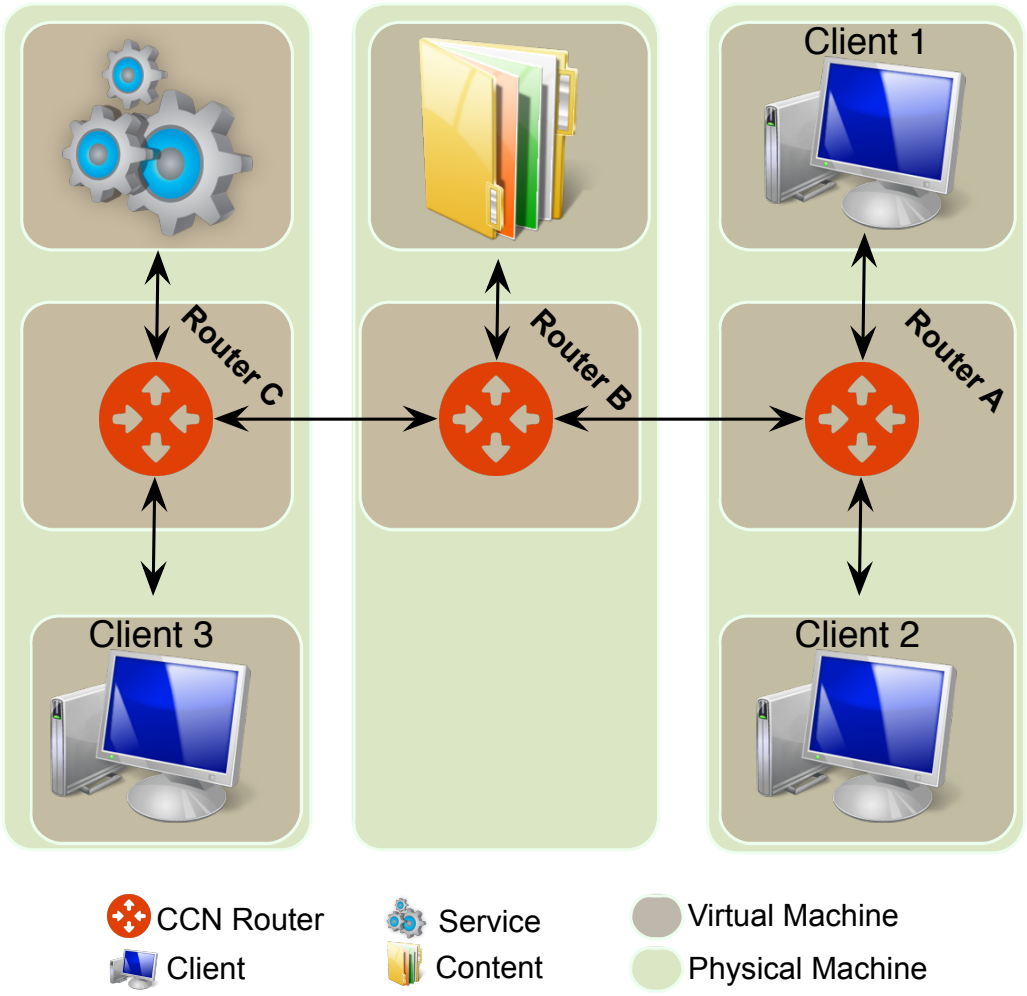


Figure 7: Overview of the testbed infrastructure.

As shown in Figure 7, each node (client, service provider, content provider,

and router) runs on a kernel-based virtual machine with 2GB of RAM. Each node also runs "ccnd", which is the routing daemon for CCNx. All network connections are Ethernet 1Gbps.

4.2 Experiment Specifications

The experiment is composed of a service provider, a content provider, and three clients. The service name is "*/SCN/services/pdf_to_text*". It takes two parameters; the PDF file content name and the page number. Then the service returns, as a string, the text of the specified page in the specified PDF file.
The content provider publishes many PDF files to the CCN network with the name prefix "*/library/repository/*" followed by the file name. Each of the published files is 200KB of size to avoid affecting the evaluation in an unexpected way. Files with different sizes takes different times for processing and communicating over the network.
Figure 6 explains how a client issues an Interest message for the pdf_to_text service with a content name as a parameter. After receiving the Interest, the service provider issues an Interest for the specified content, fetches the corresponding Data message, executes the service functionality, and sends back the result as a Data message.

4.3 Results

4.3.1 Throughput Evaluation (No Cache)

In this experiment, we disabled CCN caching and made the clients request the service at different rates, scaling from one request per second to 25 requests per second. In other words, the service provider first receives 1 request per second for five seconds. Then it receives 2 requests per second for five seconds, then 3 requests per second for five seconds, and so on up to 25 requests per second. We made sure that all sent Interests reach the service provider by changing the service parameters so that no Interest is duplicated in the PIT of any CCN router. After that we take the average of measures of response times, service execution times, and NextServe overheads at each request rate. Note that for each request: Response Time = Service Execution Time + NextServe Overhead + CCN Overhead. Response time is the end-to-end time from the moment a client issues a service Interest until it receives the reply. Service execution time, NextServe overhead, and CCN overhead are the times spent in

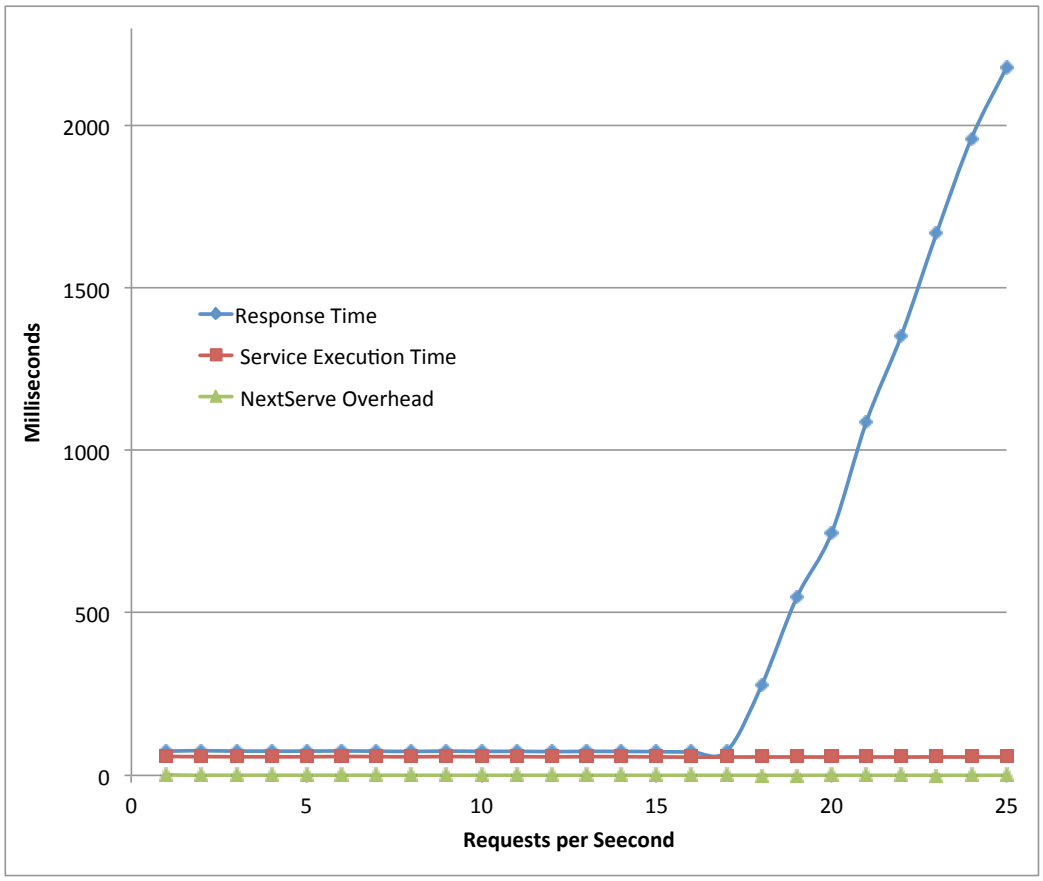


Figure 8: The measures of response time, service time, and NextServe overhead at different request rates.

the services layer, the service publishing layer, and the CCN layer correspondingly as in Figure 5.

Figure 8 shows the measurements of the various types of overhead in the experiment. There are many observations regarding this experiment. First, as expected, the per-Interest overhead of NextServe at the service provider node is constant (≈0.5 ms) and statistically insignificant. Second, the service execution time is also constant (≈57 ms). Third, and most importantly, the response time is constant (≈75 ms) for up to 17 requests per second. After that, the response time starts to increase linearly. We found out that the main reason for this result is that CCNx is not multi-threaded. This is a big problem in the current CCNx implementation because the CPU cores are not well-utilized. So each incoming request must wait until all previous requests have been handled. This explains the linear increase

in response time after 17 requests per second, which is in our scenario the maximum service throughput of the service (the number of requests processed per second). In the current implementation of CCNx, the service throughput can be computed as follows:

$$ServiceThroughput \approx \frac{1000}{ServiceTime(millisecond)}.$$

It is also worth noting that the CCN overhead is constant as long as the request rate is below the service throughput.

4.3.2 Response Time Evaluation (with Cache)

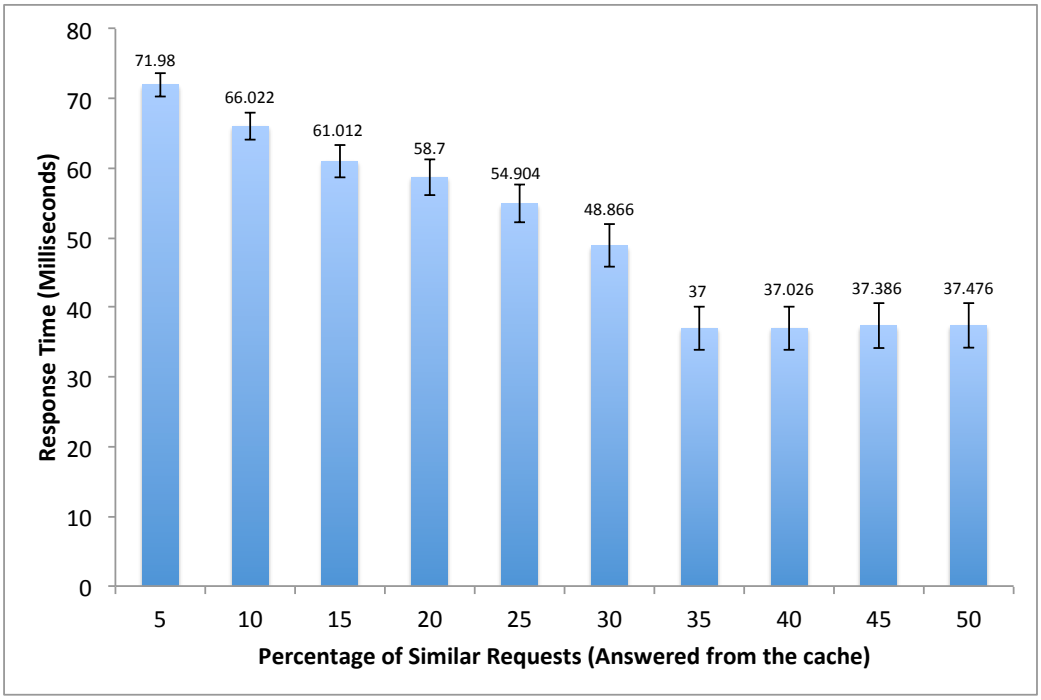


Figure 9: The gain in response time when CCNx cache is enabled and the request rate is at 15 requests per second. Confidence = 95%

In this experiment, we enabled CCNx caching. We wanted to see whether NextServe preserves the cache support of CCN. We study the caching effect when there are similar requests to the service provider with the same

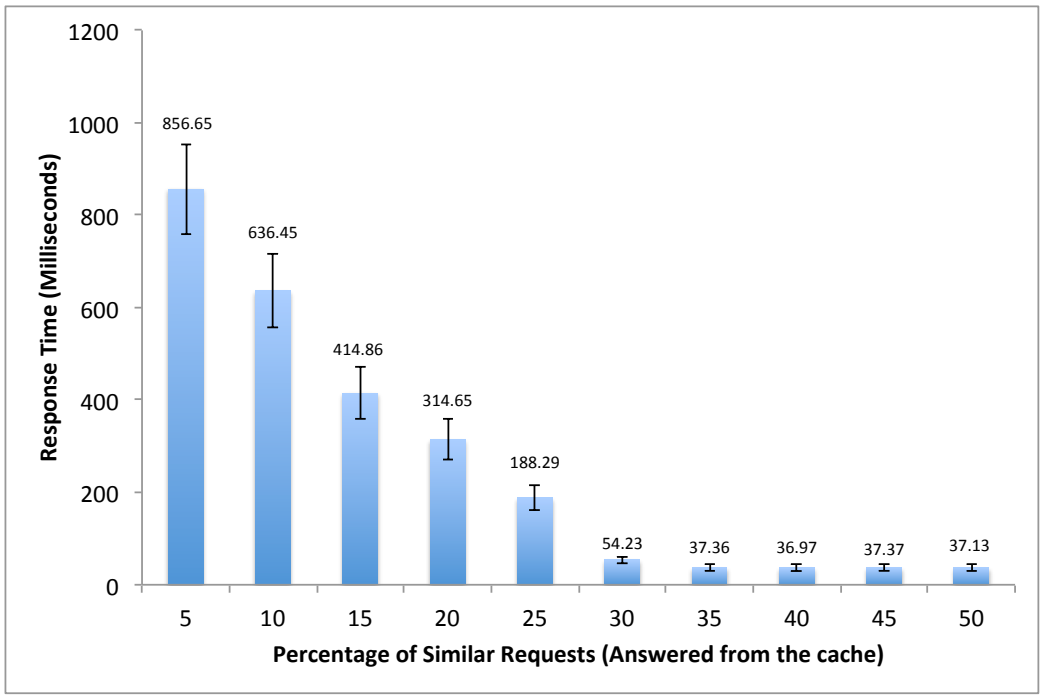


Figure 10: The gain in response time when CCNx cache is enabled and the request rate is at 25 requests per second. Confidence = 95%

parameter values. We fix the request rate and study the response time gain when there are similar requests issued by various clients. Those requests will be retrieved from the closest node, to the client, running “ccnd”. First, we fix the request rate at 15 requests per second, which is below the service throughput, and run the experiment for 10 seconds for each value of the percentage of similar requests. Figure 9 shows the averages for each data point with a confidence interval of 95%. We can see the decrease in response time when the percentage of similar requests increases. Also we notice that the response time hits a lower limit of around 37 milliseconds when the percentage of similar requests is 35%. There is no gain after that threshold because response time consists mostly of the CCN overhead.

More interestingly, when we set the request rate above the service throughput at 25 requests per second, the results in Figure 10 show that the response time follows an exponential decay. In other words, the more popular a service is, the more it benefits from CCN caching.

5 Related Work and Discussion

CCNxServ [7] is built to support services on top of CCN. However, CCNxServe does not allow for local parameters or service orchestration as supported in NextServe. Also it requires the service and the parameter content to have the same name prefix, introducing a major restriction on service implementation and publishing.
Serval [6] is an approach to service-centric networking, which provides special interfaces and layers in the TCP/IP stack to support service connections, sessions, allocation, and load balancing. Serval achieves high performance and mobility support, but it does not have any caching mechanism and it requires major changes in the TCP/IP stack.
To our knowledge, there are few, if any, evaluation studies of service support over CCN. On the other hand, performance of CCN was investigated through many experimental studies. Carofiglio *et al.* [12] show that content caching plays a significant role in the efficiency of content delivery in CCN. They also proposed that dynamic storage allocation based on the popularity of content enhances the storage efficiency in the CCN router cache. The effect of the popularity of content on the cache performance was also investigated by Rossi and Rossini [13]. Guimaraes *et al.* [14] showed that even though CCN introduces an overhead of 19% compared to TCP/IP, CCN outperforms TCP/IP when the number of consumers increases. The same results were reported by Jacobson *et al.* [1].
All the aforementioned principles and conclusions apply to NextServe because it does not change the CCN communication model but rather uses it as the underlying protocol for offering services.

6 Conclusions

In this paper, we presented the NextServe framework for supporting services over CCN. NextServe acts as a middleware layer over CCN and other conventional protocols. It allows for publishing services over CCN and for retrieving content parameters over CCN, FTP, HTTP, and any other URI-based content-name protocol. We showed how service orchestration can be achieved through the simple and user-friendly naming scheme of NextServe.

Performance of NextServe was evaluated regarding response time and cache utilization. We showed that NextServe adds very little overhead to CCN and benefits from the caching mechanism of CCN. This usage of caching enhances the performance time significantly when there are similar requests to the same service even when the request rate is higher than the service threshold.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[2] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, Aug. 2007.

[3] N. Fotiou, D. Trossen, and G. Polyzos, "Illustrating a publish-subscribe internet architecture," *Telecommunication Systems*, vol. 51, no. 4, pp. 233–245, 2012.

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

[5] T. Braun, A. Mauthe, and V. Siris, "Service-centric networking extensions," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, (New York, NY, USA), pp. 583–590, ACM, 2013.

[6] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, "Serval: an end-host stack for service-centric networking," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, (Berkeley, CA, USA), pp. 7–7, USENIX Association, 2012.

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

[8] C. Tschudin and M. Sifalakis, "Named functions for media delivery orchestration," in *Packet Video Workshop (PV), 2013 20th International*, pp. 1–8, Dec. 2013.

[9] D. Mansour, T. Braun, and C. Anastasiades, "Nextserve framework: Supporting services over content-centric networking," in *The 12th In-*

ternational Conference on Wired and Wireless Internet Communications, Springer, 2014.

[10] D. Crockford, "The application/json media type for javascript object notation (json)," RFC 4627, IETF, 7 2006.

[11] A. Carzaniga, M. Rutherford, and A. Wolf, "A routing scheme for content-based networking," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 918–928 vol.2, 2004.

[12] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental evaluation of memory management in content-centric networking," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1–6, June 2011.

[13] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatório técnico, Telecom ParisTech*, 2011.

[14] P. H. V. Guimaraes, L. H. G. Ferraz, J. V. Torres, D. M. Mattos, P. Murillo, F. Andres, L. Andreoni, E. Martin, I. D. Alvarenga, C. S. Rodrigues, *et al.*, "Experimenting content-centric networks in the future internet testbed environment," in *Communications Workshops (ICC), 2013 IEEE International Conference on*, pp. 1383–1387, IEEE, 2013.