Experiments with Subversion Over OpenNetInf and CCNx

Bengt Ahlgren
SICS
bengta@sics.se

Börje Ohlman
Ericsson Research
borje.ohlman@ericsson.com

Erik Axelsson
KTH
eaxe@kth.se

Lars Brown
KTH
lbrown@kth.se

ABSTRACT

We describe experiences and insights from adapting the Subversion version control system to use the network service of two information-centric networking (ICN) prototypes: OpenNetInf and CCNx. The evaluation is done using a local collaboration scenario, common in our own project work where a group of people meet and share documents through a Subversion repository.

The measurements show a performance benefit already with two clients in some of the studied scenarios, despite being done on un-optimised research prototypes. The conclusion is that ICN clearly is beneficial also for non mass-distribution applications.

1. INTRODUCTION

In the early days of the Internet the goal of most users was to explicitly establish connectivity between two nodes in the network and then run an application which would transfer some application-specific information. Today most Internet users are interested in getting access to a certain piece of information, not connecting to a specific node in the network, or using a particular application. The principal idea of *information-centric networking* (ICN) [1] is to put the information objects in focus instead of the interconnection of specific nodes. Then we can build a networking architecture that inherently can use a multitude of alternative ways to retrieve the desired information, including locally cached copies at nearby nodes, use of multiple access networks in parallel, information retrieval from broadcast networks like TV networks. Networks with intermittent connectivity and data mule networks also fit into this information-centric paradigm.

It is easy to argue that the ICN approach is ideal for large-scale information distribution. It is certainly true that the largest efficiency gain is expected for this type of application. In the work presented in this paper, we are instead investigating the benefit for another type of application which can take advantage of local ICN caching rather than always interacting with a remote server, making the application less dependent on good connectivity to that server. As the example application scenario we are using our own project meetings. A group of people in the same room or building is collaborating and sharing documents using a Subversion[1] version control repository. The repository is typically located in a different country and the available access network capacity is less than the demand.

We modified a Subversion server and client to use the communication service of the two ICN research prototypes OpenNetInf[2], from the 4WARD EU project, and CCNx[3], from the CCN project at PARC. We present experiences and insights from experimenting with the modified Subversion system, including evaluating the performance in the local collaboration scenario. A master thesis report [3] provides a more comprehensive description of the experiments.

The contributions of this paper are the experience from the adaptation of Subversion, and the initial performance measurements on the resulting prototypes. The measurements show a performance benefit already with two clients for common cases, despite the overhead of the non-optimised prototypes. One experience is that it was very easy to adapt the file fetching parts of Subversion to ICN.

The rest of the paper is organised as follows. In the next section, we describe the application scenario in more detail. Then follows a description of how we adapted Subversion to the ICN prototypes. We present the experimental measurement results in Section 4. Related work is covered in Section 5 and, finally, Section 6 concludes the paper.

2. APPLICATION SCENARIO

To introduce and motivate the problem space we start by describing a use case. A group of people is working together in a project. They find it convenient to use a collaboration environment that includes a common document repository (e.g., Subversion) and a Wiki. Both these tools are located at remote network servers. The group is in a meeting room using a WiFi network connected via an uplink to the Internet as illustrated in Figure 1. They use these common resources to access the same repository. This result in multiple uploads and downloads of the same document over the exter-

[1] http://subversion.apache.org/
[2] http://www.netinf.org/
[3] http://www.ccnx.org/

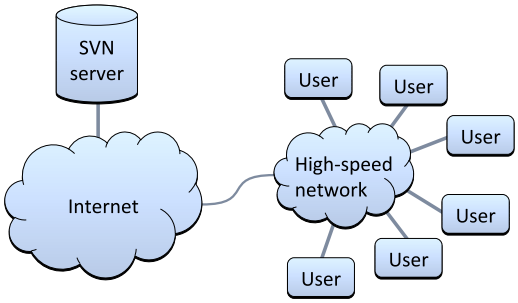

Figure 1: Local collaboration using Subversion.

nal link. Getting the same document from the same remote server multiple times is wasteful and often leads to performance problems, e.g., if the common uplink is a bottleneck.

In an information-centric network the network would in contrast provide the users with access to the nearest/best copy of a requested information object rather than offering remote access only to a specific copy located at a specific host at a remote site. In ICN, a node in the room that already has a document stored can make that document available also to the other nodes in the room. The other nodes can use any type of access to retrieve the document, e.g., WiFi, Bluetooth or infrared. This has the additional benefit of making the document available also to those users that currently lack direct connectivity to the global network infrastructure.

3. ADAPTING SUBVERSION TO ICN

In this section we introduce Subversion and the ICN prototypes, and describe how we adapted Subversion to ICN.

3.1 Subversion

The Subversion [4] version control system uses a centralised approach for sharing information. At the core of the system is the *repository*. It is a data storage which is structured in the same way as a file system tree, that is, a hierarchy of directories and files. Clients can add content to the repository so it becomes available to other clients, and clients can check out files from the repository in order to read them. The above description would fit to describe any regular file server. What makes a version control system different from a file server is that it keeps information about its own state and the changes to that state. This makes it possible for clients to request data from any given state, current or in the past. Providing this functionality in an efficient way is a prime objective of every version control system.

The basic client operations in Subversion are:

- **checkout:** Create a local copy of a certain version of a repository.
- **commit:** Upload changes in the local copy to the repository, creating a new version in the repository.
- **add:** Schedule to add a new local file or directory to the repository at the next commit.
- **update:** Update the local copy to the latest version in the repository.

3.2 Subversion over ICN implementation

This section describes the components used and the modifications made in order to adapt Subversion to the OpenNetInf and CCNx prototypes. In short, the Subversion commands which download information from the repository were adapted to be able to use OpenNetInf and CCNx for the content transfer. All other commands were left unmodified. Two protocol types, `icn+netinf://` and `icn+ccn://`, were added to be able to choose the ICN transports.

3.2.1 Server side: SVNJ

SVNJ[4] is a Java-EE servlet implementing server-side access to the Subversion repository, similar to `mod_dav_svn` for the Apache web server. SVNJ was modified in two ways. First, the data transfer commands were modified to return the OpenNetInf or CCNx object names to the client instead of the files. Second, functionality was added to make the files available through OpenNetInf and CCNx. The non-data-transfer commands were handled by a standard Apache-based Subversion server.

3.2.2 Client side: SVNKit

SVNKit[5] is an open-source Java toolkit that implements all Subversion client functionality, including a command line client program. We extended SVNKit to support file transfer with OpenNetInf and CCNx. The user decides which transport to use by specifying one of the new protocol types described above. For the `checkout` and `update` commands, SVNKit first communicates with the SVNJ server to get the OpenNetInf or CCNx object names for the files, and then retrieves the files using OpenNetInf or CCNx. All other commands are sent to the Apache-based Subversion server.

3.3 OpenNetInf prototype

OpenNetInf uses a flat name space for information objects [5]. The names have three fields: type, authenticator and label, where the authenticator is the hash of a public publisher key, and the label is chosen to be unique by the publisher, similar to names in DONA [9]. The names and locations of the information objects are registered in the NetInf name resolution service (NRS). To retrieve an object, a client first resolves its name using the NRS, and then retrieves a copy from one or more of the registered locations.

We made two modifications to the OpenNetInf prototype to support our scenario well. A parallel name resolution controller and a multicast-based resolution service have been implemented. The controller makes it possible to query multiple resolvers in parallel and return the answer from the fastest. The multicast-based service makes it possible to query nearby OpenNetInf nodes without any previous con-

[4] `http://code.google.com/p/svnj/`
[5] `http://svnkit.com/`

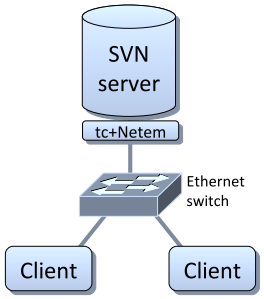

Figure 2: Experimental setup.

figuration or service discovery. When a node receives a multicast message the request is run through its local resolver. If a copy of the object is found, the node responds with an answer including itself in the list of locators. Subversion puts the repository URL in the NetInf label name field.

3.4 CCNx prototype

CCNx uses hierarchical names, somewhat similar to URLs, that are rooted in publisher prefixes. A client request, or *interest*, for an object is routed by the network towards the publisher using the object name. Each node on the path towards the publisher checks its cache for copies of the requested object. When a match is found, the object is returned on the reverse path of the request. All nodes along that path caches a copy of the object in case they get more requests for it.

We made no modifications to the CCNx prototype. The Subversion URL for a file is directly mapped to a corresponding CCNx name with the revision number added as a suffix. Every client node had two remote routes in its routing table, one for multicast communication and one for communication with the Subversion server.

4. EXPERIMENTAL RESULTS

In this section we present some measurements from the experiments with the two prototypes. The results show that the prototypes work as intended, and most importantly, that local copies of data are retrieved automatically by the ICN network service without application involvement. The absolute performance numbers should however not be taken too seriously – these research prototypes are far from optimised.

The experimental setup is shown in Figure 2. The Subversion server and the two client nodes are connected to the same gigabit Ethernet switch. The WAN link to the server is simulated using the Linux kernel traffic control (tc) mechanism together with the Netem queuing discipline. All nodes are Lenovo Thinkpad X100e with 1.6 GHz AMD Athlon NEO MV-40 CPUs running Ubuntu 10.10 Linux. The Subversion repository contained six 10 MB files.

Figure 3 shows the CPU and network utilisation at the server when the first client checks a repository out using the OpenNetInf communication service. During the first 20 seconds, the server makes the requested files available through

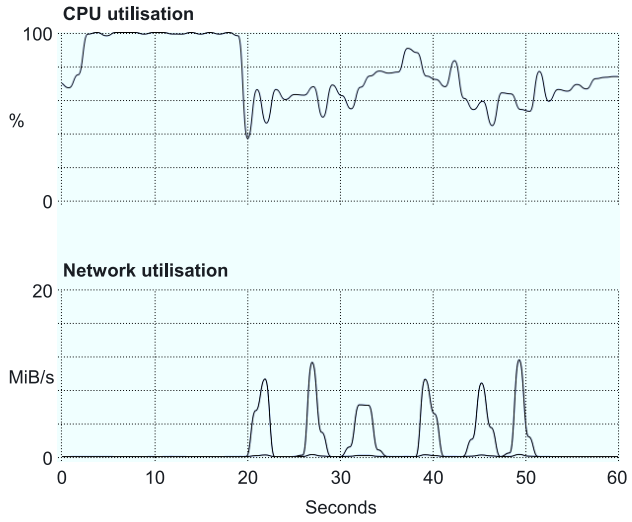

Figure 3: OpenNetInf CPU and network utilisation.

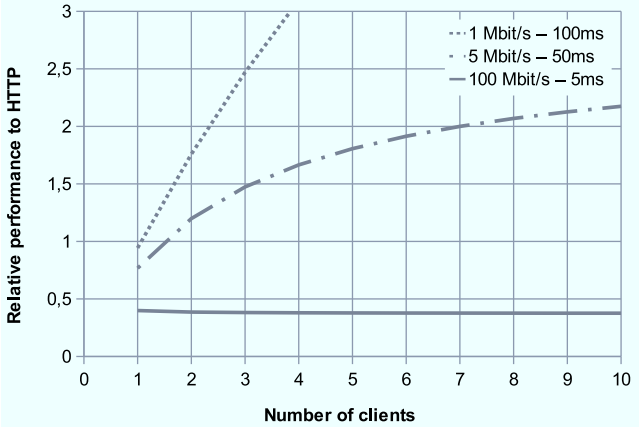

Figure 4: Subversion/OpenNetInf performance relative Subversion/HTTP.

the OpenNetInf service, resulting in close to 100% CPU utilisation. This includes calculation of cryptographic hashes and signatures of each file, and registration in the OpenNetInf name resolution system. This is a one-time cost per file that in the prototype is taken at the first request, but should of course be done in advance. During seconds 20-50 six files are transferred with good throughput to the client with short delays in between each. These delays are mostly due to signature verification at the client. The graphs clearly show the serial nature of the implementation.

The next two figures, 4 and 5, show the measured performance of a checkout operation using Subversion over OpenNetInf and over CCNx relative to the performance of legacy Subversion over HTTP. Values below 1 mean lower performance than legacy Subversion, and above mean higher. The three curves show the performance for different connection bit-rate and delay to the server, simulating different WAN characteristics for the path to the server.

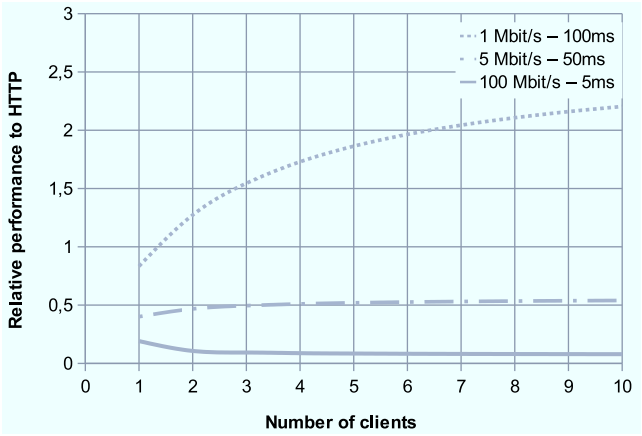

Figure 5: Subversion/CCNx performance relative Subversion/HTTP.

For Subversion/OpenNetInf, there is a performance gain for the 1 and 5 Mbit/s cases already when there are two local clients. For Subversion/CCNx there is only a gain for the 1 Mbit/s case due to higher base overhead of the prototype.

5. RELATED WORK

ICN is a new area of research, related work has been/is being done in EU projects 4WARD[6], SAIL[7], PSIRP[8], PURSUIT[9], COMET[10] and in the US projects DONA [9], CCN [7] and NDN[11]. A general overview of the ICN area initiatives is available in the ICN overview paper [1].

Some applications have already been developed that use ICN architectures. On the NetInf side Extensions to Mozilla Firefox and Mozilla Thunderbird called InFox and InBird have been developed with the OpenNetInf platform as their base [2]. On the CCN side applications include a simple chat (CCNChat), a file proxy (CCNFileProxy) application and voice-over-ccn (VoCCN) [6]. The PSIRP/PURSUIT prototyping have primarily focused on evaluating performance of a pure publish/subscribe architecture [8].

To our knowledge this work is the first where the same application has been implemented on two ICN platforms and a comparative study between the two has been made.

6. CONCLUSIONS

We have adapted a Subversion server and client to use the network service of the ICN prototypes OpenNetInf and CCNx. Experiments were done in a local collaboration scenario to evaluate the benefit of ICN for applications that can take advantage of local caching to lessen the need for good connectivity to a server.

[6] http://www.4ward-project.eu/
[7] http://www.sail-project.eu/
[8] http://www.psirp.org/
[9] http://www.fp7-pursuit.eu
[10] http://www.comet-project.org
[11] http://www.named-data.net/

It was straightforward to adapt the file transfer parts of Subversion to use ICN. It is clear that ICN has a higher base overhead than current network protocols, but the experiments revealed that the gain quickly offsets the costs. The measurements show a performance advantage for ICN already with two clients for realistic simulations of the WAN connection to the server, despite the systems being unoptimised research prototypes. The experiments also revealed that there are good opportunities for optimising the prototypes by parallelising the computation of the hashes and signatures, lessening the impact of these overheads. Automatic selection of the 'best' source is crucial for good ICN performance, a function the prototypes currently lack.

While ICN can make communication more efficient for the same content distribution scenarios that motivate the use of CDNs, the local collaboration scenario of this paper and similar scenarios that approach disruption-tolerant networking (DTN) provide additional motivation for ICN.

Acknowledgements

This work was partly supported by the EU FP7 project SAIL (no 257448) and partly by SICS Center for Networked Systems funded by VINNOVA, KKS, SSF, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus and Peerialism.

7. REFERENCES

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking (draft). In *Information-Centric Networking*, number 10492 in Dagstuhl Seminar Proceedings, 2011.

[2] B. Ahlgren et al. Netinf evaluation. Deliverable D-6.3, 4WARD EU FP7 Project, June 2010. FP7-ICT-2007-1-216041-4WARD / D-6.3, http://www.4ward-project.eu/.

[3] L. Brown and E. Axelsson. Use of information-centric networks in revision control systems. Master's thesis, Royal Institute of Technology (KTH), Jan. 2011.

[4] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato. *Version Control with Subversion*. O'Reilly Media, 2004.

[5] C. Dannewitz, J. Golić, B. Ohlman, and B. Ahlgren. Secure naming for a network of information. In *13th IEEE Global Internet Symposium*, San Diego, CA, USA, Mar. 19, 2010.

[6] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard. VoCCN: Voice-over content-centric networks. In *Re-Architecting the Internet (ReArch'09)*, Rome, Italy, Dec. 1, 2009. A CoNEXT 2009 workshop.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. L. Braynard. Networking named content. In *Proc. ACM CoNEXT*, Rome, Italy, Dec. 1-4, 2009.

[8] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: Line speed publish/subscribe inter-networking. In *Proc. ACM SIGCOMM*, Barcelona, Spain, Aug. 17-21, 2009.

[9] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 27-31, 2007.