Design of Communication Architecture to Support Stream Data over Content-centric Networking

Kenya Kawasaki
Graduate School of Information Science and Technology
Osaka University
1-5 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
k-kawasaki@ist.osaka-u.ac.jp

Shingo Ata
Graduate School of Engineering
Osaka City University
3-3-138 Sugimoto, Sumiyoshi-ku, Osaka-shi, Osaka 558-8585, Japan
ata@eng.osaka-cu.ac.jp

Masayuki Murata
Graduate School of Information Science and Technology
Osaka University
1-5 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
murata@ist.osaka-u.ac.jp

ABSTRACT

The future Internet is expected from the evolution of today's IP networks. Content-centric networking (CCN) is a promising future communication architecture, and there has been much research into its applications and architecture. However, the most of communications in CCN focus on the delivery of individual content by using application specific interest-data exchanges. There are few studies on a generic framework for the delivery of multimedia data, which contains a series of content generated periodically.

In this paper, we design and implement a system to support a new type of communication model called *stream data*, which covers a various types of multimedia data such as streaming and sensing, and its form of delivery over CCN. To obtain two important features of *stream data*, which are random access and flexibility, we design a CCN architecture that includes a seamless naming/addressing architecture to enable content provider controls. In particular, we target embedded devices, which have limited hardware resources, to realize easy *stream data* delivery. In addition, we develop a prototype to verify the feasibility of our proposed architecture.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Network Architecture and Design

1. INTRODUCTION

The Internet was designed about 50 years ago, when it was mainly targeted on the communication between computer terminals. Therefore, the design of the Internet focuses on the location of communication peers. However, by the spread of the World Wide Web (WWW), Internet users are primarily interested in content instead of the location of terminals, while the communication is still host-based.

To address the problems raised by such gap, the future Internet is expected to be a post IP networks, where Content-centric networking (CCN) or Information-centric networking (ICN) is promising as such a future communication architecture. There are many research projects that have examined CCN. For example, PURSUIT [1], SAIL [2], COMET [3], DONA [4], NDN [5], CCNx [6], and MobilityFirst [7]. In this paper, we focus on CCNx/NDN architecture.

CCN distributes content by exchanging Interest packets and Data packets. When a user wants to obtain content, the user sends an Interest packet that includes the content name to networks. The router that receives an Interest packet transmits it according to the content name. A content server that receives an Interest packet sends back the content corresponding to the Interest packet to the user as Data packet. Thus, all routing are performed with content-based, so that CCN achieves content-centric communication based on the content name.

There are many studies that implement various applications of CCN, in particular, media streaming and machine-to-machine (M2M) applications. For media streaming, voice conference [8], and video on-demand services [9, 10, 11, 12] have been developed. In addition, there are many applications in M2M environments [13, 14, 15]. However, most of these studies only implement application-specific protocols and signalings and do not tackle the design of general architecture for streaming, M2M, and other applications.

The main motivation behind this paper is that we design a new CCN architecture that can be used to support such applications in general. Specifically, we first define a communication model called *stream data* which includes various types of application data, e.g., multimedia streaming, M2M communications, information collections from sensors, and so on. We then design a communication architecture for supporting *stream data* delivery. After that we implement a prototype of *stream data* communication in case of a wireless sensor network. As a proof-of-concept, we demonstrate our implementation and experiment with embedded devices.

The organization of this paper is as follows. In Section 2, we design the naming architecture and actions at each node to realize *stream data* delivery over CCN. In Section 3, we describe an implementation of our *stream data* distri-

CFI '15, June 08 - 10, 2015, Seoul, Republic of Korea
© 2015 ACM. ISBN 978-1-4503-3564-5/15/06…$15.00
DOI: http://dx.doi.org/10.1145/2775088.2775091

bution system. In Section 4, we verify the effectiveness of the system through experiments. Finally, we describe our conclusions in Section 5.

2. DESIGN OF THE STREAM DATA DISTRIBUTION SYSTEM ON CCN

2.1 Basic principle

In this section, we outline the *stream data* distribution system to support various types of applications on CCN. We define *stream data* as a series of contents with a sequence generated over time by a single source (content provider). For example, *stream data* can be video and/or audio media data, and monitoring data from periodic sensor monitoring. *Stream data* contains a sequence of content, and thus retrieving *stream data* means retrieving multiple pieces of contents. Generally, there are two methods for retrieving sequential contents.

- Sending control messages as Interest packets: When the client starts content retrieval it sends an Interest packet to request "start data retrieval". The server that receives the Interest packet starts to send a sequence of contents continuously. After that, if a client wants to stop the retrieval, it sends an Interest packet to request to stop data retrieval.

- Sending Interest packets to retrieve contents by themselves: The client sends an Interest packet for each piece of content. The node that receives the Interest packet sends the corresponding content. The number of Interest packets would be the same as the number of pieces to request.

The first method can reduce amount of Interest packets. However, the client and server have to manage the session and content retransmission. In addition, CCNx architecture does not allow an Interest packet to request multiple Data packets [16]. The second method does not need to manage the session and content retransmission. The client sends more Interest packets than in the former method, although the client can retrieve part of the *stream data* easily. Therefore, we adopt the latter way for *stream data* communication. However, the former way can be realized if we can deploy some extensions in CCN nodes (end terminals and CCN routers). For *stream data* communication, each CCN node requires the following operations.

- **Client** Clients create Interest packets to identify the content they want to retrieve. This means that clients need to know the content publisher, publishing time, and content order and its quality (if the quality of contents is adjustable).

- **Router** When a router receives Interest packets, the router looks up a content cache based on the content name in the Interest packets. If content corresponding to the Interest packet is found, the router sends back the content to the client as a Data packet. Otherwise, the router transmits the Interest packet to the next node based on the content name. In this way, a Forwarding Information Base (FIB) is used to decide the next neighbor router to forward. When a router receives Data packets, the router transmits the Data packets to the client, it is based on a dynamic state stored in a Pending Interest Table (PIT).

- **Server** When a server have a content to share, the server names the content and publishes the content to the network, where an entry for the content is created in CCN routers' FIB. When a server receives an Interest packet that requests content stored on the server, the server sends back the content corresponding to the Interest packet as a Data packet to the client.

2.2 Name structure for stream data

CCN controls the route by the content name. Therefore, the content name structure affects the performance of the method. In addition to routing, the content name should also accurately represent the information in the content. To satisfy the requirement described in Subsection 2.1, we consider content name elements as follows.

- **Routing prefix** Routing prefix is the classified name to aggregate content information. For example, sensor information includes the node location. To append this information to the prefix, routing information for the sensors in the same area can be aggregated. If the content does not depend on the specific node, grouping associated content for a prefix can make routing more efficient. In addition, a prefix is used to avoid the collision of object names.

- **Identifier, Version** Identifier is used to identify the object. If the object is allowed to update content, it can be appended a version to the updated content.

- **Control** Control is the information for controlling the quality of the *stream data*. For example, it controls the camera resolution, bit rate, and sampling rate. Specifying the quality through the control allows the user to receive *stream data* that is suitable for their environment.

- **Sequence** Sequence is the information that represents the order of chunks. The sequence does not have to be evenly incremented number; it can be the frame number of a video or a timestamp in sensor data.

In the name elements, control and sequence can be provided by publisher of the *stream data*, but they are unknown to clients initially. Thus, we use *metadata* defined per *stream data* for getting these pieces of information.

- **Metadata** Metadata contains a set of attribute information for the *stream data*. The name of the metadata can be determined by the *stream data* that it corresponds to. Metadata consists of key-value pairs. The client requests this information with a name element to obtain a key, and then the value corresponding to the key is returned. For example, to retrieve a video, the client first gets the metadata. The keys in metadata include information about the codec, frame rate, and bit rate. The client constructs the video content's full name using these pieces of information, and begins the retrieval of the *stream data*.

Table 1 summarizes the name structure *stream data* and its information retrieval.

Table 1: Name structure of stream data

Content type	Naming
Content	/⟨routing prefix⟩/⟨identifier⟩/⟨version⟩/⟨control⟩/⟨sequence⟩
Metadata	/⟨routing prefix⟩/⟨identifier⟩/⟨metadata-name⟩

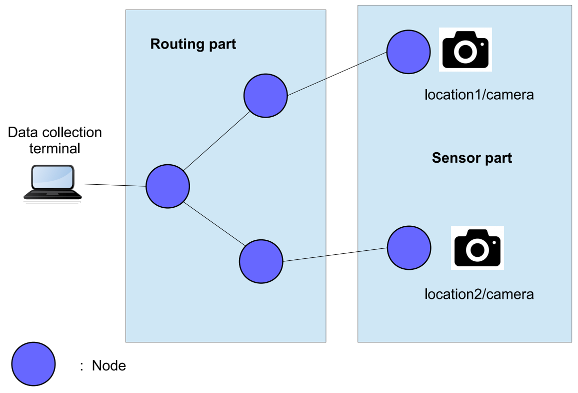


Figure 1: Overview of wireless sensor network.

3. IMPLEMENTATION OF STREAM DATA DELIVERY IN WIRELESS SENSOR NETWORK OVER CCN

3.1 Implementation overview of the wireless sensor network

Figure 1 shows the implementation overview of our experimental wireless sensor network to demonstrate a proof-of-concept of our proposed *stream data* delivery. The network contains nodes, sensors, and a data collection terminal. We consider cameras as the sensors, and images that are captured by cameras as sensing information. Nodes are classified into two categories: sensing nodes having a sensor, and routing nodes that handle routing packets. Sensing nodes generate content periodically from images obtained from the attached camera, and thus act as the publisher. The routing nodes route Interest packets and Data packets. The data collection terminal retrieves sensing information from sensing nodes.

In this paper, embedded system platforms are used for nodes. In this experiment, due to the limitation of the storage, do not archive sensing information and always aim to retrieve the latest sensing information.

3.2 Equipment

We use Armadillo-420 embedded system platforms (Atmark-techno) for the nodes. Figure 2 shows a picture of the Armadillo-420. As shown in Table 2, Armadillo-420 contains a 400 MHz ARM-based processor, 64 MB LPDDR SDRAM, and a 16 MB flash memory. Linux 2.6 is installed as an operating system, and we develop and deploy a set of userland programs to realize our framework.

We attach USB Web cameras to the Armadillo-420, and

Table 2: Hardware specifications of the Armadillo-420

Processor	Freescale i.MX257
CPU core	ARM926EJ-S
CPU core clock	400 MHz
Bus clock	133 MHz
RAM	64 MB (LPDDR SDRAM)
Flash memory	16 MB (NOR type)
Wireless LAN	Supporting IEEE 802.11b/g/n
USB	USB 2.0 × 2 (high speed × 1, full speed × 1)

Figure 2: Photograph of Armadillo-420

an MJPG (Motion JPEG) streamer to obtain the images generated by the camera. The MJPEG streamer generates a sequence of JPEG images as a video by HTTP (Hyper Text Transport Protocol).

We use the CCNx protocol suite [6] which is an implementation of CCN. The CCNx constructs an overlay network for CCN communication over the IP network. Therefore, a CCN network with an arbitrary topology can be constructed over a physical network topology.

3.3 Implementation in wireless sensor network

Sensing nodes run mjpg_streamer, obtain the image from the attached camera as a JPEG, and create the content per image frame. The content name is shown in Table 3. Each element in Table 3 corresponds to the name structure designed in Subsection 2.2.

- **Routing prefix** The fixed name "ccnx:/osaka-u.ac.jp" is used for the routing prefix.
- **Location name** The location name represents the node's physical location and the place where the images are recorded.
- **Sensor type** Sensor type corresponds to the identifier described in Subsection 2.2. Because we use cameras for the sensor, the name of the sensor type is "camera".
- **Version** Version includes the timestamp when start sensing. During sensing, the sensor nodes generate content with the same version. If a sensor node stops

Table 3: Structure of content name

Content type	Naming
Content	/⟨routing prefix⟩/⟨location name⟩/⟨sensor type⟩/⟨version⟩/⟨data format⟩/⟨frame number⟩
Metadata (format)	/⟨routing prefix⟩/⟨location name⟩/⟨sensor type⟩/⟨metadata-name⟩
Metadata (frame)	/⟨routing prefix⟩/⟨location name⟩/⟨sensor type⟩/⟨version⟩/⟨data format⟩/⟨metadata-name⟩

and resumes sensing, it generates content with a new version.

- **Data format** The format of the image data corresponds to this information. mjpg_streamer can generate a JPEG image of a specified resolution and frame rate, so this information includes resolution, frame rate, and data format. Due to the hard ware limitation of the embedded system, the resolution is specified as QSIF (176 × 112 pixels) or QCIF (176 × 144), the frame rate is 1 fps, and the data format is JPEG.

- **Frame number** Frame numbers are sequential integers appended to the images generated by the cameras.

- **Metadata** In the wireless sensor network implemented in this paper, the sensor location and type are known, and the version, data format, and frame number are not known by the data collection terminal. Thus, we define two types of metadata to construct the content's full name: the metadata for control and the metadata for the sequence. When the data collection terminal starts retrieval of the images, it obtains the available control and newest sequence (frame number), and constructs the content's full name with these pieces of information.

4. EXPERIMENTAL EVALUATIONS

In this section, we run a scenario where a client obtains images recorded with cameras to verify that the wireless sensor network implemented in Section 3 performs correctly. We describe the environment, scenario, and results.

4.1 Environment

Armadillo-420 platforms and PC are connected with a wireless link, and we construct a CCNx overlay network over this network. Two Armadillo-420 platforms have one camera each, and generate contents from recording images. To request the image recorded by one of these cameras and to ensure that the newest image frame is returned, we verify that the content name with the newest frame can be searched. Next, to request the image recording by another camera and to change the returned image seamlessly, we examine the dynamic performance for changing the content quality.

Figure 3 shows the test environment. We deploy three Armadillo-420 platforms at Floors A6F and A610, and connect them to the data collection terminal (PC) with an ad-hoc wireless link. Two Armadillo-420 platforms with cameras are sensing nodes, and the other Armadillo-420 is the routing node. The content names for retrieving *stream data* are defined in Table 4.

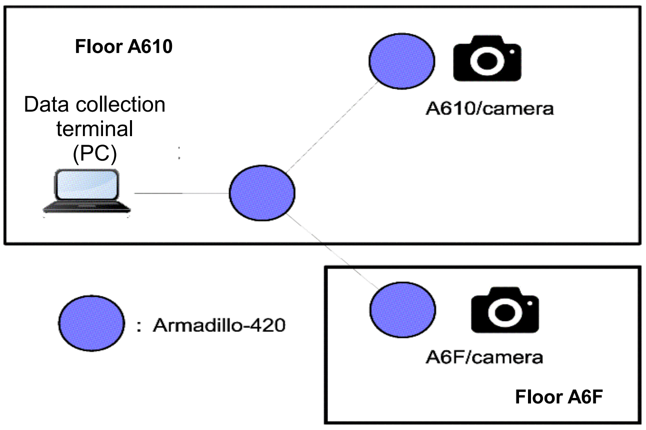


Figure 3: Experimental Environment

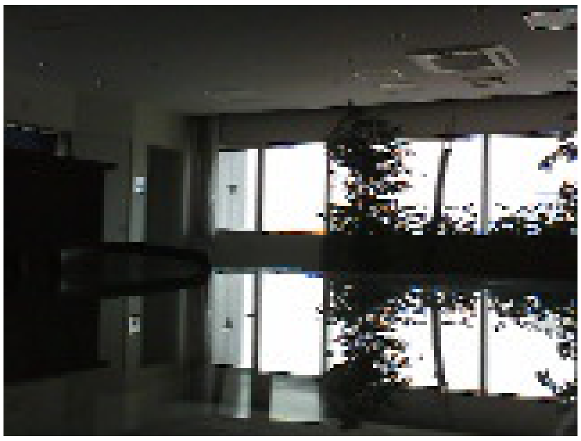
Table 4: Generating content name

Sensor location	Content name
Floor A6F	ccnx:/osaka-u.ac.jp/A6F/camera/⟨version no.⟩/jpg/(QSIF/QCIF)/1/⟨frame number⟩
Floor A610	ccnx:/osaka-u.ac.jp/A610/camera/⟨version no.⟩/jpg/(QSIF/QCIF)/1/⟨frame number⟩

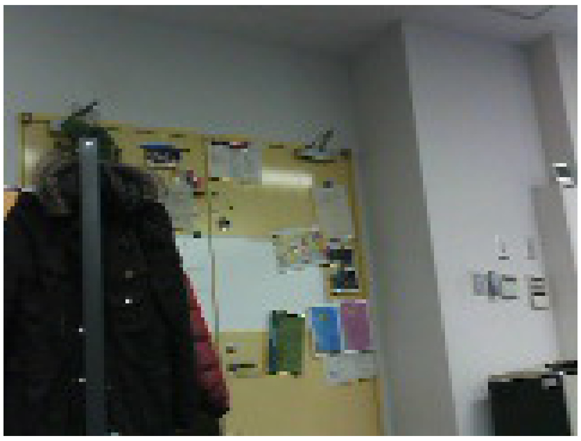
4.2 Experimental scenario

First, the sensor nodes generate metadata, including the version, data format, resolution, and frame rate, and name it "ccnx:/osaka-u.ac.jp/⟨location name⟩/camera/metadata." Next, sensor nodes generate content from each frame of the recorded image with the names shown in Table 4. Sensor nodes also generate metadata that includes the newest frame number, and name it "ccnx:/osaka-u.ac.jp/⟨location name⟩/camera/⟨version no.⟩/jpg/(QSIF/QCIF)/1/metadata."

Then, the PC retrieves the images recording on Floor A6F. First, to obtain the version, data format, resolution, and frame rate, the PC requests content named "ccnx:/osaka-u.ac.jp/A6F/camera/metadata." The PC constructs the content name from the previously returned information, and requests content named "ccnx:/osaka-u.ac.jp/A6F/camera/⟨version no.⟩/jpg/(QSIF/QCIF)/1/metadata" to obtain the newest frame number. To construct the full content name from the previous returned information and request the content with the full name, the PC retrieves the newest image frame. If the PC retrieves the frame correctly, the frame number increases incrementally and it requests the next frame. From these scenarios, we can check that the newest content name can be searched and the client can retrieve the *stream data*. In the network implemented in this paper, client cannot retrieve past data because there is no storage. However, if we implement storage, the client could search frames generated at arbitrary times from the version, frame



(a) Image retrieved from the camera on Floor A6F



(b) Image retrieved from the camera on Floor A610

Figure 4: Image retrieved by PC

rate, and frame number information.

Following these operations, the PC retrieves an image frame recorded on Floor A610. To perform the same operation as described above, we change the image to recording at Floor A610.

4.3 Results

Figure 4(a) shows the image retrieved from Floor A6F. To increase the sequence number, we confirmed that the PC retrieves the next frame continuously. Figure 4(b) shows the retrieved image when requesting the content name change to A610. Switching the camera that is publishing images requires changing only the requested content name.

These results show that the system can access the newest content and can change the retrieval stream dynamically by changing only the requested content name.

5. CONCLUSION AND FUTURE WORK

In this paper, we designed a streaming system that can be used for applications over CCN. We discussed the general system architecture to allow accessibility to arbitrary data and variability in the quality of stream data. Then, we implemented a wireless sensor network based on our system and showed that a client can obtain images continuously from cameras. In addition, we showed that changing only the requested content name can change the retrieving frame dynamically.

In the future, we will implement data storage, evaluate performance in a large system, and evaluate performance for multiple clients.

Acknowledgement

This work was supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Japan.

6. REFERENCES

[1] N. Fotiou, P. Nikander, D. Trossen, and G. Polyzos, “Developing information networking further: From PSIRP to PURSUIT,” in *Broadband Communications, Networks, and Systems*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, I. Tomkos, C. Bouras, G. Ellinas, P. Demestichas, and P. Sinha, Eds. Springer Berlin Heidelberg, 2012, vol. 66, pp. 1–13.

[2] B. Ahlgren, P. Aranda, P. Chemouil, S. Oueslati, L. Correia, H. Karl, M. Sollner, and A. Welin, “Content, connectivity, and cloud: ingredients for the network of the future,” *IEEE Communications Magazine*, vol. 49, no. 7, pp. 62–70, July 2011.

[3] G. Garcia, A. Beben, F. Ramon, A. Maeso, I. Psaras, G. Pavlou, N. Wang, J. Sliwinski, S. Spirou, S. Soursos, and E. Hadjioannou, “COMET: Content mediator architecture for content-aware networks,” in *Proceedings of Future Network & Mobile Summit 2011*, June 2011, pp. 1–8.

[4] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A data-oriented (and beyond) network architecture,” *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 181–192, October 2007.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *Proceedings of 5th International Conference on Emerging Networking Experiments and Technologies*, December 2009, pp. 1–12.

[6] “CCNx,” PARC, 2013. [Online]. Available: http://www.ccnx.org/

[7] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, “MobilityFirst: A robust and trustworthy mobility-centric architecture for the future internet,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, Decenber 2012.

[8] 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 *Proceedings of the 2009 workshop on Re-architecting the Internet*, December 2009, pp. 1–6.

[9] D. Kulinski and J. Burke, “NDNVideo: Random-access Live and Pre-recorded Streaming using NDN,” NDN, *Technical Report NDN-0007*, September 2012. [Online]. Available: http://www.named-data.net/techreport/TR007-streaming.pdf

[10] A. Difino, R. Chiariglione, and G. Tropea, “Video Services in Information Centric Networks: Technologies and Business Models,” *Infocommunications Journal 2013*, vol. 5, pp. 1–9, 2013.

[11] B. Han, X. Wang, N. Choi, T. T. Kwon, and Y. Choi, “AMVS-NDN: Adaptive Mobile Video Streaming with Offloading and Sharing in Wireless Named Data Networking,” in *Proceedings of IEEE INFOCOM NOMEN workshop 2013*, April 2013.

[12] Y. Liu, J. Geurts, J.-C. Point, S. Lederer, B. Rainer, C. Muller, C. Timmerer, and H. Hellwagner, “Dynamic adaptive streaming over CCN: A caching and overhead analysis,” in *Proceedings of 2013 IEEE International Conference on Communications (ICC)*, June 2013, pp. 3629–3633.

[13] Z. Ren, M. Hail, and H. Hellbruck, “CCN-WSN - a lightweight, flexible content-centric networking protocol for wireless sensor networks,” in *Proceedings of 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Apr 2013, pp. 123–128.

[14] L. Grieco, M. Ben Alaya, T. Monteil, and K. Drira, “Architecting information centric ETSI-M2M systems,” in *Proceedings of 2014 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, March 2014, pp. 211–214.

[15] B. Saadallah, A. Lahmadi, and O. Festor, “CCNx for Contiki: implementation details,” INRIA, Technical Report RT-0432, November 2012. [Online]. Available: http://hal.inria.fr/hal-00755482

[16] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, kc claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh, “Named Data Networking (NDN) Project,” NDN, *Technical Report NDN-0001*, 2010.