# Multicast Service Deployment using Information Centric Cloud-Native Network Functions

Yusaku Hayamizu and Hitoshi Asaeda
National Institute of Information and Communications Technology (NICT), Japan
{hayamizu, asaeda}@nict.go.jp

***Abstract*—One-to-many or many-to-many communication is certainly beneficial for various network services, such as online meetings and entertainment live-streaming. The multicast paradigm is one of the most effective communication techniques for enabling such services. However, owing to its various limitations, it is not yet widely deployed to the Internet. For simplified and effective multicast service deployments, we propose Information-Centric Cloud-Native Network Function (IC-CNF), which can be used to quickly create a multicast island in which various applications will benefit from a variety of multicast functionalities. To validate the effectiveness of IC-CNF, we implemented an actual system using an emerging microservice technology, i.e., Docker and conducted real-world experiments. The experimental results show that the proposed IC-CNF significantly reduces 4K live video streaming traffic by 71.6% compared with IP-based unicast networking. Conceptually, multiple multicast islands created by IC-CNF can be deployed and connected to the Internet; thus, a wider multicast network can be organized while maintaining efficiency.**

***Index Terms*—ICN, CCNx, NDN, Information-Centric Networking, Named-Data Networking, Cefore, deployment, video streaming, Cloud-Native Network Function, CNF**

## I. Introduction

Multicasting is the most effective networking technology for one-to-many or many-to-many communication services such as remote meetings and live video streaming. The internet protocol (IP) multicast is a representative paradigm, and its effectiveness has been demonstrated in various research domains. However, their IP multicast solutions have certain shortcomings [1]–[3], such as improved routing protocol scalability to support complex routing coordinates and configurations, which otherwise entails high resource and operational costs. Recently, the Internet Engineering Task Force (IETF) revisited the IP multicasting deployment barrier and analyzed how this emergent technology can be deployed from the protocol perspectives [4], yet it is tough to be realized. Application layer multicast [5] and peer-to-peer (P2P) communications do not require significant protocol changes in the network layer; hence, they have gained considerable attention in terms of multicast support improvements [6], [7]. Nevertheless, it is difficult to establish and maintain optimized forwarding paths for various types of dynamic multicast services and participants .

The concept of Information-Centric Networking (ICN) [8] has been advanced to several architectures, such as Content-Centric Networking (CCNx) [9], [10] and Named-Data Networking [11]. Fundamental research and experimentation have been widely conducted [12], and various efforts have been extended to provide improved routing, forwarding, caching, and security methods. As such, a consensus on designs and protocols seems imminent. One of the most noteworthy features of ICN is its multicast functionality at the network layer, which is achieved using name-based communications. ICN was originally designed as a clean-slate network architecture for the future Internet; however, it is now viewed as a practical network architecture running atop the IP network as an overlay technology [13], [14]. Deployment studies on Internet-of-Things (IoT) sensor networks [13] and 4G Long-Term Evolution (LTE) networks [15] have focused on ICN deployments to last-mile or edge areas. This assumption is reasonable because deploying an ICN to an edge network is easier than deploying it to a core network, owing to the low initial costs [16].

Recent cloud-native ecosystems, e.g., Docker [17] and Kubernetes [18], are emerging microservice technologies that hold promise for edge and fog computing solutions. Docker containers provide numerous benefits for software resource operations and management, including portability, performance, agility, and scalability. After several years of experimentation and deployments, ICN has not been employed on the Internet yet, despite its promised effectiveness; however, ICN software platforms such as *Cefore* [19], [20], which is a CCNx-1.0-compliant implementation, are integrated as microservice technologies that offer a quick and simple ICN construction and deployment capability as network services.

In this paper, we design and implement a deployable ICN framework for IP networks using emerging cloud-native network softwarization technologies. We first discuss the system design and propose a new *multicast island* concept for extant edge computing platforms to facilitate multicasting at edge nodes while reducing video traffic bandwidth[1]. To this end, we integrate Cefore with Docker technologies and provide our novel Information-Centric Cloud-Native Network Function (IC-CNF) platform to deploy ICN functions as microservices for creating multicast islands with all of the inherent benefits. To validate the effectiveness of our proposed deployable IC-CNF, we implement an actual system using Docker and conduct real-world experiments. The results demonstrate that

[1]As a proof-of-concept, we especially highlight the features of video streaming applications; however, our proposal can be also applied to general applications based on one-to-many or many-to-many communication models.

the proposed approach significantly reduces traffic loads at the server, compared with IP-based unicast-only live streaming.

The remainder of this paper is organized as follows. Section II explains the basics of the ICN and Cefore software platforms. Our deployable ICN framework is then proposed, and our ICN/Cefore integration concept with Docker technologies is explained in Section III. Section IV presents the experimental results obtained using a real testbed. Finally, in Section VI, we conclude this paper by discussing future directions.

## II. Background Technology

### A. ICN: Information-Centric Networking

Figure 1 presents an overview of the ICN communication model, which adopts a receiver-driven method in which a consumer (user) transmits an Interest (request) packet to retrieve a ContentObject (data) packet from producer/publisher (content servers). The Interest packet contains the name of the required content, e.g., “video.mp4”, and the ContentObject packet contains the corresponding name of the Interest packet. These packets are forwarded by intermediate routers using name-based forwarding mechanisms.

A Forwarding Information Base (FIB) is used for Interest packet forwarding. When a router receives an Interest packet and decides to forward it to an upstream neighbor based on an FIB lookup, it records the incoming interface (face) of the Interest packet in a Pending Interest Table (PIT) to forward the corresponding ContentObject packet. For ContentObject packets, PIT is used for forwarding by recording the reverse paths of the transferred Interest packets. The intermediate routers in the data path store the ContentObject packets in their Content Store (CS) based on their caching policies. The in-network cache can then be used for future requests from other users.

Notably, one of the most important/interesting features of ICN is “multicast” at the network layer. Hence, when more than one consumer requests the same content, the Interest packets are aggregated by a branching router. When the corresponding ContentObject packets is sent from the producer, the ContentObject packet is replicated at the router and forwarded to multiple consumers while referring to the PIT trials. Using this multicast technique, ICN is expected to reduce the traffic load at producer and core networks.

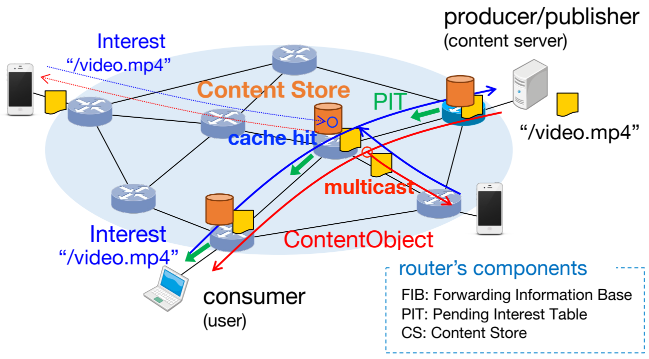

Fig. 1. An overview of ICN communications model.

### B. Cefore: CCNx-based extensible packet forwarding engine

To realize ICN communications, the Cefore [19] open-source software platform was developed.

*1) Core component:* Cefore’s packet format is compliant with CCNx-1.0, which was standardized by the Internet Research Task Force (IRTF) RFCs 8549 and 8609 [9], [10]. Cefore emphasizes the following three design policies:

- Light weight: the software implementation must be compact, and the platform should be usable by resource-constrained devices, such as sensor nodes.
- Usability: the platform should be easily configured, set up, reloaded, and connected to the experimental environments. Ideally, its emulation/simulation capacity will be easily applied and tested using real network equipment.
- Extensibility: the platform should be easily extensible to accommodate the novel functions that satisfy future network requirements.

To satisfy these policies, Cefore handles Interest and ContentObject packets using the *cefnetd* core forwarding daemon, which contains a minimum set of functions (i.e., FIB and PIT), making it lightweight and scalable. Other compute-intensive functions, such as in-network caching and computing, are implemented using plugins or external daemons for extensibility and usability. Csmgrd is a CS daemon to which cefnetd can connect via the transmission control protocol or a local socket.

*2) Plugin extension:* Figure 2 shows an overview of Cefore’s pluggable architecture. In addition to these core components, Cefore is easily customizable by adding plugin libraries. Therefore, researchers or network administrators can develop and configure new mechanisms for cefnetd and/or csmgrd without modifying their codes. For example, cefnetd can be equipped with the *forwarding strategy* and *transport* plugins to receive ICN benefits, such as multi-path or multi-source communications. Csmgrd has a *cache plugin* that selects its cache eviction method, e.g., First-In-First-Out (FIFO), Least Recently Used (LRU), and Least Frequently Used (LFU) depending on the operator’s requirements.

*3) Utilities:* The Cefore software package includes useful utilities such as `cefputfile` and `cefgetfile` for uploading and downloading data, respectively. The CCNinfo, a CCNx network operation and management tool, is also included (see specifications in [21]). In the current implementation, all communications are performed over TCP/UDP as an IP overlay.

## III. IC-CNF: Information-Centric Cloud-Native Network Function

### A. System Design

In this section, we propose a deployable ICN framework for provisioning ICN/Cefore nodes as cloud-native functions into edge networks. Figure 3 illustrates our assumed scenario in which an ICN router (Cefore) is deployed to provide microservices function at the edge computing infrastructures

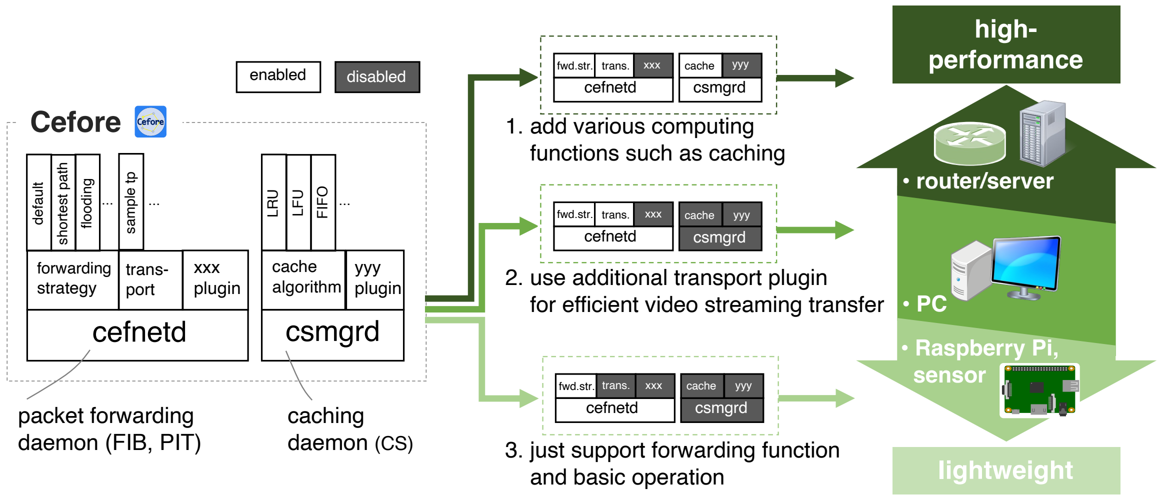

Fig. 2. Cefore's pluggable architecture.

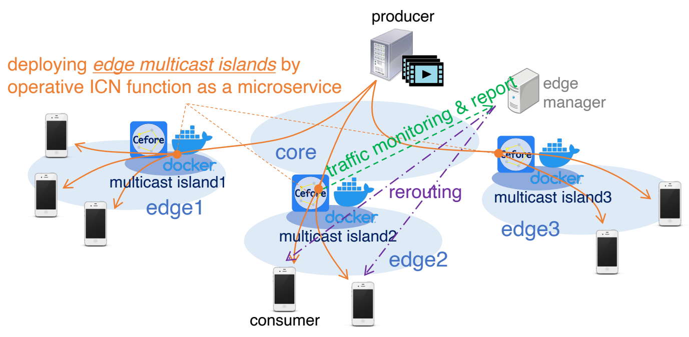

Fig. 3. A deployable ICN framework with multicast islands in the edges.

over the Internet, which facilitates the multicast capability via the traffic aggregation of streaming users to reduce bandwidth demands between the core network and the server. To this end, we deploy ICN functions at the edge and construct *multicast islands* for multiple consumers in the edge.

To achieve this architecture, we require two components: an *operation function* and a *deployment platform*. Using the operation function, we simply monitor the IP traffic of the edge site, predict its demand, and deploy the ICN function at the appropriate time. Overall, we provide a network monitoring function to report the deployment timing to an edge manager such as k8s. After reporting the deployment timing, Cefore on a Docker container is started as a multicast island. The edge manager changes the routing table of each consumer in the targeted edge to reroute traffic to the multicast island for using the ICN's multicast function as with the multicast island2 example shown in Fig. 3. To interface the ICN functions with existing cloud-native paradigms, we need a new management tool for managing ICN control plane and data plane such as [22]. Note that we have reserved the detailed design and implementation of this management tool for our future work.

## *B. Deploying ICN/Cefore as a Docker Microservice*

To deploy the ICN function in an edge-computing infrastructure, we require a new deployment platform. In this section, we describe the significant advantages of the Docker microservice technology when applying it to construct ICN networks over IP. The main reasons for using Docker are as follows:

- Lightweight

  Compared with virtual machines, a Docker container is exceptionally lightweight. Thus, we can build numerous containers on one physical machine, which enriches the evaluation scenarios of ICN networks and improves experimental scalability.
- Performance

  Since Docker containers do not contain operating systems, they can be easily and quickly initiated and terminated, which facilitates comfortable testing and evaluation.
- Scalability

  The setup requires multiple ICN nodes to provide different functions in the network at the same time. Notably, the microservices concept meets this requirement. Useful option tools (e.g., `docker-compose`[2]) can be used for flexibly and quickly setting up containers and providing services.

To build our Cefore/Docker images, the "Dockerfile" is prepared for the `base` function, which outlines the necessary functions for ICN services at the container node.

```
----------------------------------------------------
# base/Dockerfile
FROM ubuntu:20.04
LABEL maintainer="hayamizu <hayamizu[at]nict.go.jp>"
RUN mkdir -p /cefore
WORKDIR /cefore
RUN apt update
RUN apt install -y git build-essential libssl-dev automake
RUN apt -y clean
```

[2]https://docs.docker.com/compose/

```
RUN git clone https://github.com/cefore/cefore.git
WORKDIR /cefore/runner_test
--------------------------------------------------
```

In the `base` function, we specify the base image of the operating system, i.e., Ubuntu 20.04, install the Cefore's dependencies, and download published Cefore source codes from the Internet, e.g., GitHub.com.

We can also prepare this and other Dockerfiles to enhance the ICN nodes' service functions. For example, if we were to construct a simple ICN network with consumer, router, and producer nodes, two types of microservice Dockerfiles can be used: one for the end-application, i.e., `min`, and another for the router's caching function, i.e., `cache`. End applications do not require special functions; thus, the file can be written as follows:

```
--------------------------------------------------
# min/Dockerfile
FROM cefore/base
WORKDIR /cefore/cefore
RUN ./configure
RUN make; make install; make clean
RUN ldconfig
ENV USER root
COPY ./entrypoint.bash /cefore
ENTRYPOINT /cefore/entrypoint.bash
--------------------------------------------------
```

This file builds only Cefore binaries for preparing the deployment initiation.

To add the caching function, i.e., `cache`, we modify the `min` file such that the ICN node can emulate a router with CS. The configuration options, i.e., `--enable-cache --enable-csmgr`, and configuration parameters, i.e., cefnetd/csmgrd caching functions, are added to the `min` service.

```
--------------------------------------------------
# cache/Dockerfile
FROM cefore/base
WORKDIR /cefore/cefore
RUN ./configure --enable-cache --enable-csmgr
RUN make; make install; make clean
RUN ldconfig
RUN echo "CS_MODE=2" > /usr/local/cefore/cefnetd.conf
RUN echo "CACHE_TYPE=memory" > /usr/local/cefore/csmgrd.conf
ENV USER root
COPY ./entrypoint.bash /cefore
ENTRYPOINT /cefore/entrypoint.bash
--------------------------------------------------
```

As presented in the aforementioned examples, these ICN functionalities can be easily and flexibly deployed as microservices according to the network and application requirements.

## IV. Experiment

### A. Implementation

We implemented a prototype of IC-CNF running on a Docker container. We deployed 20 Docker containers including consumer nodes, edge/core routers, and producer, on a physical host machine running Ubuntu 22.04. The host machine is with 2.80 GHz Intel Xeon Platinum 8362 CPU $\times$ 128, and 1TB of memory.

We used Docker version 24.0.5 as a base library and leveraged `docker-compose` tools for easily and quickly deploying ICN functions as multicast islands. by using IC-CNF framework. As explained previously, we built a Cefore on Docker container image in advance before deploying as multicast islands, and as necessary, i.e., when traffic load has exceeded the pre-configured threshold, we start the Cefore/Docker container on the host machine.

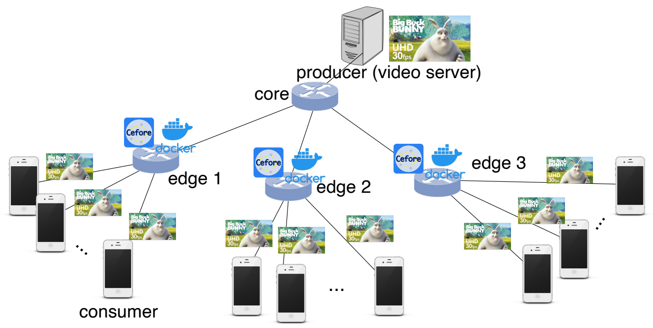

Fig. 4. Evaluation model.

### B. Model

To investigate the effects of our proposed deployable ICN framework, we conducted an experimental evaluation using a 4K live video streaming scenario with 10 minutes duration[3]. Figure 4 illustrates the evaluation model used for this experiment. The producer at the content server sent a 4K video encoded with approximately 20 Mbps quality to the network. There were 15 consumers, five at each edge node, and they joined the live streaming event. We randomly set the join delay between consumers to $n$ and $n+1$ ($n = 1 \cdots 14$) based on an exponential distribution with an arrival rate of $\lambda = 0.1$.

We evaluated the traffic based on a unicast-only IP network and Information-Centric Cloud Native Network Function (IC-CNF) network configuration. In the unicast-only IP network, all consumers must download video data from the origin server. With IC-CNF, we deployed an ICN function based on the amount of traffic passing through each edge node and accordingly constructed multicast islands at the appropriate timing. The evaluation metrics included traffic load in terms of throughput at the server, download rate of each consumer, and total traffic transferred by the core router.

### C. Result

Figure 5 shows the time variations of the traffic load in the producer (server) node. With IP (unicast), the traffic load monotonically increased with the addition of users. After all users joined the live streaming, the traffic load saturated at approximately 300 Mbps, which approximated the demands of 15 streams (users) based on 20 Mbps allocated for each. With IC-CNF, we found that our proposed deployment strategy works well with the join timing of streaming users; thus, it successfully reduced server traffic loads. In the current configuration, we aggregated the traffic of five users around 100 Mbps.

[3]https://peach.blender.org/

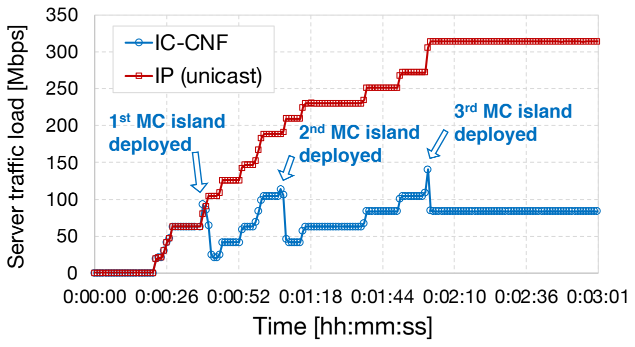

Fig. 5. Time variation of server traffic load.

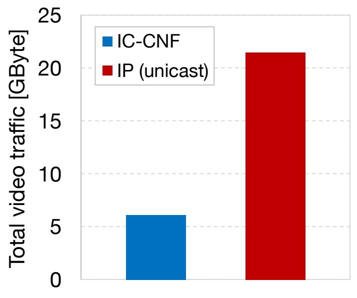

Fig. 6. Total traffic transferred at the core router.

As the total traffic approached this threshold, our proposed monitoring tool successfully detected and reported the traffic concentrations. The edge manager then rerouted the traffic requests to the newly created multicast island, where they were aggregated, resulting in significantly decreased server loads. Ultimately, 73.3% of the total server traffic load was reduced at the steady state: IP (unicast) = 314.0 Mbps and ICN = 83.7 Mbps. We also confirmed that the download rate of all consumers was sufficiently high at  20 Mbps, which meets 4K streaming quality.

Figure 6 shows the amount of video traffic transferred by the core router. Our proposed deployable ICN framework reduced 71.6% of its data traffic because the edge multicast performed adequately. This significant reduction in core traffic brought various benefits, including congestion mitigation, latency improvements by server load reduction, and energy-savings for data commmunicatoins.

## V. Related Work

### A. Multicast

There are several related work to address IP multicast deployment issues. ODMT [23] was a proposal for dynamic multicast tunneling that bridges native and non-native multicast domains. Although ODMT provides an on-demand inter-provider multicast tunneling dynamically, it requires complex equipments, Controller Node (CoN), Forwarding Node (FoN) and Controller node Resolver (CR). This requirement leads to an additional deployment barrier.

One of the major approaches for multicast tunneling is the Automatic Multicast Tunneling (AMT) [24] specified in the IETF PIM working group. AMT is a protocol that enables users who do not have native multicast connectivity to join multicast sessions and has been implemented by multiple vendors such as [25]. AMT replicates unicast packets for users and transmits them via AMT relay and gateway nodes, and hence the bandwidth cost for the packet replication will be higher than expected.

Moreover, although Crowcroft [26] observed the difficulties in both ICN and IP multicast deployment, these deployment barriers are different. Because of the different IP address semantics, IP multicast requires not only IP multicast routing paths are different from the unicast, but also IP multicast applications must be developed with the multicast specific socket options to invoke IGMP/MLD join and leave operations. Usually, application developers do not consider the routing protocols and paths on which their applications run; therefore there is a big obstacle for application development as well. In contrast, ICN does not impose to distinguish the protocol for application programming so that it is not necessary for application programmers to decide communication styles either unicast or multicast for their applications.

### B. Cloud-native network function

Cloud-native network function (CNF) is a new networking softwarization tool utilized for network management/orchestration optimization. CNF covers a lot of underlying technologies on network virtualization. D. Breitgand et. al. [27] proposed cloud native MANO architecture that exploits benefits of Kubernetes (k8s) [18], which is a de facto industry standard for CNF orchestration. Halpern et al., [29] proposed cloud-native traffic steering architecture without relying on SDN technologies. Service Function Chaining (SFC) [28] has been originally proposed as an SDN-based architecture. The authors of [29] integrated CNF with the SFC concept and newly proposed a cloud-native service function chaining technology (CN-SFC) which does not rely on the SDN controller for ensuring effective traffic steering among cloud-native network functions. K8s is used for the orchestration platform of service functions deployed as CNF. K. Kanai et al., [30] proposed information-centric service mesh framework for improving in-network computing performance by using k8s platform. This framework provides parallel data processing for dynamically provisioning in-networking computing resources. These studies are beneficial and can be good compasses for us to manage/control states of deployed multicast islands as CNFs because we are planning to combine our developed IC-CNF with k8s platform as aforementioned.

## VI. Conclusion

In this study, to deploy multicast communication services in the Internet, we designed an multicast island in which multicast services are purely enabled, and we proposed a novel IC-CNF that quickly and easily deploys multicast functionalities using ICN-based edge-computing paradigms. IC-CNF is enabled using the Cefore open-source software platform, which enables ICN communications using CCNx-1.0 messages defined by the IRTF. Cefore was integrated with emerging microservices technologies (e.g., Docker) for quick and simple ICN deployments. In our experiment, we evaluated the traffic reduction performance of the proposed scheme in a live 4K video streaming scenario and demonstrated that our proposed deployable ICN framework significantly reduces server and core router traffic loads while ensuring sufficient throughput for each consumer. In future work, we plan to embed this IC-CNF using representative orchestration technologies (e.g., Kubernetes) while enhancing its network management and operation schemes.

## Acknowledgment

This work was supported in part by the JSPS KAKENHI Grant Numbers 22K17888 and 20H04187.

## References

[1] B. Quinn, et al., "IP Multicast Applications: Challenges and Solutions," IETF RFC3170, September 2001.

[2] C. Diot, B. N. Levine, B. Lyles, H. Kassem and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, no. 1, pp. 78-88, Jan.-Feb. 2000.

[3] H. Asaeda, et al., "Gap Analysis in IP Multicast Dissemination," *Springer LNCS 4866 (AINTEC2007)*, pp. 199-212, November 2007.

[4] D. Farinacci, et al., "Multicast Lessons Learned from Decades of Deployment Experience," IETF Internet-Draft, draft-ietf-pim-multicast-lessons-learned (work in progress), July 2023.

[5] A. El-Sayed, V. Roca, and L. mathy, "A Survey of Protocols for an Alternative Group Communication Service," *IEEE Network*, vol. 17, no.1, pp.46-51, January 2003.

[6] M. Hosseini, D. T. Ahmed, S. Shirmohammadi and N. D. Georganas, "A survey of application-layer multicast protocols," in IEEE Communications Surveys & Tutorials, vol. 9, no. 3, pp. 58-74, Third Quarter 2007.

[7] S. Alekseev, et al., "A New Algorithm for Construction of a P2P Multicast Hybrid Overlay Tree based on Topological Distances," *Computer Science*, January 2016.

[8] V. Jacobson, et al., "Networking Named Content," in *Proc. ACM CoNEXT'09*, vol. E102-B, no. 9, September 2019.

[9] M. Mosko, et al., "Content-Centric Networking (CCNx) Semantics," IRTF RFC8569, July 2019.

[10] M. Mosko, et al., "Content-Centric Networking (CCNx) Messages in TLV Format," IRTF RFC8609, July 2019.

[11] L. Zhang, et al., "Named Data Networking," *ACM SIGCOMM CCR*, vol. 44, no. 3, pp 66-73, July 2014.

[12] G. Xylomenos et al., "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024-1049, Second Quarter 2014.

[13] A. Rahman, et al., "Deployment Considerations for Information-Centric Networking (ICN)," IRTF RFC8763, April 2020

[14] Shailendra, S.,et al., "A novel overlay architecture for Information Centric Networking," 2015 21st National Conference on Communications, NCC 2015, April 2016.

[15] P. Suthar, et al., "Experimental Scenarios of Information-Centric Networking (ICN) Integration in 4G Mobile Networks," IRTF RFC9269, August 2022.

[16] S. Fayazbakhsh et al., "Less pain, most of the gain: Incrementally deployable ICN," in *Proc. ACM SIGCOMM'13*, October 2013, pp. 147–158.

[17] "Docker," https://www.docker.com/ , (accessed on 25 September 2023).

[18] "Kubernetes," https://kubernetes.io/ , (accessed on 25 September 2023).

[19] "Cefore," https://cefore.net , (accessed on 25 September 2023).

[20] H. Asaeda, et al., "A Survey of Information-Centric Networking: The Quest for Innovation," *IEICE Transactions on Communications,* Online ISSN 1745-1345, Print ISSN 0916-8516, August 2023.

[21] H. Asaeda, et al., "CCNinfo: Discovering Content and Network Information in Content-Centric Networks," IRTF RFC9344, February 2023.

[22] Y. Hayamizu, et al., "Controller-Assisted Adaptive Video Streaming Experimented in Cloud-Native ICN Platform," in *Proc. IEEE INFOCOM 2023 Workshops (ICCN)*, Hoboken, NJ, USA, May 2023, pp. 1-6.

[23] A. Basuki, et al., "ODMT: On-demand Inter-domain Multicast Tunneling," *Journal of Information Processing*, vol.20, no.2, pp.347–357, February 2012.

[24] G. Bumgardner, "Automatic Multicast Tunneling," IETF RFC7450, February 2015.

[25] "Automatic Multicast Tunneling," Cisco Systems, Inc., https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipmulti_pim/configuration/xe-16-10/imc-pim-xe-16-10-book/imc-auto-mlt-tun.pdf (accessed on 30 September 2023).

[26] J. Crowcroft, "Statement: Hard lessons for ICN from IP multicast," in *Proc. ACM ICN 2022*, Osaka, Japan, September 2022.

[27] D. Breitgand, et al., "Toward True Cloud Native NFV MANO," in *Proc. Network of the Future (NoF 2021)*, Coimbra, Portugal, November 2021.

[28] J. Halpern, et al., "Service Function Chaining (SFC) Architecture," IETF RFC7665, October 2015.

[29] B. Dab, et al., "An Efficient Traffic Steering for Cloud-Native Service Function Chaining," in *Proc. Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Paris, France, 2020.

[30] K. Kanai, T. Tsuda, H. Nakazato, and J. Katto, "Information-Centric Service Mesh for Autonomous In-Network Computing," in *Proc. ACM Conference on Information-Centric Networking (ICN'22), pp. 159–161.*