National Institute of
Information and Communications Technology

# Cefore: CCNx-based Cloud-native Network Function for Deploying ICN as a Service

**Yusaku Hayamizu**, Atsushi Ooka, Kazuhisa Matsuzono, Hitoshi Asaeda

National Institute of Information and Communications Technology (NICT), Japan

IETF115

5-11 November, London, UK

# Outline

- Background/Motivation
- Cefore: CCNx-based Extensible Packet Forwarding Engine
- Cefpyco: Cefore Python Compact Package
- Cefore x Docker Integration
- Sample Scenarios
- Conclusion

# Background / Motivation

- ICN [1]
  - changing NW from "*host-centric*" to "*content-centric*"
- CCNx [2,3] / NDN [4]
  - Content-Centric Networking (IETF/IRTF RFC)
  - Named-Data Networking (NDN project)
- Cefore [5,6]
  - open-source software enabling ICN communications
  - CCNx1.0-compliant packet forwarding engine developed/maintained by NICT

one missing piece might be...

a **deployment solution** of developed ICN modules into the Internet infrastructures

# Aim of presentation

1. Introduction of Cefore
   - the Cefore software platform for enabling CCNx-based communications
   - application development with Cefpyco
     - a Python wrapper program that helps developing CCNx applications

2. Cefore/Docker integration
   - Cefore's integration with the emerging Docker technologies for rapid and easy deployment of ICN

introduce a method for easily deploying **ICN as a (micro)Service** and quickly constructing ICN-based networks

# Outline

- Background/Motivation
- Cefore: CCNx-based Extensible Packet Forwarding Engine
- Cefpyco: Cefore Python Compact Package
- Cefore x Docker Integration
- Sample Scenarios
- Conclusion

# Cefore: CCNx-based Extensible Packet Forwarding Engine

- CCNx1.0
  - defined in the RFCs 8569 and 8609, specified by IRTF ICNRG

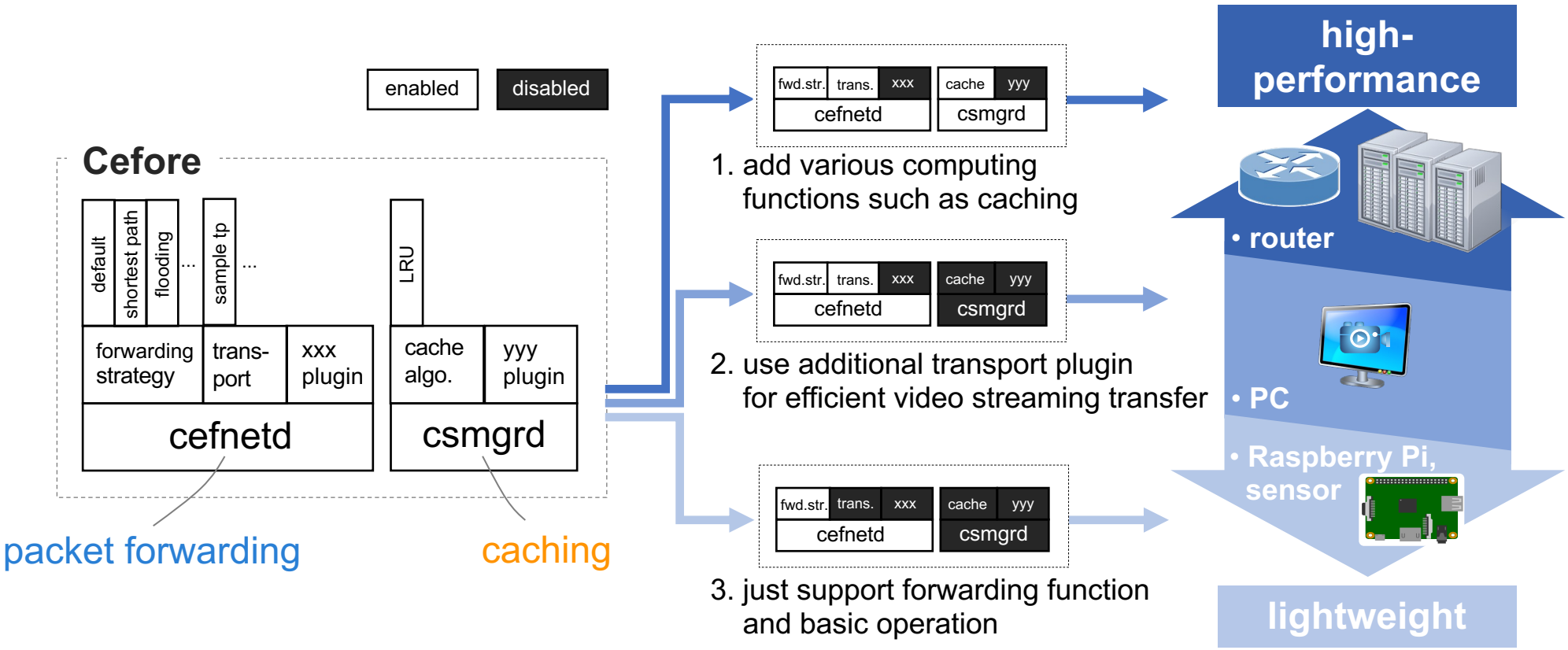
- Cefore
  - originally designed in 2016
  - CCNx1.0 packet (Interest/ContentObject) forwarding/caching engine
  - open-source, and published in the web* and github+

* https://cefore.net/
+ https://github.com/cefore

# Design policies

- *Lightweight*
  - the platform should be usable for 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 should be easily conducted and tested using real network equipment
- *Extensibility*
  - the platform should be easily extensible to accommodate novel functions to satisfy future network needs

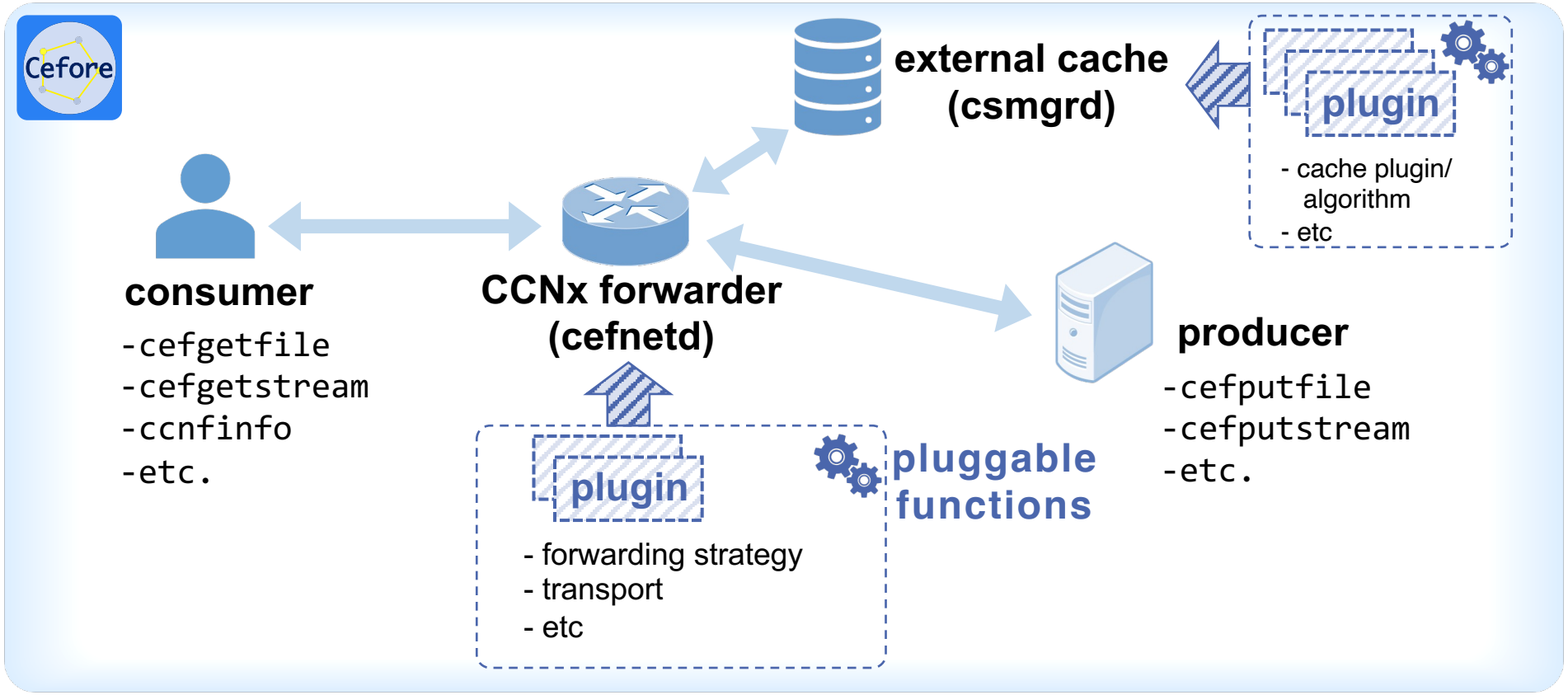
# Pluggable architecture of Cefore




- Researchers can install necessary ICN functions depending on their requirements while considering their machine resource constraints

# The Cefore software package

- Cefore provides **all-in-one package** for CCNx-based communications
  - core daemons (cefnetd/csmgrd with extensible plugins)
  - consumer utility (cefgetfile/cefgetstream) & producer utility (cefputfile/cefputstream)
  - network management tools (CCNinfo)



# Cefpyco

- Cefpyco (Cefore Python Compact package)*
  - a Python-based wrapper program that help developing CCNx applications running with Cefore
  - enables easy coding for python programmers (compared to the original C language)
  - Example: sending an Interest packet

**C language**

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <cefore/cef_define.h>
#include <cefore/cef_client.h>
#include <cefore/cef_frame.h>
#include <cefore/cef_log.h>

int main(int argc, char *argv[]) {
    CefT_Client_Handle fhdl;
    CefT_Interest_TLVs params_i;
    int res;
    cef_log_init ("cefpyco");
    cef_frame_init();
    res = cef_client_init(port_num, conf_path);
    if (res < 0) return -1;
    fhdl = cef_client_connect();
    if (fhdl < 1) return -1;
        memset(&params_i, 0, sizeof(CefT_Interest_TLVs));
    res = cef_frame_conversion_uri_to_name("ccnx:/test",
params_i.name);
    if (res < 0) return -1; // Failed to convert URI to name.;
    params_i.name_len = res;
    params_i.hoplimit = 32;
    params_i.opt.lifetime_f = 1;
    params_i.opt.lifetime = 4000ull; /* 4 seconds */
    params_i.opt.symbolic_f = CefC_T_OPT_REGULAR;
    params_i.chunk_num_f = 1;
    params_i.chunk_num   = 0;
    cef_client_interest_input(fhdl, &params_i);
    if (fhdl > 0) cef_client_close(fhdl);
    return 0;
}
```

**33 lines -> 4 lines**

**Python**

```python
import cefpyco

with cefpyco.create_handle() as h:
    h.send_interest("ccnx:/test", 0)
```

Cefpyco is a user-friendly implementation as developers can call CCNx functions such as sending Interest/Data

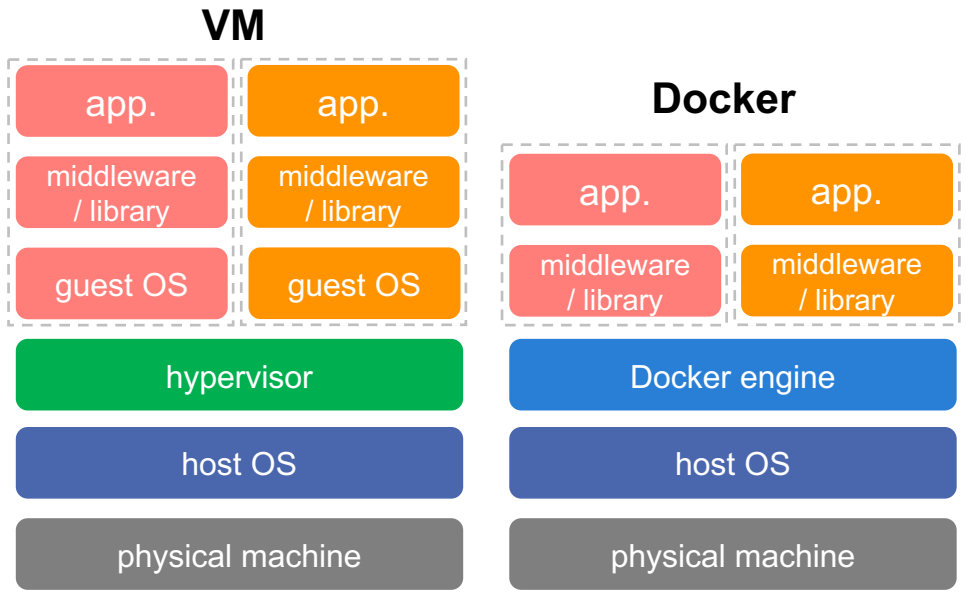
*https://github.com/cefore/cefpyco

# Outline

- Background/Motivation
- Cefore: CCNx-based Extensible Packet Forwarding Engine
- Cefpyco: Cefore Python Compact Package
- Cefore x Docker Integration
- Sample Scenarios
- Conclusion

# What is Docker?

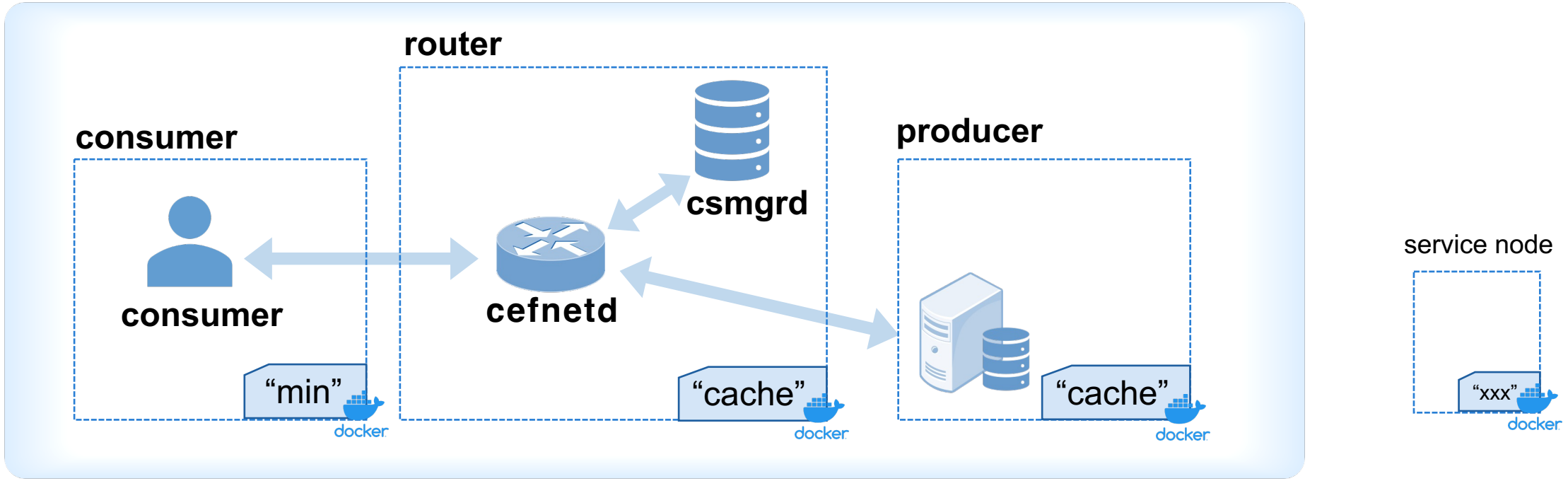
- Docker
  - a platform of container-based virtualization technology for quick and scalable deployment of network services
- Benefits
  - Lightweight
    - a Docker container is very lightweight compared with VM
    - -> we can build many containers in one physical machine
    - -> this enriches evaluation scenario of ICN networks and improves scalability of experiments
  - Performance
    - Docker containers do not contain OS
    - -> they can be easily and quickly initiated and terminated
    - -> this facilitates comfortable test and evaluation of ICN services
  - Scalability
    - there is a requirement that multiple ICN nodes providing different functions co-exist in a network
    - the concept of microservices that each service image is built for each purpose fits this requirement
    - useful option tools such as `docker-compose` can be used for flexibly and quickly setting up Docker containers




comparison of VM and Docker

# Example scenario of Cefore/Docker-based networking

- Scenario
  - The consumer requests a file
  - The producer responses to the request and send back data
  - The CCNx router stores received data into CS (csmgrd)



# Example 1 – writing a Dockerfile

- define a microservice as a ``base'' service
  - base function as an ICN node
  - necessary functions for providing ICN services as a container node

base/Dockerfile

```
FROM ubuntu:20.04
LABEL maintainer="hayamizu <hayamizu@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
RUN git clone https://github.com/cefore/cefore.git
WORKDIR /cefore/runner_test
```

install basic libraries for building Cefore

download the Cefore software from the github

Afterward, other enhanced ICN services, e.g. ``min'' and ``cache,'' inherit this ``base'' image

# Example 2 – writing a Dockerfile

- define a microservice as a ``min'' service
  - minimum functions serving as a ICN node, i.e., installation & app. preparation

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
```

configure & make & install Cefore

set the entrypoint, i.e., just starting Cefore daemon (cefnetd)

define a service “min” that provide minimum ICN functions (application tools)

# Example 3 – writing a Dockerfile

- define a microservice as a ``cache'' 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
```

configure Cefore by enabling ``csmgr/cache'' option

make & install Cefore

modify the configuration files.
CS_MODE=2 (csmgrd)
CACHE_TYPE=memory

set the entrypoint, i.e., starting Cefore daemons (cefnetd & csmgrd)

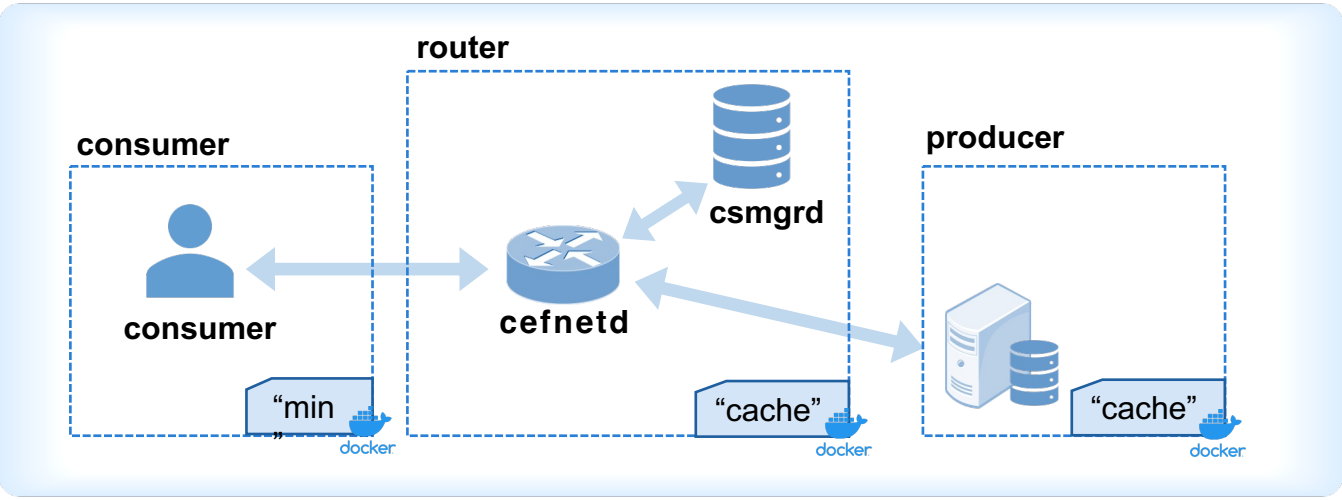
define “cache” service by adding caching function (cache/csmgrd) to the base ICN functions

# Leveraging docker-compose*

- `docker-compose`
  - a tool for defining and running multi-container Docker applications
  - easy service configuration using a YAML file
  - can create and start all the services from the configuration with a single command

  -> easy to conduct scenario-based experiments(emulations)

  like network simulations such as ns-3




*https://docs.docker.com/compose/

Example: docker-compose.yml

```
version: "3.3"
services:
    producer:
        image: cefore/cache
        container_name: "producer"
        hostname: "producer"
        working_dir: "/cefore"
        networks:
            downward:
              ipv4_address: 10.0.1.10
    router:
        image: cefore/cache
        container_name: "router"
        hostname: "router"
        working_dir: "/cefore"
        networks:
            downward:
              ipv4_address: 10.0.1.20
    consumer:
        image: cefore/min
        container_name: "consumer"
        hostname: "consumer"
        working_dir: "/cefore"
        networks:
            downward:
              ipv4_address: 10.0.1.100
networks:
    downward:
      name: downward
      driver: bridge
      ipam:
        driver: default
        config:
          - subnet: 10.0.1.0/24
```

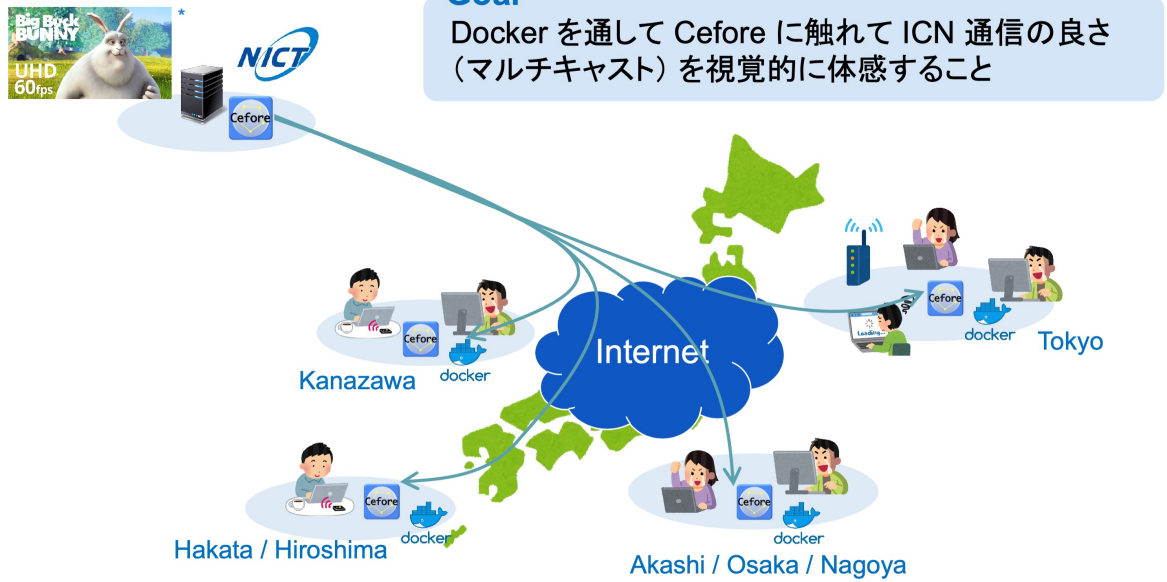
# Outline

- Background/Motivation
- Cefore: CCNx-based Extensible Packet Forwarding Engine
- Cefpyco: Cefore Python Compact Package
- Cefore x Docker Integration
- Sample Scenarios
- Conclusion

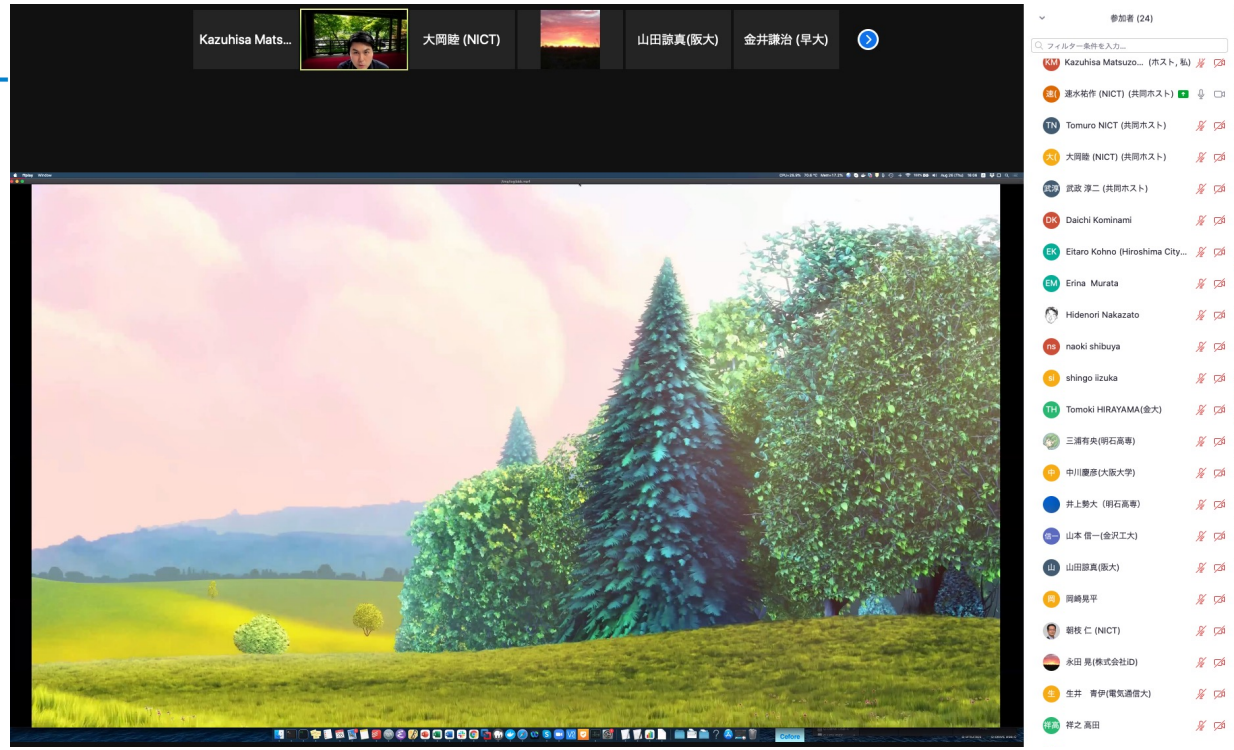
# Sample scenario – Video streaming over the Internet

- IEICE ICN summer workshop 2021 [fully-online]
  - Cefore/Docker hands-on
  - Multicast video streaming using Cefore/Docker platforms*
    - The producer is located at NICT (Tokyo)
    - The consumers receive the video streaming from their homes/schools/companies

## PracticeA: Cefore/Docker を用いたマルチキャストストリーミング




* https://peach.blender.org/




*You can get sample codes from https://github.com/cefore/2021-hands-on [materials are in Japanese only]

# Conclusion

- Cefore
  - CCNx-based extensible packet forwarding engine
  - All-in-one package for CCNx-based communications
- Cefpyco
  - Useful development tool for creating new ICN applications
- Cefore/Docker integration
  - Quick and scalable deployment of CCNx functions
- Sample scenario
  - Video streaming
- Future work
  - A possibility of collaboration with the emerging Docker orchestration technologies such as Kubernetes

Cefore/Docker-based networking can be one possible option for easily constructing ICN networks over the existing Internet infrastracture

# Reference


[1] V. Jacobson, et al., “Networking Named Content,” in Proc. ACM CoNEXT’09, vol. E102-B, no. 9, Sept. 2019.

[2] “Content-Centric Networking (CCNx) Semantics,” https://datatracker. ietf.org/doc/rfc8569/ , Accessed on 7 July 2022.

[3] “Content-CentricNetworking(CCNx)MessagesinTLVFormat,”https: //datatracker.ietf.org/doc/rfc8609/ , Accessed on 7 July 2022.

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

[5] “Cefore,” https://cefore.net , Accessed on 7 July 2022.

[6] H.Asaeda,etal.,“Cefore: Software Platform Enabling Content-Centric Networking and Beyond,” IEICE Trans. Commun., vol.E102-B, no.9, pp.1792-1803, Sept. 2019.

National Institute of
Information and Communications Technology

# Thank you.

Cefore

https://cefore.net/

GitHub

https://github.com/cefore