

Cefore チュートリアル ～ ICN/Cefore 解説と Cefore プログラミング～

速水祐作

国立研究開発法人情報通信研究機構(NICT)
2025年9月2-3日

2025年電子情報通信学会情報指向ネットワーク技術(ICN)研究会ワークショップ@仙台商工会議所



目次

- 事前準備
 - Ubuntu VM の準備
 - Wi-Fi AP 接続
 - [参考] ラズパイのセットアップ
- ICN/Cefore の概要と基本的な通信
 - 情報指向ネットワーク技術(ICN)
 - Cefore(セフォール)
 - Cefore による ICN 通信
- Cefore/Cefpyco を用いたPythonアプリ開発
 - Cefpyco: Cefore 用 アプリケーション開発用 Python ライブライ
 - Cefpyco を用いた基本通信
- ラズパイを用いたライブ配信アプリ
 - 動画配信技術の基礎
 - Cefore を用いたライブ配信
 - Cefore/Cefpyco を用いたライブ配信

事前準備: Ubuntu (VM) の設定



動作環境 - Ubuntu over Windows/macOS

Host OS のスペック

- CPU
 - 最低: 6 cores
 - 推奨: 8 cores
- メモリ
 - 最低: 8 GB
 - 推奨: 16 GB
- HDD
 - 最低: 200 GB
 - 推奨: 400 GB

仮想マシンの設定

- CPU
 - 最低: 4 cores
 - 推奨: 6 cores
- メモリ
 - 最低: 6 GB
 - 推奨: 8 GB
- HDD
 - 最低: 32GB
 - 推奨: 64GB



[参考] 仮想マシン(VM) エンジンのインストール

- Virtual Box
 - Windows
 - <https://qiita.com/HirMtsd/items/d43fc5215a88cbf414c9>
 - macOS (intel CPU)
 - <https://note.com/mio301/n/n419555b8e07c>
- UTM
 - macOS (Apple Silicon M1/M2/M3/M4 chip)
 - <https://envader.plus/article/66>
 - ※ 最初から Ubuntu Desktop をインストールできないので、Ubuntu server image をダウロードしてから、GUI化のため sudo apt install ubuntu-desktop する

NOTE

- Ubuntu 22.04 の iso イメージのダウンロードに時間がかかりますので、事前にダウンロードしてインストールしておいて頂けますと幸いです。
- サーバへのリモート接続ではなく、手元の laptop PC に VM 環境をご用意ください



Ubuntu 設定 -- Ubuntu 22.04

- 必要なライブラリのダウンロード

```
cefore:~$ sudo apt update -y
cefore:~$ sudo apt upgrade -y
cefore:~$ sudo apt install git gcc make ffmpeg net-tools emacs vim nano
cefore:~$ sudo apt update
```

- TCP/UDP ソケットバッファサイズの調整
 - config ファイル "88-cefore.conf" の作成
 - /etc/sysctl.d/ に配置することで再起動後も自動反映

```
cefore:~$ cat 88-cefore.conf
net.core.wmem_default=20971520
net.core.wmem_max=41943040
net.core.rmem_default=20971520
net.core.rmem_max=41943040
net.ipv4.udp_mem=10481520 20971520 41943040
net.ipv4.udp_wmem_min=10481520
net.ipv4.udp_rmem_min=10481520
net.ipv4.tcp_mem=10481520 20971520 41943040
net.ipv4.tcp_wmem=10481520 20971520 41943040
net.ipv4.tcp_rmem=10481520 20971520 41943040
cefore:~$ sudo sysctl -f /etc/sysctl.d/88-cefore.conf
```

NOTE: Experientially, we would recommend to increase the socket buffer size of kernel parameters in advance, when you conduct an experiment with high-speed data rate.



Ubuntu 設定 -- Cefore ダウンロード [推奨]

Readme.md

README

Code

https://github.com/cefore/cefore.git

もしくは、以下のコマンドで github から
Cefore をダウンロード

```
cefore:~$ git clone https://github.com/cefore/cefore.git
```

<https://github.com/cefore/cefore>



Ubuntu 設定 -- Cefore インストール

- Downloading source codes
 - <https://cefore.net/>
 - <https://github.com/cefore/cefore>
- Installing dependencies

```
$ sudo apt-get install libssl-dev automake
```

- Installing Cefore

```
$ cd cefore-0.10.0g
$ autoconf          # 場合によっては、autoreconf が必要
$ automake
$ ./configure --enable-csmgr --enable-cache
$ make
$ sudo make install
$ sudo ldconfig      # binaries are to be installed in the /usr/local/bin, sbin
```

Please see more details Section 2 “Installation” of README.

*<https://cefore.net/doc/Readme.html>



Ubuntu 設定 -- Cefore ハンズオン資料のダウンロード

The screenshot shows a GitHub repository page for '2024-hands-on'. The repository has one branch ('main') and one tag ('Initial commit'). The README file contains the following text:

2024-hands-on
Cefore hands-on materials used for the ICN workshop held in the 2024 IEICE Society Conference.

On the right side of the page, there is a red box around the 'Code' button in the top navigation bar, with a red arrow pointing to it labeled 'ここをクリック' (Click here). Below the 'Code' button, there is another red box around the copy icon in the 'Clone' section, with a red arrow pointing to it labeled 'URLをコピー' (Copy URL).

A large white box on the right contains the following text:

以下のコマンドで github から
ハンズオン資料(スクリプト含む)をダウンロード

```
cefore:~$ git clone https://github.com/cefore/2024-hands-on.git
```

At the bottom of the page, there is a 'Contributors' section with two entries:

- yusaku-hayamizu Yusaku Hayamizu
- cefore cefore

※ VM (Ubuntu) の人は、必要ファイルのダウンロード終了後、WiFiブリッジ設定に変更



Ubuntu 設定-- Wi-Fi AP (raspi) への接続

- Wi-Fi AP の情報

```
SSID: pi1  
Pass: icn-ken2025  
IP: 10.0.0.1/24 (DHCP)
```

- 動作確認

- Ubuntu で ifconfig を使って wifi AP のアドレスが設定できているか
- raspi – Ubuntu 間のIP接続確認
 - ping pi1.local
 - ping 10.0.0.1

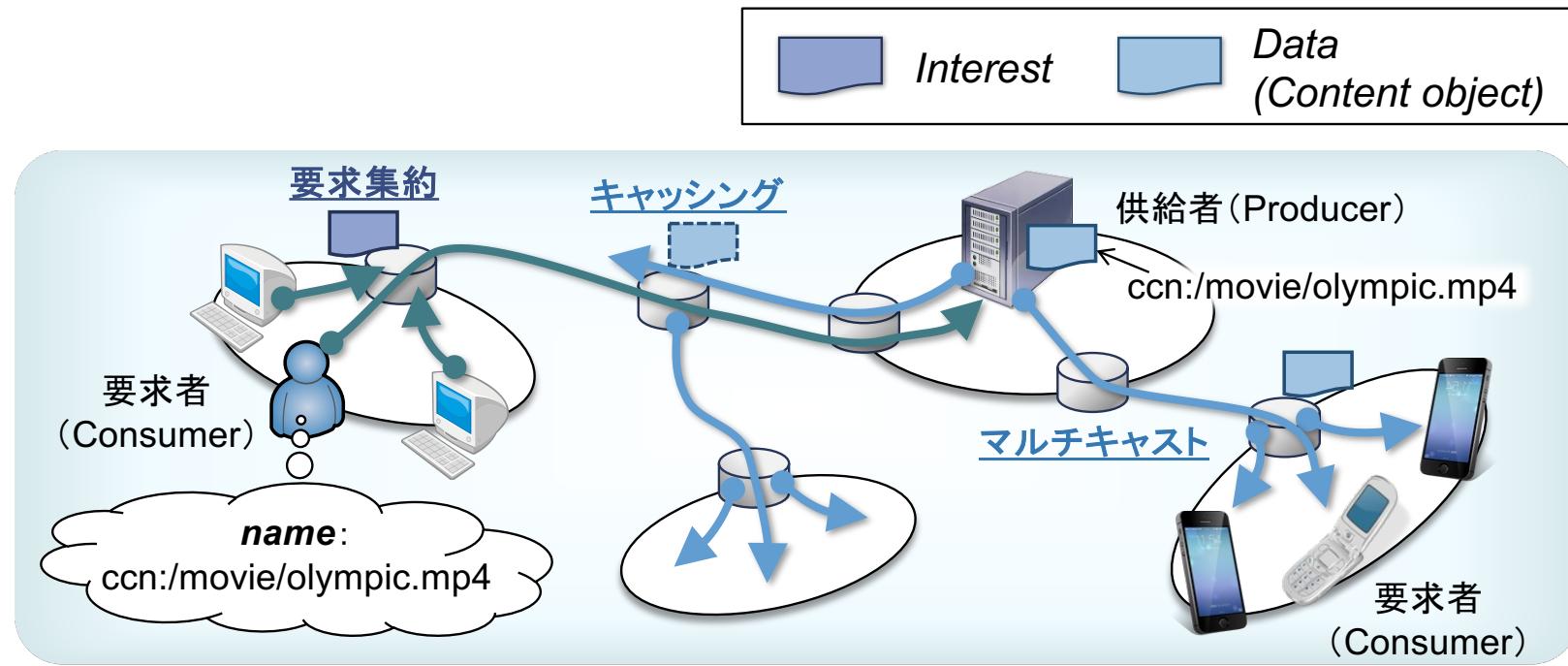
```
hayamizu@cefore:~$ ifconfig  
enp0s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
        inet 192.168.102.4 netmask 255.255.255.0 broadcast 192.168.102.255  
          inet6 fe80::5870:29ff:fe6b:3244 prefixlen 64 scopeid 0x20<link>  
            ether 5a:70:29:6b:32:44 txqueuelen 1000 (Ethernet)  
              RX packets 220 bytes 24079 (24.0 KB)  
              RX errors 0 dropped 0 overruns 0 frame 0  
              TX packets 228 bytes 27643 (27.6 KB)  
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

ICN / Cefore の概要

情報指向ネットワーク (ICN) の概要

情報指向ネットワーク技術 (Information-Centric Networking: ICN)

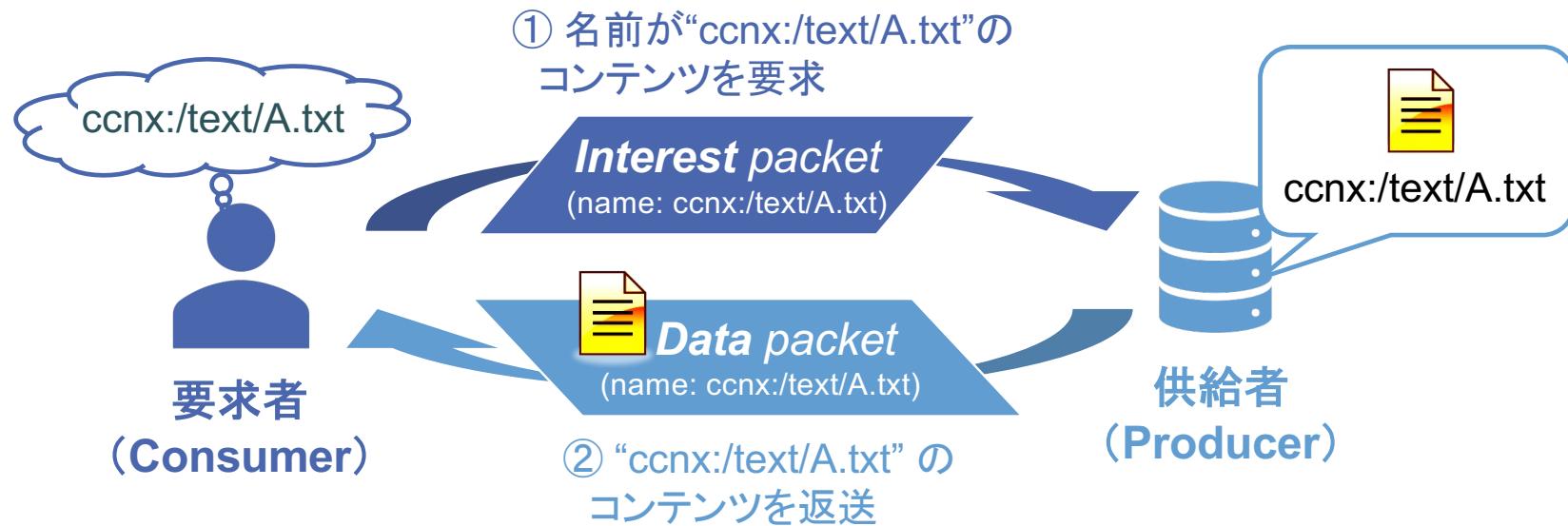
- ホスト中心ではなくコンテンツ中心のネットワークアーキテクチャ
 - IP アドレスではなくコンテンツ名を使用
- コンテンツを効率的に配布・取得するための仕組みをサポート



ICN の概略図

コンテンツ名に基づく通信

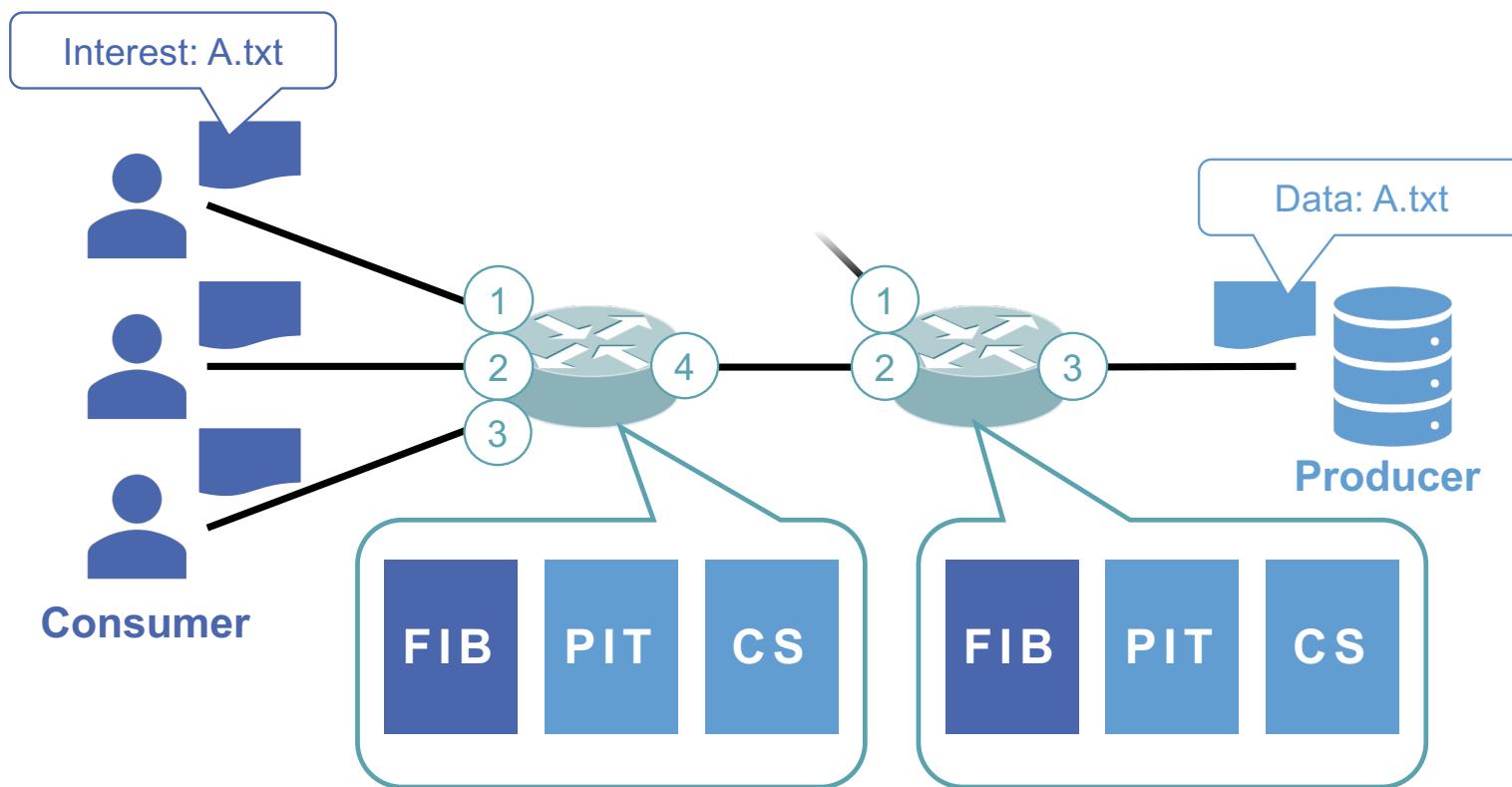
- 消費者(Consumer)と供給者(Producer)で通信
 - IP アドレスのような場所(通信相手)の情報は不要
- 2種類のパケットを用いて通信
 - Interest: コンテンツを要求するためのパケット
 - Data/Content Object: コンテンツを返送するためのパケット





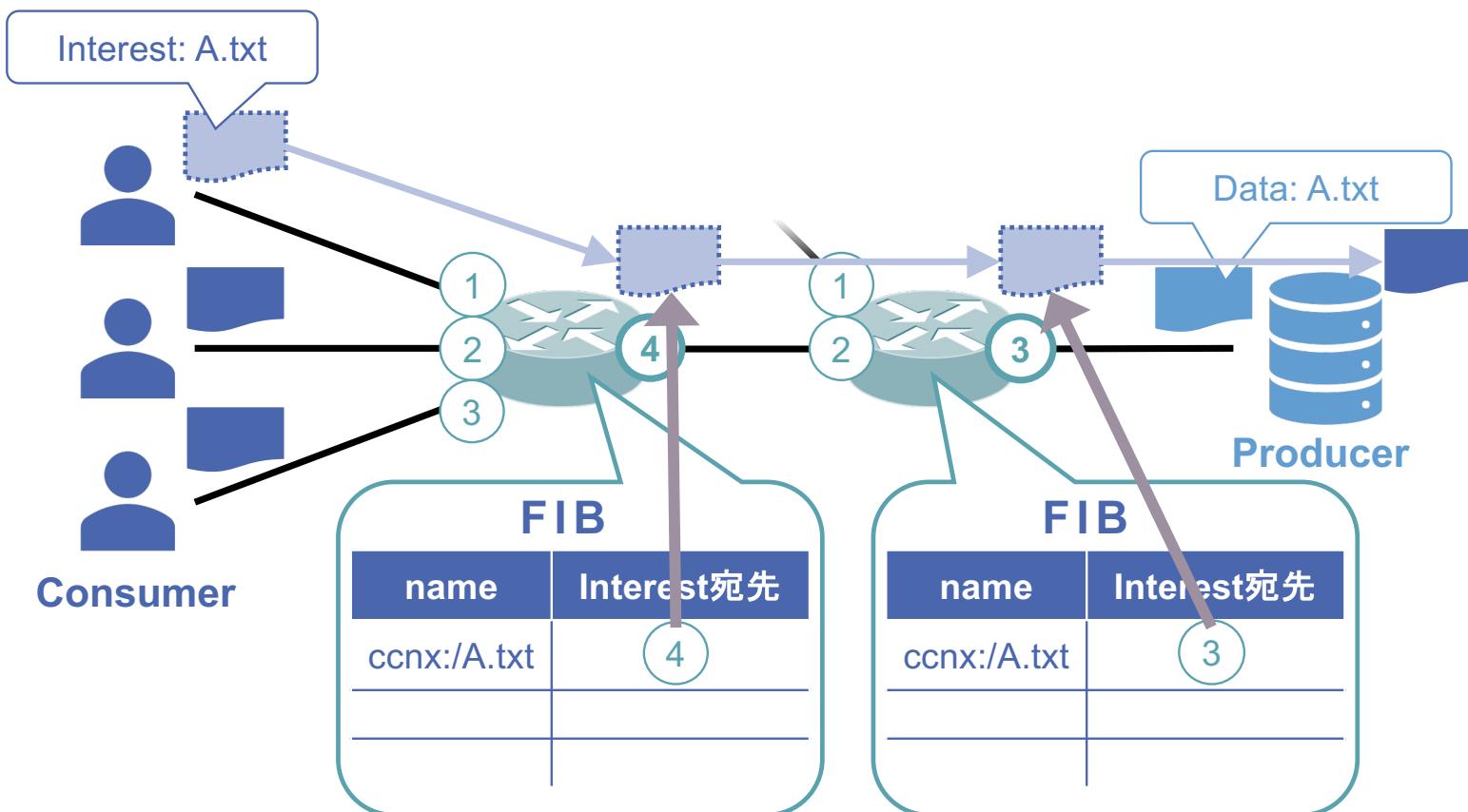
CCN の通信例

- CCN ルータはどのように通信を効率化するのか？
→例:3人の Consumer が ccnx:/A.txt を要求



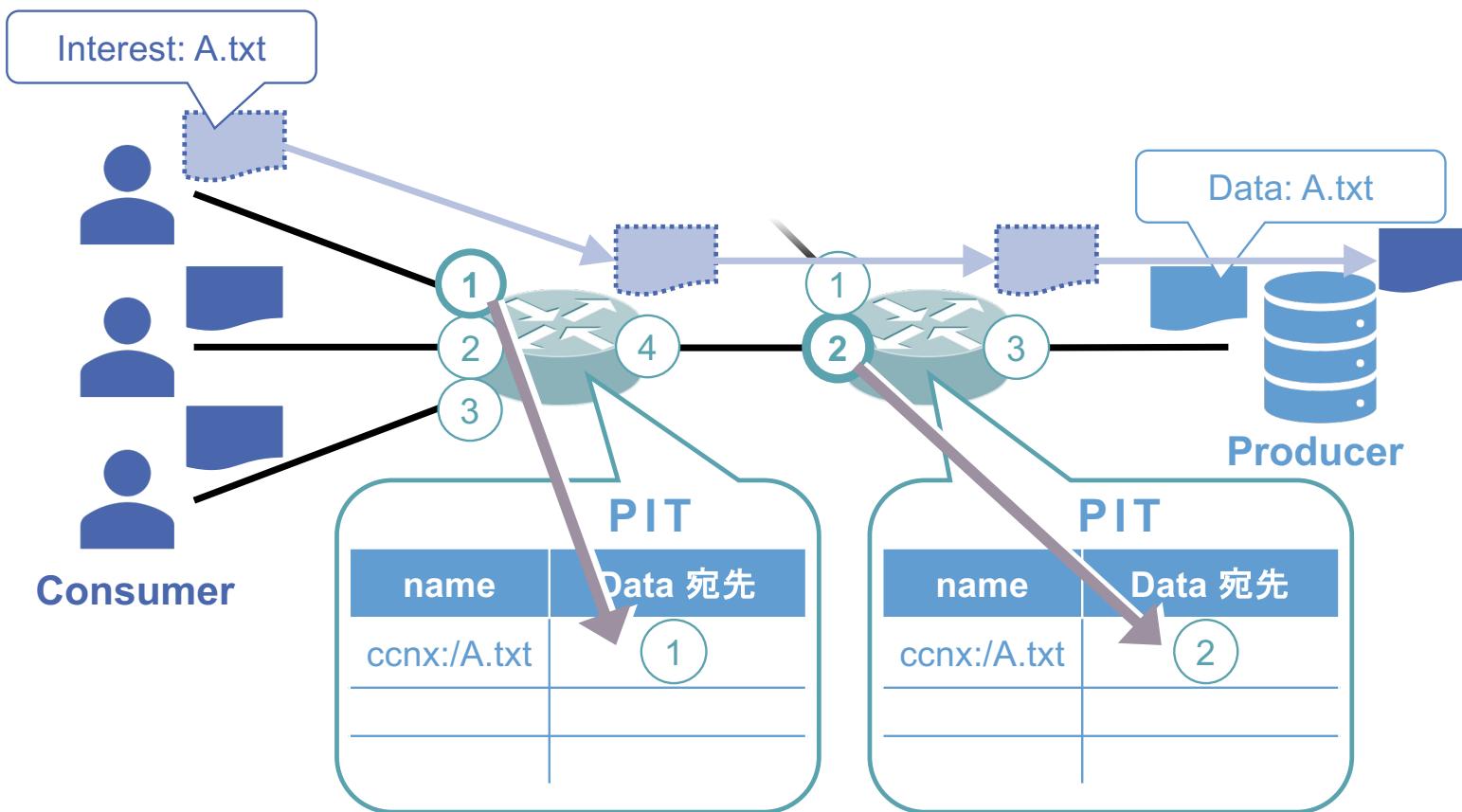
(1) Interest 転送

- 1人目の Consumer が Interest パケットを送出する
 - CCN ルータは **Forwarding Information Base (FIB)** に従つて Interest パケットを転送する



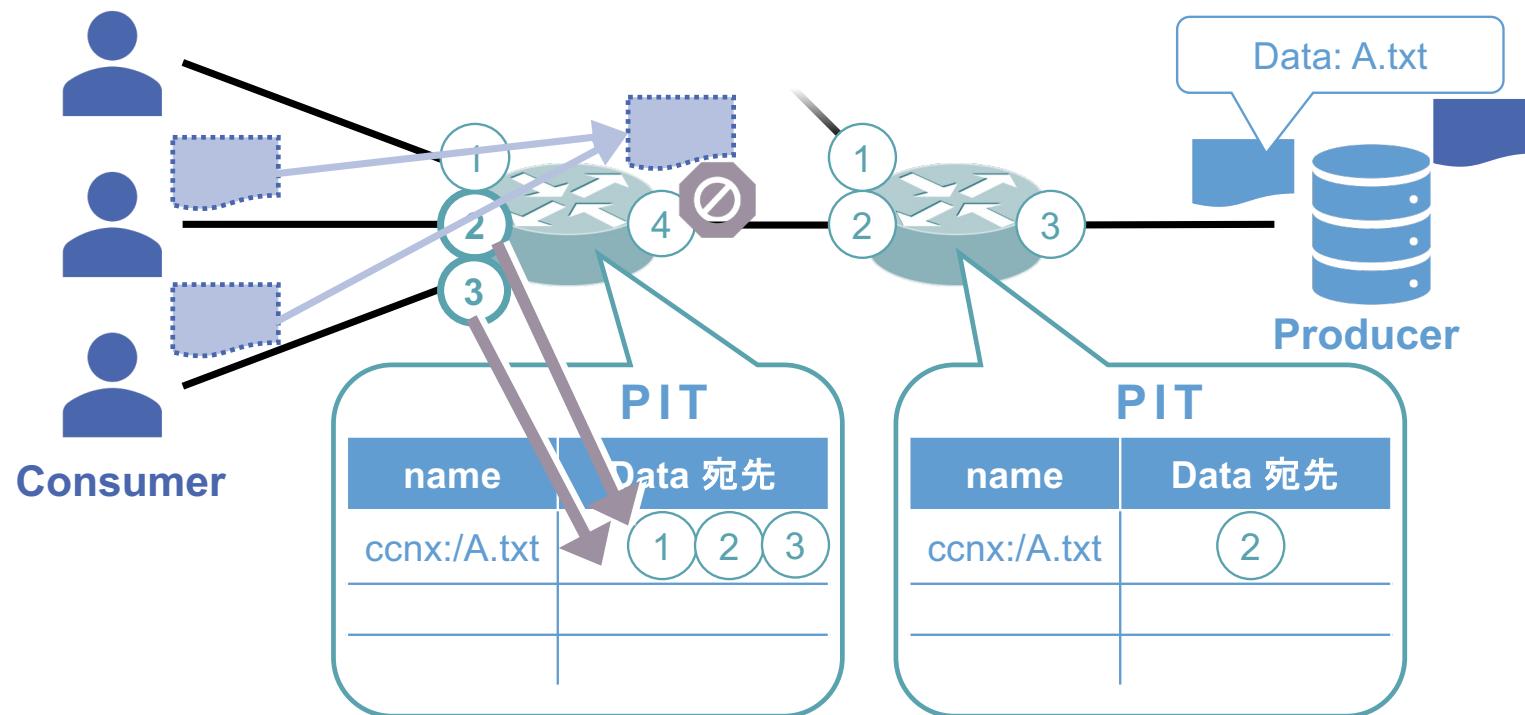
(2) 要求待ち Interest の記憶

- ルータは Interest を転送すると同時に、要求のあったポートを **Pending Interest Table (PIT)**に記憶する
 - 後で Data パケットの返送先として利用する



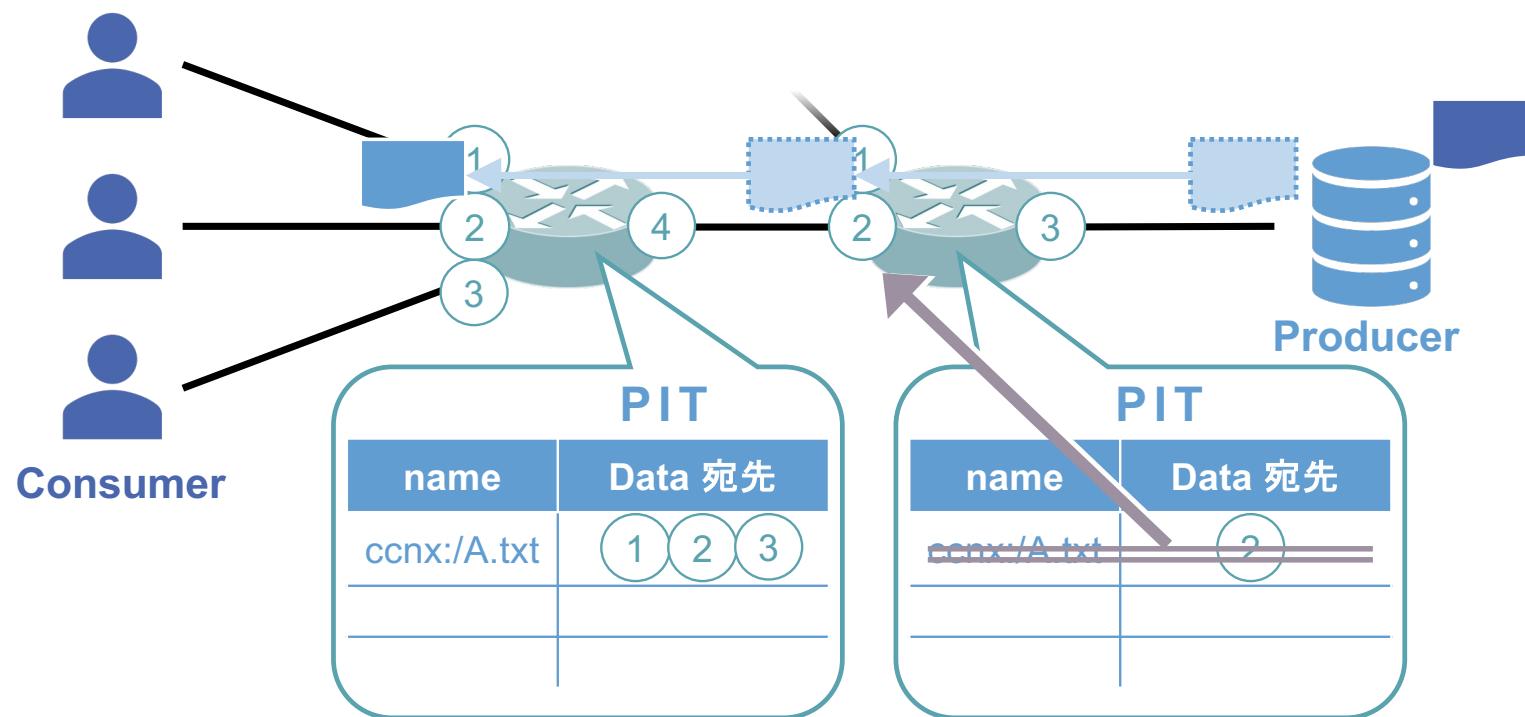
(3) 重複 Interest の集約

- Data パケットが返ってくるまでの間に、他の2人が Interest パケットを送出すると、ルータが集約する
 - ルータは以前と同じ要求だと分かるので転送はしない
 - PIT に3ポート分の情報が記憶される



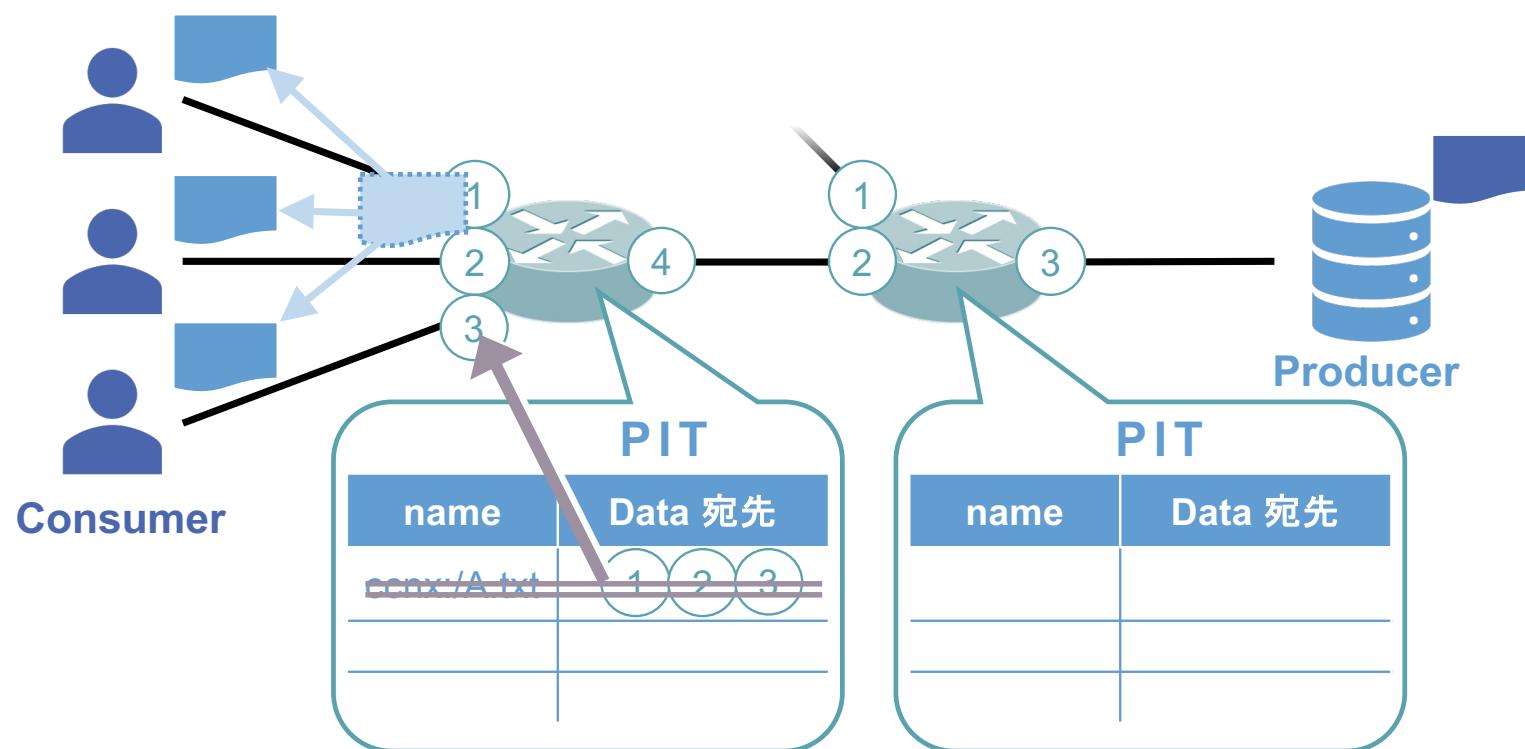
(4) Data パケットの返送

- Producer は Data パケットを返送する
 - Data パケットが転送されると要求が満たされたとみなし、PIT エントリは削除される
 - PIT エントリが無い限り Data パケットは転送されない



(5) Data パケットのマルチキャスト

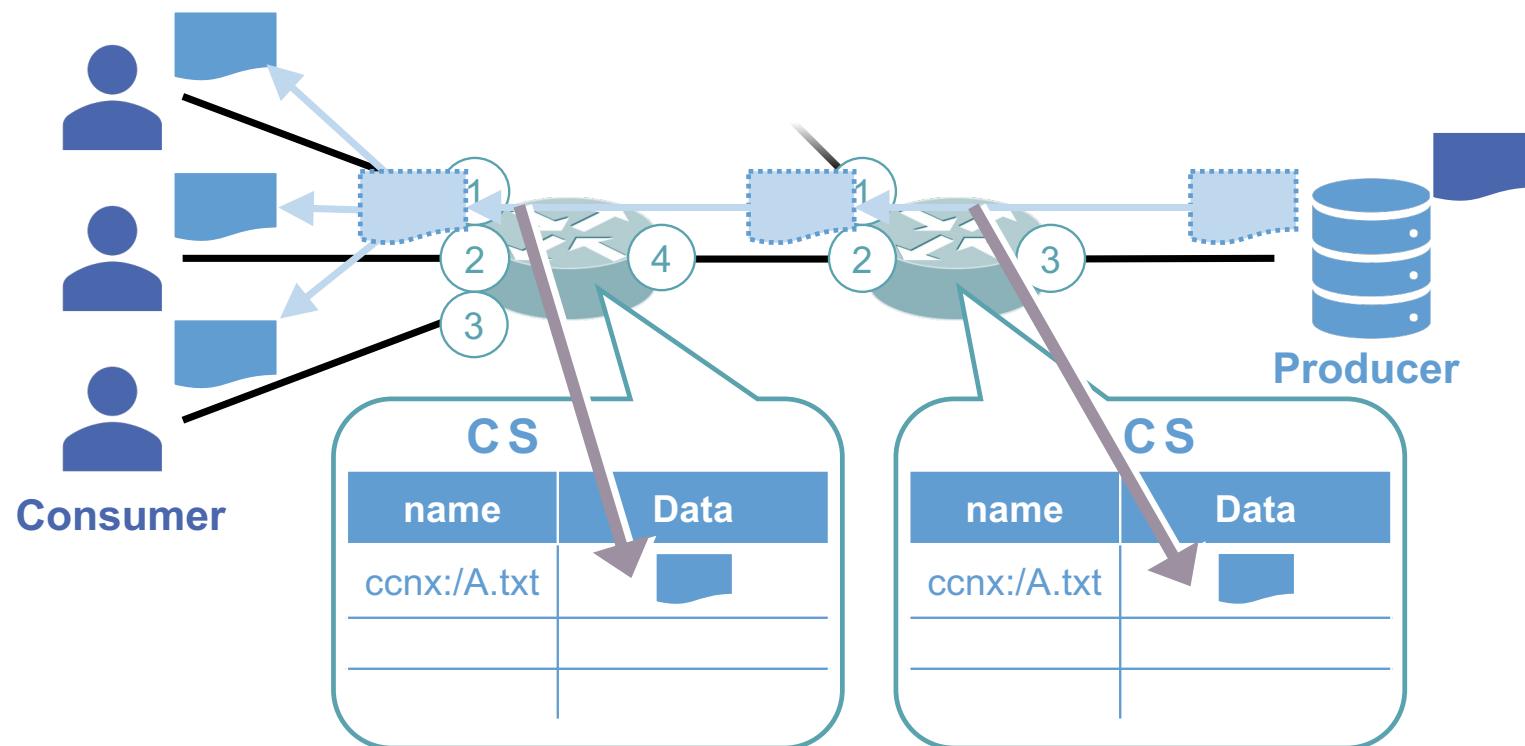
- PIT に複数の要求が集約されている場合は、すべての要求に
対して**マルチキャストされる**
 - 結果的に Producer が送出するパケットは1つで済み、
サーバの負荷が軽減される





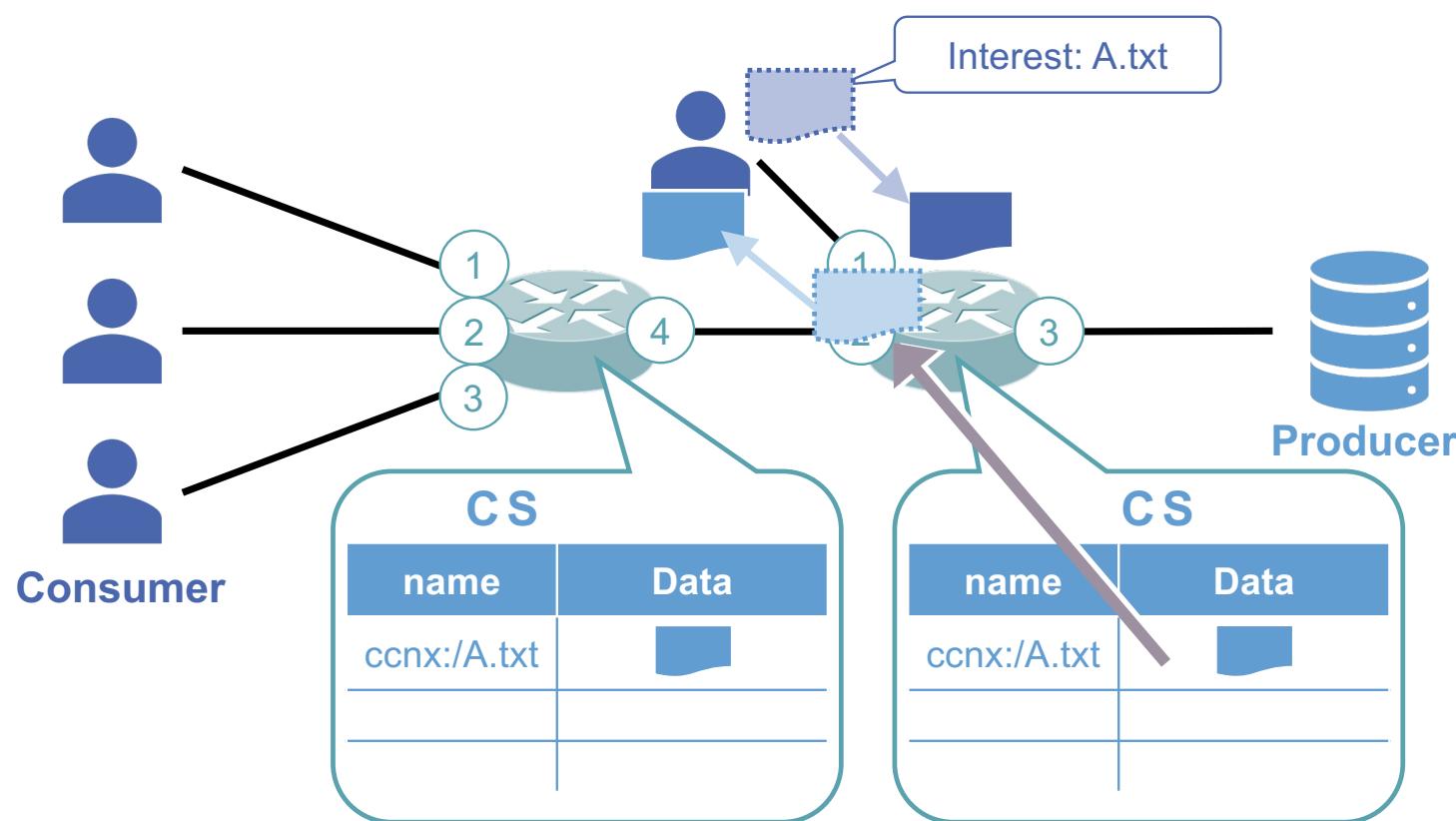
(6) Data パケットのキャッシング

- Data パケットを転送したルータはその Data パケットを **Content Store (CS)** にキャッシングする



(7) ルータによるキャッシュ応答

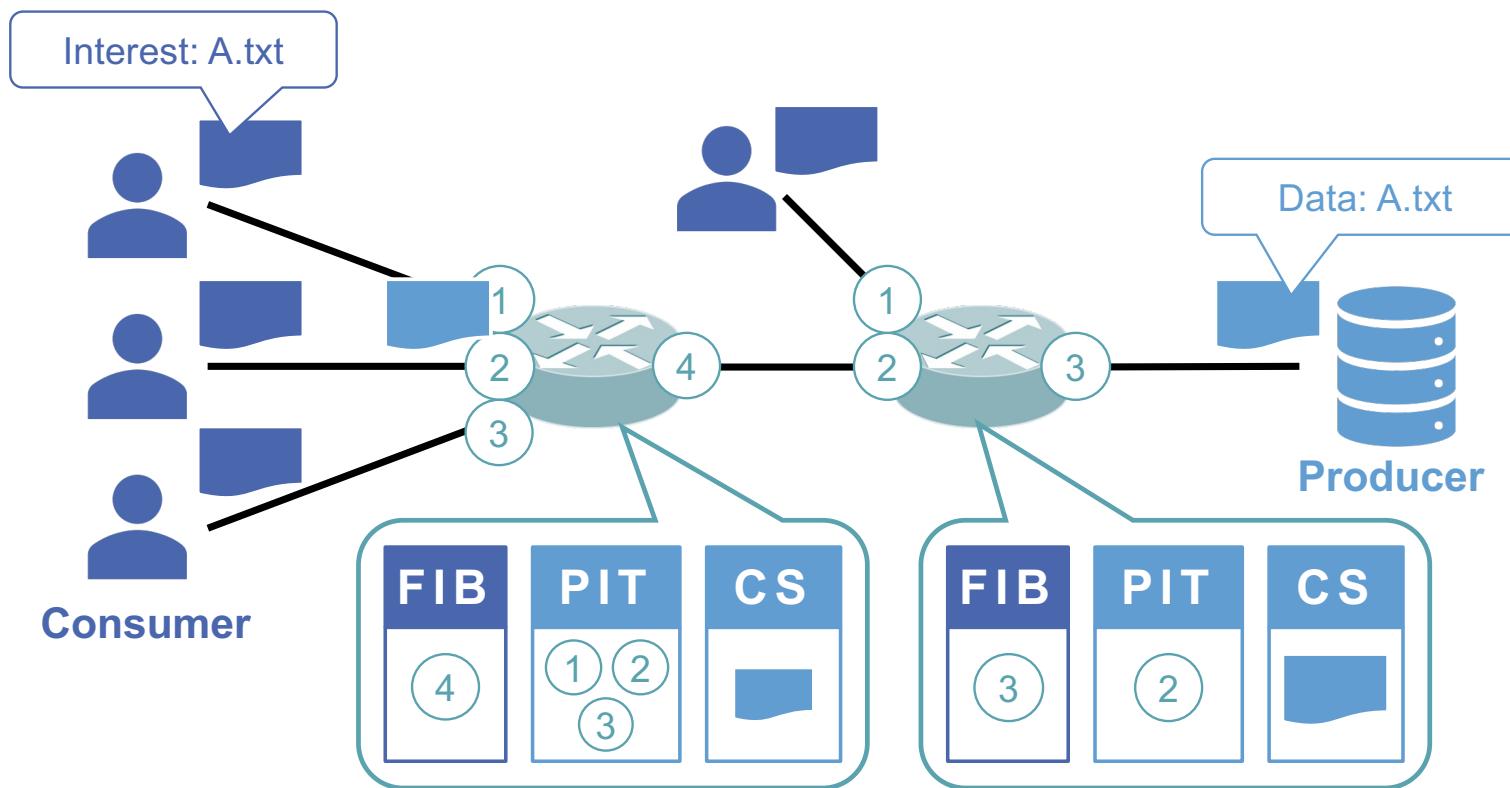
- 別の Consumer が後から同じ A.txt を要求
 - ルータはコンテンツ名を見れば同じだと分るので Producer に転送せず **直接 Data パケットを返送する**





CCN の通信例：まとめ

- CCN ルータは FIB・PIT・CS の3つのテーブルによって効率的なコンテンツの取得・配布をサポート



ICN のオープンソースソフトウェア実装： Cefore(セフォール)

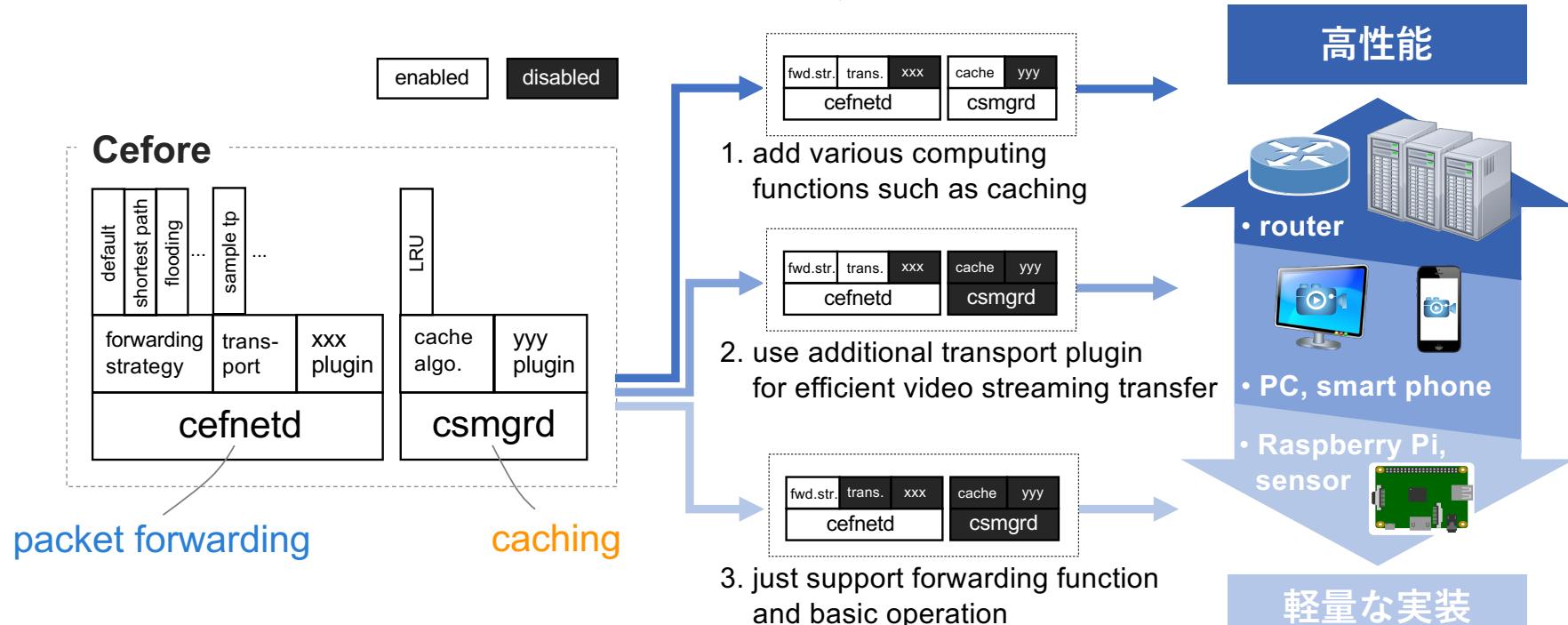


- NICT で開発する日本発 ICN 通信用ソフトウェアプラットフォーム
 - 詳細は次ページ以降で説明
 - IRTF の標準化文書 (RFC8569, RFC8609) にて定義された CCNx 1.0 プロトコルに準拠
- 設計実装
 - cefnetd/csmgrd
 - ICN packet forwarder engine および caching engine(公開)
 - cefpyco
 - Cefore アプリケーション開発支援用の Python ライブラリ(公開)
 - Cefbabel
 - Cefore (cefnetd) 用ルーティングプロトコル
 - Cefore-Emu
 - Mininet ベースの ICN エミュレータ(非公開)
 - CeforeSim
 - ns-3 ベースの ICN シミュレータ(非公開)



Cefore の概要

- 軽量かつ汎用的な CCN ソフトウェア実装

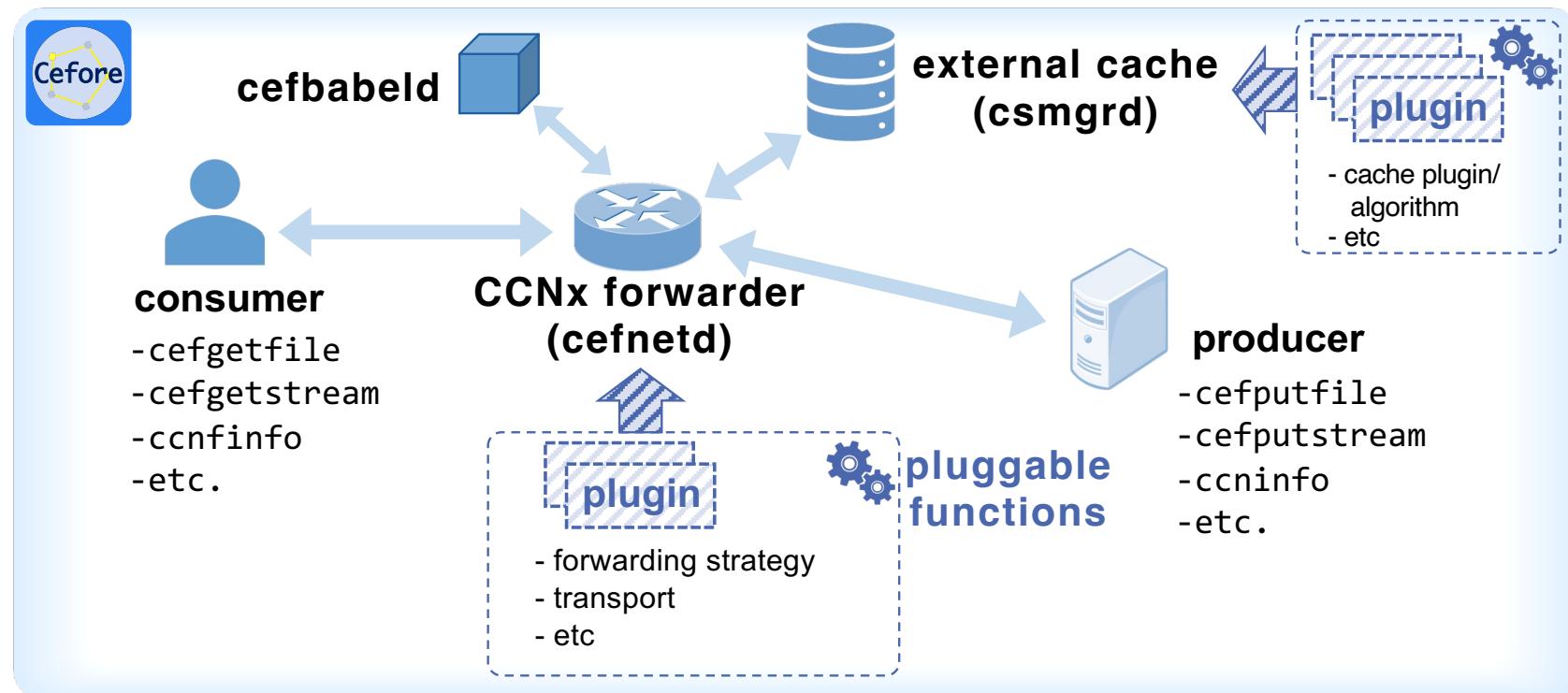


- リソースの乏しいセンサーノードでは**軽量構成**
- 最小機能以外はプラグインまたは外部機能として**機能拡張可能**



Cefore ソフトウェアパッケージ

- Cefore は ICN 通信に必要なすべての機能を “all-in-one package” として提供
 - アプリ、ルータ(転送部・キャッシュ部)、ネットワーク状態観測ツール





Ceforeの仕様

- 開発言語:C言語
- OS
 - Linux(ubuntu 22.04 or later)
 - macOS
 - Raspberry Pi OS
- CCNx-1.0のパケットフォーマットに準拠*
 - Type-Length-Value(TLV)フォーマット
 - Cefore独自のプロトコル拡張はOptional Hop-by-hopヘッダに記述
- TCP/IP上でICN通信
 - CCNx/UDP, CCNx/TCP

* "CCNx Messages in TLV Format", <https://tools.ietf.org/html/rfc8609>



cefore.net で公開 → Github へ移行

- マニュアルやソースコードをダウンロード可能
 - ICN 研究会の Cefore チュートリアル等も参照

Cefore

Information Centric Networking Platform

Top About Instruction Download Plugins Development Simulator Emulator Testbed Android Publications Mailing list Blog

Download

- source code (cefore-0.8.1.zip)
- CUTEi vm appliance(cuteivm -v1.1.tar.xz)
- Cefore for Android
- Applications

Utilities

- cefcyco-0.3.8 (Python package for cefore app)
- Tiger Plugin (Security)
- ABCD Plugin (Transport)
- SNAKE Plugin (Routing)
- Plugin (Cache)

<https://cefore.net/>

ICN 電子情報通信学会
情報指向ネットワーク技術特別研究専門委員会
Technical Committee on Information-Centric Networking(ICN) ☰

| Cefore

HOME > Cefore

ION/CCN 通信を実現するソフトウェアプラットフォーム Cefore

◦ Cefore の概要
ION/CCN の研究促進や実用化を目的とし、情報通信研究機構(NICT)はCCNベースのオープンソースソフトウェアであるCeforeを開発しています。Ceforeの特徴は「基本機能と拡張機能の分離」と「拡張機能開発の容易性」です。Ceforeは(1)CCNメッセージのエンコード/デコード、FIBとPITの管理やルーティングなどのCCNの基本機能(2)、基本機能以外の拡張機能を実装する「plugins」(ラグランジアン)、(3)ネットワーク内キャッシュ(Content store)など在内的部品を構成します。

https://www.ieice.org/cs/icn/?page_id=1208

cefore

Github repository

Cefore

1. Overview

Cefore is a software platform that enables ICN-based communications using CCNx-1.0 messages defined in [RFC8569](#) and [RFC8609](#). Cefore consists of (1) "cefned" daemon, which implements the CCN's basic function such as CCN Interest/Data handling, and FIB and Pending Interest Table (PIT) management, (2) "csmgrd" daemon, which implements Content Store, (3) arbitrary plugin library implementations that extend cefnrd's or csmgrd's functionalities, and (4) network tools/commands and sample applications.

1.1. Environments

Cefore (ver. 0.11.0) can run on top of Linux (Ubuntu) and macOS as the following table. At least 4 GB of memory (RAM) and 4 CPU cores are required to run csmgrd (Content Store manager daemon).

1.2. Components

Cefore consists of the functions listed in the following table. Some functions can be enabled or disabled during the build, and when

<https://github.com/cefore>



<https://cefore.net/>



<https://github.com/cefore>



Cefpyco の概要

- Cefpyco (CEFore PYthon COmpact package)
 - Cefore アプリ開発用の Python パッケージ
 - C言語より容易に Cefore アプリを開発可能
 - 例: Interest を送信するコード

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <ctype.h>
5 #include <cefore/cef_define.h>
6 #include <cefore/cef_client.h>
7 #include <cefore/cef_frame.h>
8 #include <cefore/cef_log.h>
9
10 int main(int argc, char *argv[]) {
11     CefT_Client_Handle fhdl;
12     CefT_Interest_TLVs params_i;
13     int res;
14     cef_log_init ("cefpco");
15     cef_frame_init();
16     res = cef_client_init(port_num, conf_path);
17     if (res < 0) return -1;
18     fhdl = cef_client_connect();
19     if (fhdl < 1) return -1;
20     memset(&params_i, 0, sizeof(CefT_Interest_TLVs));
21     res = cef_frame_conversion_uri_to_name("ccn:/test", params_i.name);
22     if (res < 0) return -1; // Failed to convert URI to name.;
23     params_i.name_len = res;
24     params_i.hoplimit = 32;
25     params_i.opt.lifetime_f = 1;
26     params_i.opt.lifetime = 4000ull; /* 4 seconds */
27     params_i.opt.symbolic_f = CefC_T_OPT_REGULAR;
28     params_i.chunk_num_f = 1;
29     params_i.chunk_num = 0;
30     cef_client_interest_input(fhdl, &params_i);
31     if (fhdl > 0) cef_client_close(fhdl);
32     return 0;
33 }
```

C言語版

Python版

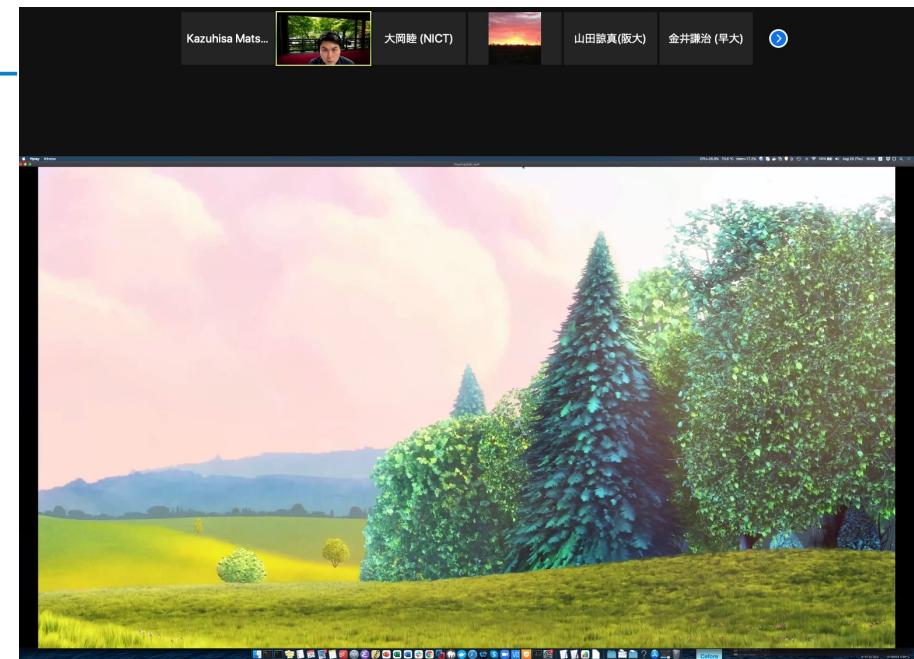
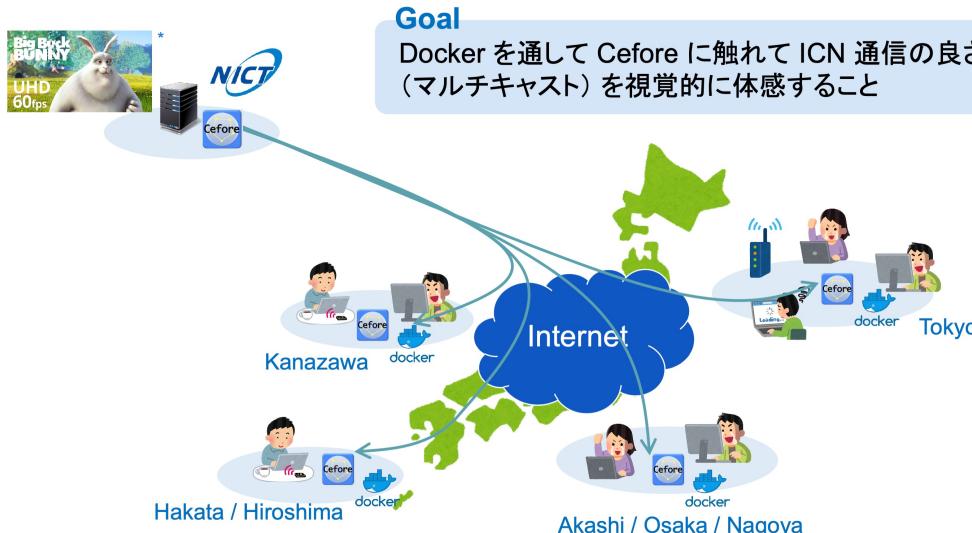
33行→4行

```
1 import cefpyco
2
3 with cefpyco.create_handle() as h:
4     h.send_interest("ccn:/test", 0)
```

インターネット規模のリアルタイムビデオ配信

- IEICE ICN summer workshop 2021 [オンライン開催]
 - Cefore/Docker ハンズオン
 - Cefore/Docker platforms* を活用したICNマルチキャスト通信
 - NICT (Tokyo) にサーバを配置
 - 参加者は日本全国から異なる通信品質で、それらに応じたビデオ配信を受信

 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]

Cefore の基本操作: 起動・停止・状態確認から Interest / ContentObject によるデータ取得(キャッシュ利用)まで



Cefore デーモン (cefnetd/csmgrd) の起動・停止

practi

*Ubuntu 側で実行

- cefnetd の起動

```
% cefnetdstart
```

- cefnetd の停止

```
% cefnetdstop
```

- csmgrd の起動

```
% csmgrdstart
```

- csmgrd の停止

```
% csmgrdstop
```

NOTE: When you configure to use both cefnetd and csmgrd, first you need to start csmgrd, and then start cefnetd.



Cefnetd のステータス確認

practi

- Checking the status of cefnetd
 - **cefstatus**
 - CCNx ver.
 - Rx/Tx Interest #
 - Rx/Tx ContentObject #
 - Cache Mode
 - Face Table
 - FIB
 - PIT
 - **cefstatus -v**
 - 起動している cefore のバージョン確認

```
root@router:/cefore# cefstatus
Version          : 1
Port             : 9896
Rx Interest     : 0 (RGL[0], SYM[0], SEL[0])
Tx Interest     : 0 (RGL[0], SYM[0], SEL[0])
Rx ContentObject: 0
Tx ContentObject: 0
Cache Mode      : Excache
Faces : 6
  faceid = 4 : IPv4 Listen face (udp)
  faceid = 0 : Local face
  faceid = 5 : IPv6 Listen face (udp)
  faceid = 6 : IPv4 Listen face (tcp)
  faceid = 16 : Local face
  faceid = 7 : IPv6 Listen face (tcp)
FIB(App) :
  Entry is empty
FIB :
  Entry is empty
PIT(App) :
  Entry is empty
PIT :
  Entry is empty
```

```
root@consumer:/cefore# cefstatus -v
Cefore version 0.10.0g
```



FIB 追加・削除 (static routing)

practi

- **cefroute**
 - Insertion
 - `cefroute add ccnx:/aaa udp pi1.local`
 - Deletion
 - `cefroute del ccnx:/aaa udp pi1.local`
- Alternative: preparing an FIB configuration file
 - `/usr/local/cefore/cefnetd.fib`
 - `cefnetd` automatically loads this file when starting its process
- Routing Protocol
 - TBA

```
root@producer:/cefore# cat /usr/local/cefore/cefnetd.fib
ccnx:/ udp [ラズベリーパイのIPアドレス]
```

+ Setting FIB using cefnetd.fib

```
root@producer:/cefore# cefstatus
Version      : 1
Port         : 9896
Rx Interest   : 0 (RGL[0], SYM[0], SEL[0])
Tx Interest   : 0 (RGL[0], SYM[0], SEL[0])
Rx ContentObject : 0
Tx ContentObject : 0
Cache Mode    : Excache
Controller    : 192.168.0.99
Faces : 7
  faceid = 4 : IPv4 Listen face (udp)
  faceid = 0 : Local face
  faceid = 16 : address = 10.0.1.1:9896 (udp)
  faceid = 17 : Local face
  faceid = 5 : IPv6 Listen face (udp)
  faceid = 6 : IPv4 Listen face (tcp)
  faceid = 7 : IPv6 Listen face (tcp)
FIB(App) :
  Entry is empty
FIB : 1
  ccnx:/example
    Faces : 16 (-s-) RtCost=0
PIT(App) :
  Entry is empty
PIT :
  Entry is empty
```

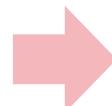


Csmgrd のステータス確認

- Checking the status of csmgrd
 - csmgrstatus NAME_PREFIX

+ initial state

```
root@producer:/cefore# csmgrstatus ccnx:/  
  
Connect to 127.0.0.1:9799  
***** Connection Status Report *****  
All Connection Num : 1  
  
***** Cache Status Report *****  
Number of Cached Contents : 0
```



+ after putting the 3 contents

```
root@producer:/cefore# csmgrstatus ccnx:/  
  
Connect to 127.0.0.1:9799  
***** Connection Status Report *****  
All Connection Num : 1  
  
***** Cache Status Report *****  
Number of Cached Contents : 3  
  
[0]  
Content Name : ccnx:/ccc  
Version : None  
Content Size : 4 Bytes  
Cache Hit : 0  
Request Count : 0  
Freshness : 290 Sec  
Elapsed Time : 8 Sec  
  
[1]  
Content Name : ccnx:/bbb  
Version : None  
Content Size : 4 Bytes  
Cache Hit : 0  
Request Count : 0  
Freshness : 283 Sec  
Elapsed Time : 14 Sec  
  
[2]  
Content Name : ccnx:/aaa  
Version : None  
Content Size : 4 Bytes  
Cache Hit : 0  
Request Count : 0  
Freshness : 275 Sec  
Elapsed Time : 22 Sec
```

+ specify name prefix

```
root@producer:/cefore# csmgrstatus ccnx:/aaa  
  
Connect to 127.0.0.1:9799  
***** Connection Status Report *****  
All Connection Num : 1  
  
***** Cache Status Report *****  
Number of Cached Contents : 1  
  
[0]  
Content Name : ccnx:/aaa  
Version : None  
Content Size : 4 Bytes  
Cache Hit : 0  
Request Count : 0  
Freshness : 268 Sec  
Elapsed Time : 29 Sec
```



Cefore のチューニング

- 主要なパラメータ
 - cefnetd.conf
 - CS_MODE
 - 0: no content store [default]
 - 1: cefnetd's local cache
 - 2: external content store (csmgrd)
 - FORWARDING_STRATEGY
 - default: Forward the Interest to a face in the longest-prefix-matched(LPMed) FIB entry [default]
 - flooding: Forward the Interest to all the faces registered in the LPMed FIB entry
 - shortest_path: Forward the Interest to the face that has the minimum routing cost in the LPMed FIB entry
 - csmgrd.conf
 - CACHE_CAPACITY
 - The maximum number of cached ContentObjects in csmgrd
 - 819,200 [default]
 - CACHE_TYPE
 - filesystem: cache located on UNIX filesystem [default]
 - memory: cache located on memory (RAM)
 - CACHE_ALGORITHM
 - libcsmgrd_fifo
 - libcsmgrd_lru
 - libcsmgrd_lfu



設定パラメータ*

Parameter	Description	Default
CEF_LOG_LEVEL	Specifies the log output type for the cefnetd. Range: 0 <= n <= 3	0
PORT_NUM	Port number cefnetd uses. Range: 1024 < p < 65536 If the startup option "-p port_num" is used, the port number specified by the "-p port_num" option takes precedence over this parameter.	9896
PIT_SIZE	Max number of PIT entries. Range: 1 < n < 65536	2048
FIB_SIZE	Max number of FIB entries. Range: 1 < n < 65536	1024
CS_MODE	ContentStore mode Cefore uses.	0: No cache used 1: cefnetd's local cache 2: csmgrd
LOCAL_CACHE_CAPACITY	Max number of Cobs to use for the local cache in cefnetd. Range: 1 < n <= 8000000 Approximate memory usage: Cob size * 2 * num. of Cobs.	65535
CSMGR_NODE	csmgrd's IP address	localhost
CSMGR_PORT_NUM	TCP port number used by csmgrd to connect cefnetd. Range: 1024 < p < 65536	9799
FORWARDING_STRATEGY	Forwarding strategy when sending Interest messages. default : Forward the Interest to a face in the longest-prefix-matched (LPMed) FIB entry flooding : Forward the Interest to all the faces registered in the LPMed FIB entry shortest_path: Forward the Interest to the face that has the minimum routing cost in the LPMed FIB entry	0

*README is available at: <https://cefore.net/doc/Readme.html>



Cefputfile/cefgetfile によるファイル転送

practi

- シナリオ
 - producer @ raspi がコンテンツ ccnx:/hello/YOUR_NAME.txt を csmgrd に upload
 - consumer @ ubuntu がアップロードされたコンテンツを要求
- 事前準備
 - /usr/local/ceofre/cefnetd.conf の CS_MODE=2 (csmgrd) に変更
- Producer

```
$ dd if=/dev/zero of=100KB-file.txt bs=1024 count=100
$ sudo csmgrdstart
$ sudo cefnetdstart
$ cefputfile ccnx:/hello/100KB-file -f ./100KB-file.txt -t 3600 -e 3600
$ csmgrstatus ccnx:/hello
```

- Consumer
 - 経路設定

```
$ cefroute add ccnx:/hello udp pi1.local
```

- データ要求

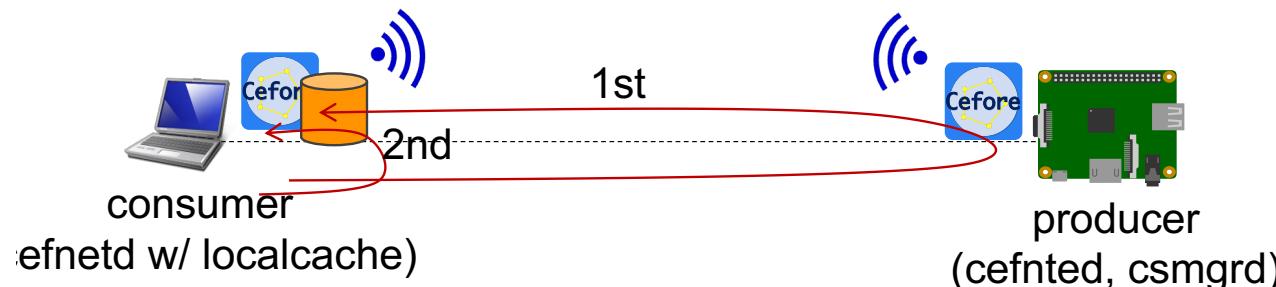
```
$ cefgetFile ccnx:/hello/100KB-file -s 1 -f ./output.txt
```



In-network caching の効果

- Consumer (Ubuntu) の cefnetd を CS_MODE=1にして、local cache を有効化することで、キャッシングによる効果を体感
- 手順
 - Consumerにおいて cefnetd を停止
 - /usr/local/cefore/cefnetd.conf を編集し、CS_MODE=1 に変更 (default=0, no-cache)
 - cefnetd を再起動
 - cefgetfile で再度コンテンツを取得

```
$ sudo cefnetdstop
$ emacs -nw /usr/local/cefore/cefnetd.conf
$ sudo cefnetdstart
$ cfroute add ccnx:/hello udp pi1.local
$ cefstatus
$ cefgetfile ccnx:/hello/100KB-file -s 1 -f ./output-1st.txt
  # Consumer のローカルキャッシングにデータが保存される
$ cefgetfile ccnx:/hello/100KB-file -s 1 -f ./output-2nd.txt
```



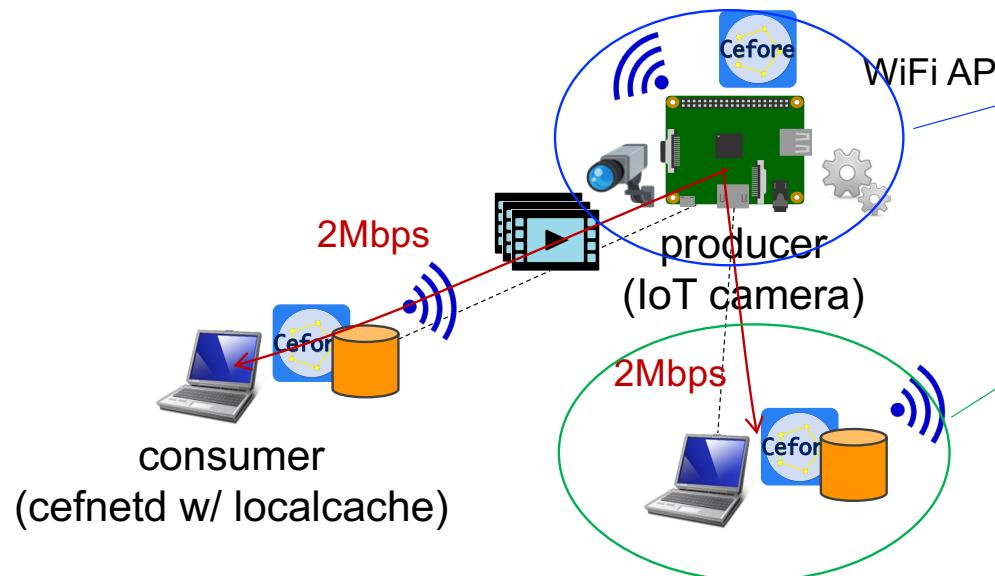
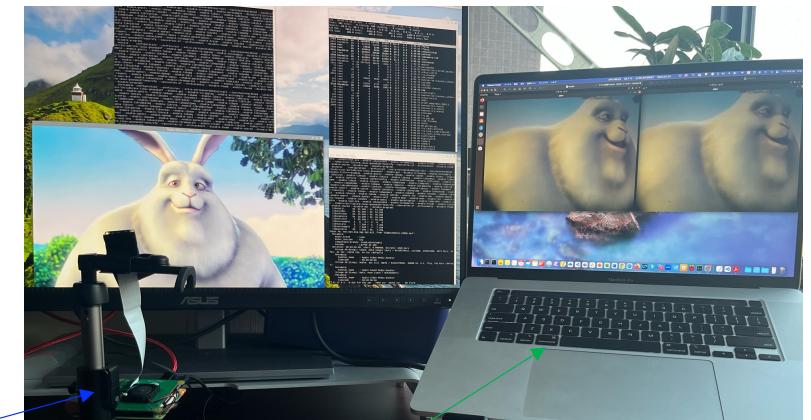
```
hayamizu@cefore:~/cefore$ cefgetfile ccnx:/hello/100KB-file -s 1 -f ./output-1st.txt
[cefgetfile] Start
[cefgetfile] Parsing parameters ... OK
[cefgetfile] Init Cefore Client package ... OK
[cefgetfile] Conversion from URI into Name ... OK
[cefgetfile] Checking the output file ... OK
[cefgetfile] Connect to cefnetd ... OK
[cefgetfile] URI=ccnx:/hello/100KB-file
[cefgetfile] Start sending Interests
[cefgetfile] Completed to get all the chunks.
[cefgetfile] Unconnect to cefnetd ... OK
[cefgetfile] Terminate
[cefgetfile] Rx Frames (All)          = 100
[cefgetfile] Rx Frames (ContentObject) = 100
[cefgetfile] Rx Bytes (All)          = 105500
[cefgetfile] Rx Bytes (ContentObject) = 102400
[cefgetfile] Duration              = 0.378 sec
[cefgetfile] Throughput            = 2235595 bps
[cefgetfile] Goodput                = 2169905 bps
[cefgetfile] Jitter (Ave)           = 3775 us
[cefgetfile] Jitter (Max)           = 9995 us
[cefgetfile] Jitter (Var)           = 856371 us
hayamizu@cefore:~/cefore$ cefgetfile ccnx:/hello/100KB-file -s 1 -f ./output-2nd.txt
[cefgetfile] Start
[cefgetfile] Parsing parameters ... OK
[cefgetfile] Init Cefore Client package ... OK
[cefgetfile] Conversion from URI into Name ... OK
[cefgetfile] Checking the output file ... OK
[cefgetfile] Connect to cefnetd ... OK
[cefgetfile] URI=ccnx:/hello/100KB-file
[cefgetfile] Start sending Interests
[cefgetfile] Completed to get all the chunks.
[cefgetfile] Unconnect to cefnetd ... OK
[cefgetfile] Terminate
[cefgetfile] Rx Frames (All)          = 100
[cefgetfile] Rx Frames (ContentObject) = 100
[cefgetfile] Rx Bytes (All)          = 105500
[cefgetfile] Rx Bytes (ContentObject) = 102400
[cefgetfile] Duration              = 0.013 sec
[cefgetfile] Throughput            = 71145578 bps
[cefgetfile] Goodput                = 69055045 bps
[cefgetfile] Jitter (Ave)           = 118 us
[cefgetfile] Jitter (Max)           = 186 us
[cefgetfile] Jitter (Var)           = 825 us
```

Cefore/ラズパイを用いた ライブ配信アプリ

ゴール: ラズパイで撮影した動画をノードPCから再生する

1. ラズパイを用いた ICN ライブ配信

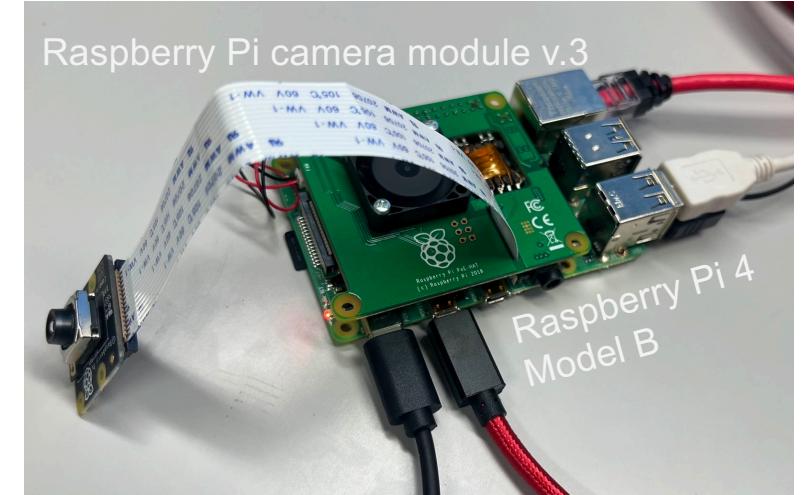
- ラズパイでは、publisher アプリが撮影した動画をCCNxパケットに変換してデータを送信
- 参加者は consumer アプリとして動画を受信するリクエスト(Symbolic Interest)を送信して、データストリームを受信
- 受信したデータをプレイヤーで再生





ラズベリーパイ + 標準カメラモジュールによる動画撮影

- libcamera-vid
 - options
 - --width 撮影する画像の横サイズ
 - --height 撮影する画像の縦サイズ
 - --bitrate 撮影する画像のビットレート
 - -n 動画のプレビュー表示をOFF
 - -t 撮影時間[ms]、t=0 でエンドレスで撮影
 - -o ファイル出力先



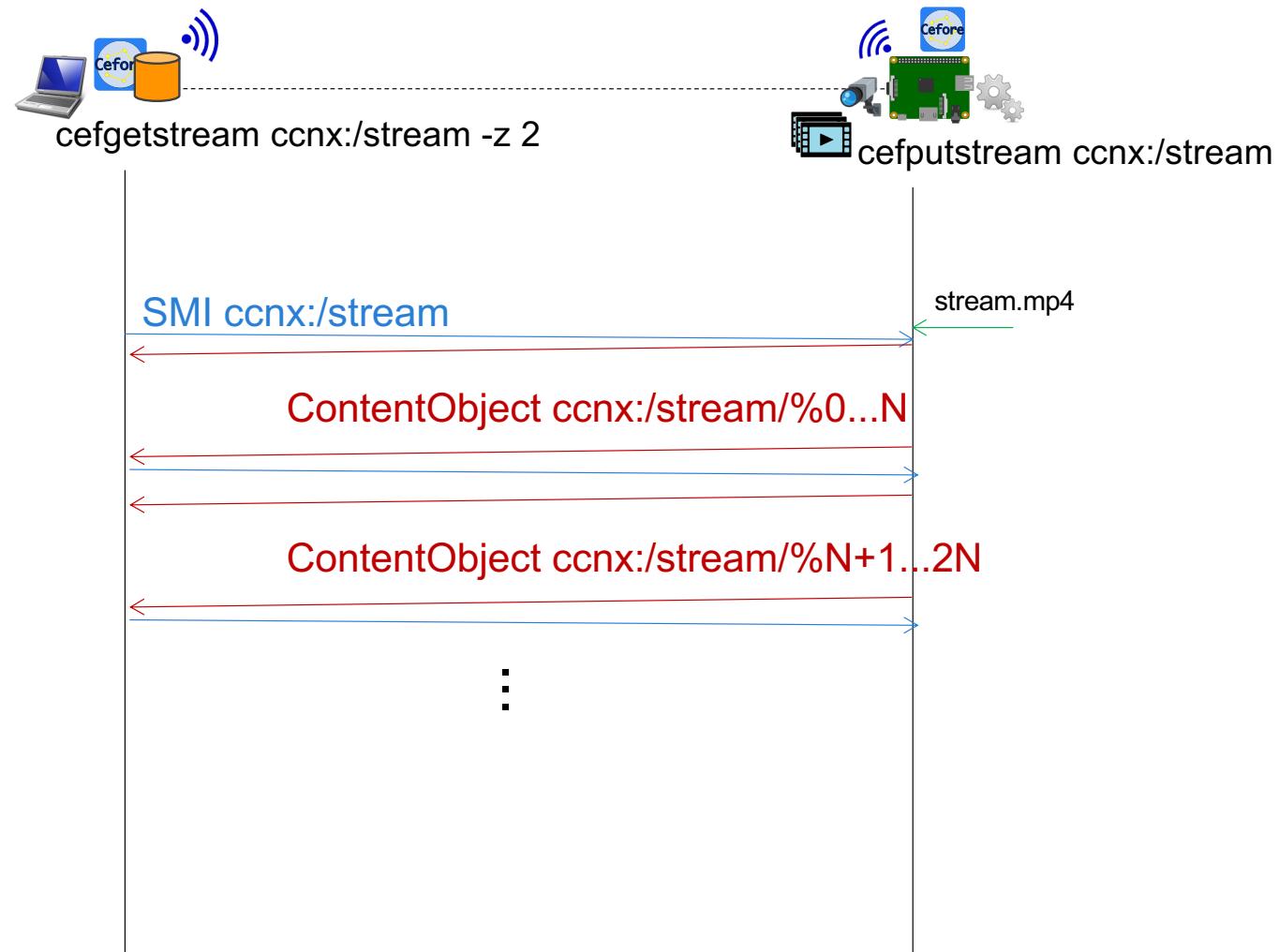
• ライブエンコーディング設定*

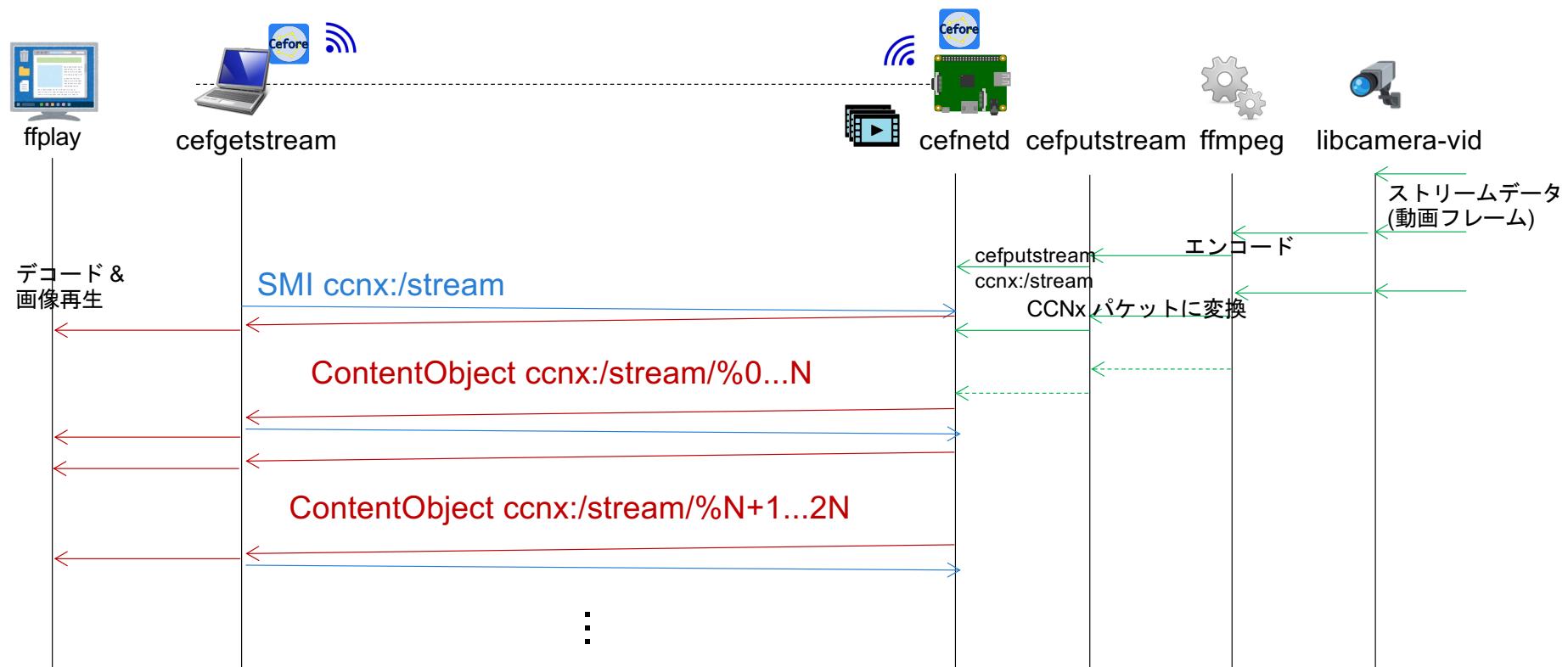
取り込み解像度 / フレームレート	最小ビットレート設定(Mbps)	最大ビットレート設定(Mbps)
2160p (4K) 、 30 fps	8 Mbps	35 Mbps
1440p 、 30 fps	5 Mbps	25 Mbps
1080p 、 30 fps	3 Mbps	8 Mbps
240p ~ 720p 、 30 fps	3 Mbps	8 Mbps

* <https://support.google.com/youtube/answer/2853702?hl=ja> より引用



通信シーケンス(概要)

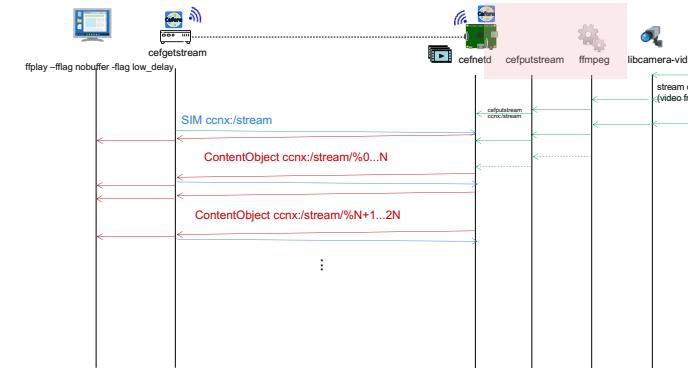






Publisher 側のプログラム

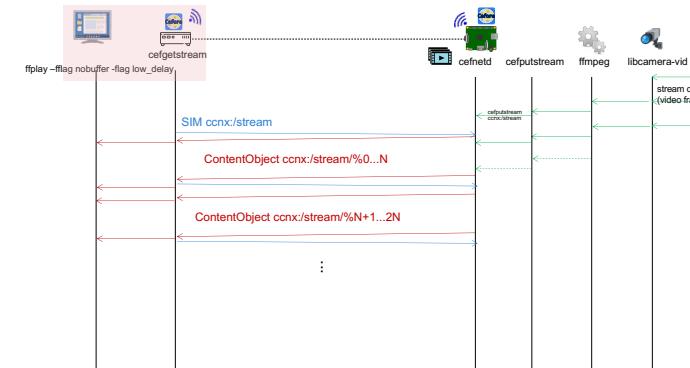
- **ffmpeg**
 - `ffmpeg -i - -vcodec mpeg4 -b:v 2M -acodec aac -f mpegs -`
 - `-vcodec` ビデオのコーデックを指定(MP4)
 - `-b:v` encoding rate を指定(2Mbps)
 - `-acodec` 音声のコーデック(AAC)
 - `-f` フォーマットでMPEG-TSを指定しリアルタイムビデオに変換
- **ceputstream**
 - `ceputstream ccnx:/stream -r 2 -b 1400 -t 3600 -e 3600`
 - `cebefore` 用のリアルタイム通信(送信)用ツール
 - データの標準出力に対応し、Cobを送信
 - 名前、送信レート (-r), ペイロードサイズ(-b), RCT/expiry (-t/-e) 等の設定が可能





Consumer 側のプログラム

- **cefgetstream :**
 - 取得したデータを標準出力するためのツール
 - `cefgetstream -z` オプションで Symbolic Interest (SMI) を送信可能
- **Symbolic Interest (SMI)**
 - 通常の CCNx/NDN の Interest (RGI: Regular Interest) に代わり、ストリーミングなどのリアルタイム通信にて使用される、NICT 独自拡張の Interest
 - -z option にて指定された送信間隔で keep-alive のようにセッションを維持しながら、通信を継続することで、Interest トラヒックを大幅に削減
- **ffplay**
 - 動画の再生用ツール
 - `ffmpeg` でリアルタイムエンコードされた動画をストリーミング再生



*K. Matsuzono and H. Asaeda, "NMRTS: content name-based mobile real-time streaming," in IEEE Communications Magazine, vol. 54, no. 8, pp. 92-98, August 2016, doi: 10.1109/MCOM.2016.7537182.



設定手順

practi

- ラズパイの操作(グループで一人だけが代表して実行)
 - libcamera-vid による動画撮影、ffmpeg によるリアルタイムエンコード、cefputstream によるCCNx ContentObject 送信を一度に実行

```
cefore:~/\$ libcamera-vid --width 640 --height 480 --bitrate 2048000 -n -t 0 -o /dev/stdout | ffmpeg -i - -vcodec mpeg4 -b:v 2M -acodec aac -f mpegs - | cefputstream ccnx:/stream -r 2 -b 1400 -t 3600 -e 3600
```

- livcam.bash を実行するのと同じ
- Ubuntuの操作(それがVM上で実行)
 - cefgetstream で受信したContentObject を ffplay でリアルタイムにデコード & 再生
- playback.bash を実行するのと同じ

```
cefore:~/\$ cefgetstream ccnx:/stream -z 2 | ffplay -fflags nobuffer -flags low_delay -
```

Cefore/Cefpyco を用いた Python アプリ開発



Cefpyco 演習の目次

- Cefpycoの概要
- Cefpycoのインストール (Ubuntuを対象)
- Cefpycoを用いた簡易Consumer, Publisherアプリの作成・実行
 - Regular Interest(通常のInterest)を使用する場合
 - Symbolic Interestを使用する場合
- Cefpycoを用いたリアルタイムビデオ受信・再生用Consumerアプリ
 - Symbolic Interestを利用



cefpycoの概要

- cefpyco (CEFore PYthon COnpact package)
 - ceforeアプリ開発用のpythonパッケージ
 - C言語で開発するより記述が楽
 - 例: Interest を送るコード

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <ctype.h>
5 #include <cefore/cef_define.h>
6 #include <cefore/cef_client.h>
7 #include <cefore/cef_frame.h>
8 #include <cefore/cef_log.h>
9
10 int main(int argc, char *argv[]) {
11     CefT_Client_Handle fhdl;
12     CefT_Interest_TLVs params_i;
13     int res;
14     cef_log_init ("cefpyco");
15     cef_frame_init();
16     res = cef_client_init(port_num, conf_path);
17     if (res < 0) return -1;
18     fhdl = cef_client_connect();
19     if (fhdl < 1) return -1;
20     memset(&params_i, 0, sizeof(CefT_Interest_TLVs));
21     res = cef_frame_conversion_uri_to_name("ccnx:/test", params_i.name);
22     if (res < 0) return -1; // Failed to convert URI to name. ;
23     params_i.name_len = res;
24     params_i.hoplimit = 32;
25     params_i.opt.lifetime_f = 1;
26     params_i.opt.lifetime = 4000ull; /* 4 seconds */
27     params_i.opt.symbolic_f = CefC_T_OPT_REGULAR;
28     params_i.chunk_num_f = 1;
29     params_i.chunk_num = 0;
30     cef_client_interest_input(fhdl, &params_i);
31     if (fhdl > 0) cef_client_close(fhdl);
32     return 0;
33 }
```

C言語版

33行→4行

Python版

```
1 import cefpyco
2
3 with cefpyco.create_handle() as h:
4     h.send_interest("ccnx:/test", 0)
```

Cefpycoのインストール (Ubuntu)



Cefpycoのダウンロード

- <https://github.com/cefore/cefpyco> ← ここからダウンロード
 - 解説(英語)あり

The screenshot shows the GitHub repository page for 'cefpyco'. The left sidebar lists files and folders: Atsushi Ooka (cefpyco 0.10.0.4), cefapp (cefpyco 0.10.0.1), src (cefpyco 0.10.0.4), test (cefpyco 0.10.0.4), .clang-format (cefpyco 0.10.0.1), .gitignore (cefpyco 0.10.0.1), CMakeLists.txt (cefpyco 0.10.0.1), LICENSE (cefpyco 0.10.0.1), README.html (cefpyco 0.10.0.1), README.md (cefpyco 0.10.0.1), build.bash (cefpyco 0.10.0.1), pyproject.toml (cefpyco 0.10.0.2), setup.cfg (cefpyco 0.10.0.4), and setup.py (cefpyco 0.10.0.1). The right side shows the 'About' section with no description, website, or topics provided. It includes links for Readme, View license, Activity, 9 stars, 2 watching, 4 forks, and a 'Report repository' button. Below that is the 'Releases' section with v0.10.0.4 (Latest, May 24) and '+ 4 releases'. Further down are sections for Packages (No packages published), Languages (C 56.6%, Python 29.3%, CMake 8.3%, Shell 5.8%), and Insights.

cefpyco / Public

Code Issues Pull requests Actions Projects Security Insights

master 1 Branch 10 Tags

Atsushi Ooka cefpyco 0.10.0.4

cefapp cefpyco 0.10.0.1

src cefpyco 0.10.0.4

test cefpyco 0.10.0.4

.clang-format cefpyco 0.10.0.1

.gitignore cefpyco 0.10.0.1

CMakeLists.txt cefpyco 0.10.0.1

LICENSE cefpyco 0.10.0.1

README.html cefpyco 0.10.0.1

README.md cefpyco 0.10.0.1

build.bash cefpyco 0.10.0.1

pyproject.toml cefpyco 0.10.0.2

setup.cfg cefpyco 0.10.0.4

setup.py cefpyco 0.10.0.1

README License

Clone

HTTPS GitHub CLI

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

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

About

No description, website, or topics provided.

Readme

View license

Activity

9 stars

2 watching

4 forks

Report repository

Releases 5

v0.10.0.4 Latest on May 24

+ 4 releases

Packages

No packages published

Languages

C 56.6% Python 29.3% CMake 8.3% Shell 5.8%

ここをクリック



ライブラリのインストール & Cefpycoビルド・インストール

ダウンロードしたcefpyco-master.zipを展開し、cefpyco-master(ディレクトリ)に移動

```
username:~/cefpyco-master$ ls
build.bash  cmake_install.cmake  install_manifest.txt  README.html  src  cefapp
CMakeLists.txt  LICENSE  README.md  test  CMakeCache.txt  CTestTestfile.cmake
Makefile  setup.cfg  CMakeFiles  dist  pyproject.toml  setup.py
username:~/cefpyco-master$ sudo apt-get install cmake python3-pip python3-dev python3-venv
username:~/cefpyco-master$ pip3 install --upgrade build
username:~/cefpyco-master$ pip3 install numpy click rich pytest pytest-sugar
username:~/cefpyco-master$ cmake .  ←
username:~/cefpyco-master$ make
username:~/cefpyco-master$ make install
```

cmake [スペース] [ピリオド]
ピリオドを忘れずに

```
username:~/cefpyco-master$ python3
>>> import cefpyco  ←
>>> (Ctr-D)
```

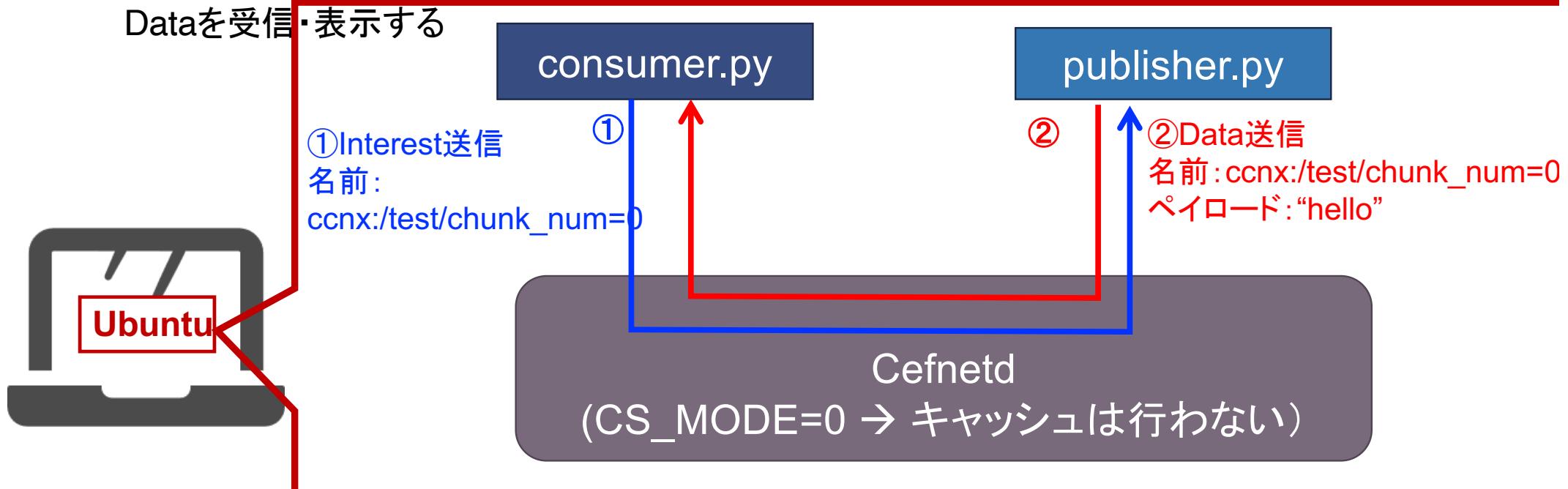
エラーが表示されないことを確認

Cefpyco を用いた簡易Consumer, Publisher アプリの作成・実行



ゴール:CefpycoでInterestとDataのやり取りを行う

- **publisher.py**の動作
 - ccnx:/testというコンテンツ名のチャンク番号0を要求するInterestを受信した場合、helloメッセージ(Data)を返答する
- **consumer.py**の動作
 - ccnx:/testというコンテンツ名のチャンク番号0を要求するInterestを送信し、Dataを受信・表示する





補足:pythonの文法

- C言語等のセミコロンや括弧の代わりにインデントで文・ブロックを表現

ブロックの範囲を
一目で見分けられる

```
# a=1, b=1のときはbと表示
# a=1, b≠1のときはaと表示
# a≠1のときは何もしない
if a == 1:
    if b == 1:
        print("b")
    else:
        print("a")
```

インデント幅が揃っていないと
バグ扱いなので要注意

```
if a == 1:
    print("correct")
else:
    print("error")
    print("error")
```

Tab文
字

エラー例
• 空白4文字と空白2文字
• 空白4文字とタブ1文字

- with構文:煩雑な開始/終了/例外処理を省略できる文法
 - 代表例:ファイルオープン・クローズ

with構文無しの場合

```
print("Begin.")
try:
    h = cefpyco.CefpycoHandle()
    h.begin()
    print("Do something.")
except Exception as e:
    print(e)
    # 例外処理
finally:
    h.end()
print("End.")
```

with構文を用いた場合

```
print("Begin.")
with cefpyco.create_handle() as h:
    print("Do something.")
print("End.")
```



consumer.py(コンシューマアプリ)

Regular Interest
(通常のInterest
を使用を想定

- ccnx:/testというコンテンツ名のチャンク番号0を要求する
Regular Interestを送信し、Dataを受信・表示する

consumer.py

```
import cefpyco

with cefpyco.create_handle() as handle:
    while True:
        handle.send_interest("ccnx:/test", 0)
        info = handle.receive()
        if info.is_succeeded and (info.name == "ccnx:/test") and (info.chunk_num == 0):
            print("Success")
            print(info)
            break
```

cefnetdへの接続

"ccnx:/test"というコンテンツ名のチャンク番号0のデータを要求するInterestを送信

データを待ち受け、受信後、データの情報をinfoに格納



publisher.py(パブリッシャーアプリ)

Regular Interest
(通常のInterest
を使用を想定

- ccnx:/testというコンテンツ名のチャンク番号0を要求する
Regular Interestを受信した場合、helloメッセージ(Data)を返答する

publisher.py

```
import cefpyco

with cefpyco.create_handle() as handle:  
    handle.register("ccnx:/test")  
    while True:  
        info = handle.receive()  
        if info.is_succeeded and (info.name == "ccnx:/test") and (info.chunk_num == 0):  
            handle.send_data("ccnx/test", "hello", 0)  
            print(info)
```

cefnetdへの接続
受信したいInterestのプレフィックス名を指定
データの名前
データのチャンク番号
ccnx:/test/chunk_num=0という名前のデータ要求する
Interestを受信した場合、
"hello"というメッセージを格納したDataを返信

※プログラムを強制終了したい場合は、Ctrl-Cを入力



実行手順

Regular Interest
(通常のInterest
を使用を想定)

※ cefnetdstartを起動する前に、/usr/local/cefore/cefore.confファイルにて、
CS_MODE=0 となっているか確認

```
username:~/cefpco-master$ cefnetdstart  
username:~/cefpco-master$ python3 publisher.py
```

ターミナル1

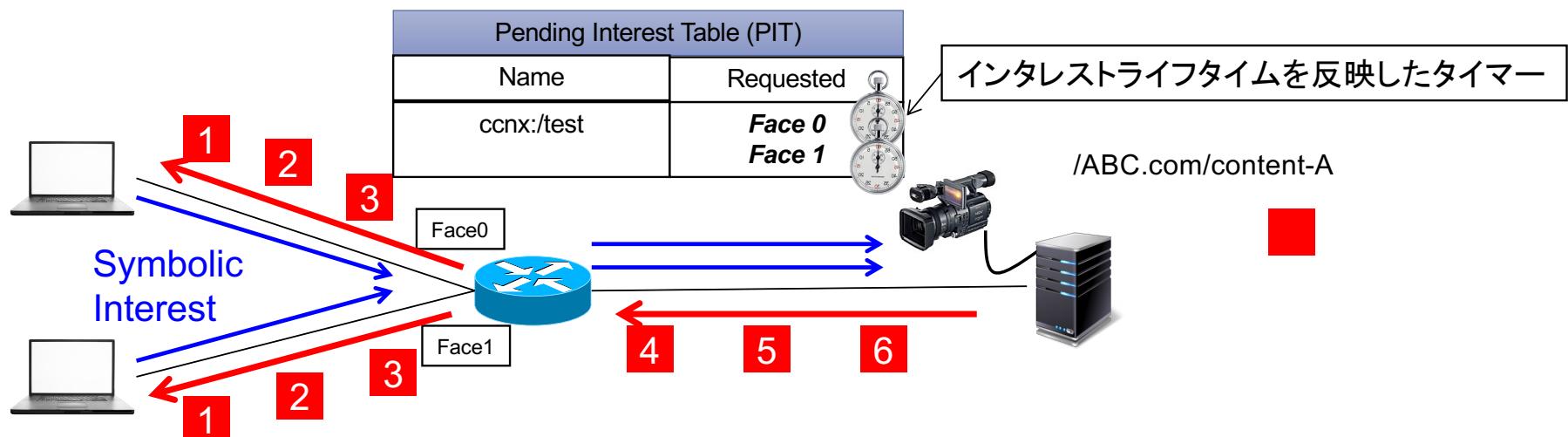


```
username:~/cefpco-master$ python3 consumer.py  
(適切にデータ受信を行えた場合、ここにinfoが表示される)
```

ターミナル2

Symbolic Interestとは？

- Regular Interest(通常のInterest):
 - ccnx:/test/chunk_num=XX (← プレフィックス名 + チャンク番号を指定)
- Symbolic Interest
 - NICT 独自拡張のInterest
 - ccnx:/text (← プレフィックス名のみを指定)
 - チャンク番号は関係なく、指定したプレフィックス名を持つ全てのデータを、指定した時間(インターレストライフタイム default=4s)受信するためのInterest
 - リアルタイムデータを継続して受信したい場合に有益





consumer_symbolic.py

Symbolic Interest
を使用を想定

- ccnx:/testというコンテンツ名のチャネル番号0を要求する
Symbolic Interestを送信し、Dataを受信・表示する

```
import cefpyco

with cefpyco.create_handle() as handle:
    while True:
        ループ[handle.send_symbolic_interest("ccnx:/test")
              info = handle.receive()
              if info.is_succeeded and (info.name == "ccnx:/test"):
                  print("Success")
                  print(info)]
```

cefnetdへの接続

"ccnx:/test"というコンテンツ名のデータを
要求するSymbolic Interestを送信

データを待ち受け、受信後、データの情報をinfoに格納

consumer_symbolic.py



publisher_symbolic.py

Symbolic Interes
を使用を想定

- ccnx:/testというコンテンツ名のデータを1秒おきに送信する。(チャンク番号は0から1ずつ増加)

publisher_symbolic.py

```
import cefpyco
from time import sleep

chunkNum = 0
with cefpyco.create_handle() as handle:
    while True:
        handle.send_data("ccnx/test", "hello", chunkNum)
        print("Send Data, ccnx:/test, chunkNum=", chunkNum, sep=',')
        chunkNum += 1
        sleep(1)
```

cefnetdへの接続



※プログラムを強制終了したい場合は、Ctrl-Cを入力



実行手順

Symbolic Interes
を使用

※ cefnetdstartを起動する前に、/usr/local/cefore/cefore.confファイルにて、
CS_MODE=0 となっているか確認

```
username:~/cefpco-master$ cefnetdstart  
username:~/cefpco-master$ python3 publisher_symbolic.py
```

ターミナル1



```
username:~/cefpco-master$ python3 consumer_symbolic.py  
(適切にデータ受信を行えた場合、ここに受信したデータinfoが逐次表示される)
```

ターミナル2



(参考)CCNxPacketInfoのプロパティ

プロパティ名	型	説明
<code>is_succeeded,</code> <code>is_failed</code>	bool	パケット受信の成否フラグ
<code>is_interest, is_data</code>	bool	受信したパケットがInterest/Dataか否かを表すフラグ (受信失敗時には両方ともFalseとなる)
<code>name</code>	string	URI形式(ccnx:/~)の名前
<code>name_len</code>	int	URI形式の名前の長さ(name TLV長ではない)
<code>chunk_num</code>	int	チャunk番号
<code>payload</code>	bytes	(Dataパケットの場合)コンテンツのデータ
<code>payload_s</code>	string	(Dataパケットの場合)コンテンツのデータ(文字列として取得、 バイナリデータの場合は無効)
<code>payload_len</code>	int	(Dataパケットの場合)コンテンツのデータのバイト長
<code>version</code>	int	受信パケットのバージョン値
<code>type</code>	int	受信パケットのタイプ値
<code>actual_data_len</code>	int	受信したパケットのヘッダを含むバイト長
<code>end_chunk_num</code>	int	コンテンツの最後のチャunk番号(指定時のみ有効)



(参考) CefpycoHandle API

CefpycoHandle メソッド名	引数・返り値 (key=valueはデフォルト値のある省略可能引数)	説明
begin	<ul style="list-style-type: none">ceforedir=None: cefnetd.confの入ったディレクトリパスportnum=9896: cefnetd のポート番号	cefnetdへの接続を開始する。with構文を利用する場合は自動で呼ばれる。
end	無し	cefnetdへの接続を修了する。with構文を利用する場合は自動で呼ばれる。
send_interest	<ul style="list-style-type: none">name: コンテンツ名 (ccnx:/...)chunk_num=0: チャンク番号(負の数の場合はチャンク番号無し)symbolic_f=INTEREST_TYPE_REGULAR: Interestのタイプを指定hop_limit=32: ホップ数lifetime=4000: Interest ライフタイム(現在の時刻からのミリ秒)	指定した名前のコンテンツを要求するInterestパケットを生成して送信する。
send_data	<ul style="list-style-type: none">name: コンテンツ名 (ccnx:/...)payload: Dataパケットのペイロードchunk_num=-1: チャンク番号(負の数の場合はチャンク番号無し)end_chunk_num=-1: コンテンツの最後のチャンク番号(負の数の場合は省略)hop_limit=32: 最大ホップ数expiry=36000000: コンテンツ期限(現在の時刻からのミリ秒)cache_time=-1: 推奨キャッシュ時間(負の数の場合は省略)	指定した名前とペイロードに基づいてDataパケットを生成して送信する。
receive	<ul style="list-style-type: none">error_on_timeout=false: タイムアウト時にエラーを投げるか否かtimeout_ms=4000: 受信開始からタイムアウトまでの時間(ミリ秒)返り値: CcnPacketInfo (別スライド参照)	InterestまたはDataパケットを指定した時間だけ待ち受け(デフォルト4秒)、受信パケットの情報を返す。
register	<ul style="list-style-type: none">name: 受信したいInterestのプレフィックス名を指定	受信したい Interest のプレフィックス名を cefnetd に登録し、receive で Interest を受け取れるようにする。
deregister	<ul style="list-style-type: none">name: 登録を解除したい名前を指定	register で登録した名前を解除する。
send_symbolic_interest	chunk_numとsymbolic_fが無い以外はsend_interestと同様	通常の Interest の代わりに Symbolic Interest(Cefore 独自機能)を送信する。

- ICN の概説
- Cefore の紹介
- Practice
 - Cefore の基本操作
 - ネットワーク内キャッシュの効果を体感
 - ラズパイを使ったビデオ配信
 - Cefpryco によるアプリ開発
- 今後のプラン
 - チュートリアル・ハンズオンの継続開催
 - 新たなアプリケーションで発展的なシナリオ(ロボットなど)での実施



CQ 出版 Interface 2024年2月号



第1部 情報指向ネットワーク技術「ICN」の概要

第1章 わざわざ遠くのサーバにアクセスするの効率が悪い…同じデータを持つ近くの機器から取得すればよいのでは?

アドレスではなく情報名で探す! 情報指向ネットワーク技術「ICN」

朝枝 仁

第2部 コード改変なしに機能追加やカスタマイズできるから ラズベリー・パイでもハイエンド・サーバでも使える ICN通信ソフトウェア基盤 「Cefore」の基礎知識

速水 祐作, 大岡 雄, 松浦 和久, 朝枝 仁

Cédoisは、国際研究開発促進会議(NICI)が実現した日本初のICN互通ツールです。PIT(Pending Interest Task Force)RFC 8691と、RFC 8690で定義されたICNx-1.0のパケット・フォーマットに準拠しています。アグリゲーション・ツリー上にデータを蓄積するため、データを複数のノードに配布する機能が実装されています。Ubunt 20.04.22版とRasperry Pi OS(bulseye)上で動作します。TCP/IP用のIPv4オーバーIPv6もサポートされています。

ICNの通信を行います。

Ceforeの設計コンセプト

● 促使ルートの選択、情報を追加・削除して

私たちはが利用するはとどの
ビスでは、www.example.e.co
用いて選択を行います。

● ルートの選択、情報を追加・削除して

私たちはが利用するはとどの
ビスでは、www.example.e.co
用いて選択を行います。

● ルートの選択、情報を追加・削除して

私たちはが利用するはとどの
ビスでは、www.example.e.co
用いて選択を行います。

● Ceforeの特徴

- 日本語の ICN 解説本
- 「はじめての ICN」に、是非購入ご検討ください！
 - <https://interface.cqpub.co.jp/magazine/202402/>

第4部 複数ユーザーに同時に一斉配信! ICNのマルチキャスト通信機能を試す 実験②…カメラ映像の ライブ配信&受信

速水 祐作, 大岡 雄



図解1 マルチキャスト動画の構成

ラズベリー・パイによって撮影された動画をUbuntu(左側)に接続したPC(右側)にて再生される。Ubuntuの表示マシン上でコンソール上で実行したgiggle.pyとls命令の結果。

本章では、ラズベリー・パイを用いてより実践的な

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

ラズベリー・パイ用カメラ: キジニア Raspberry Pi Cameraで撮影した動画をリアルタイム・エンコードするツール: エンコーダー: ディミングで、別途端末に対してICNパケット

(Interest/Content Object)を用いてライブ配信するツール: リアルタイム・エンコードするツール: ゲンシヤー: リアルタイム・エンコードするツール: リアル

タイム・エンコードするツール: リアル

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

動画をリアルタイム・配信用のエンコーディングで

あるgiggle.pyを用いる

・giggle.py

エンコードしたデータをICN通信用のContent

Objectに変換し、固定レートで送信するための

ツール: 送信用のQRコード

QRコード: ラズベリー・パイを用いて実現する

ICNアーキテクチャを構成します。

● 実験に使うツールやコマンド

使用するツール: コマンドは次の通りです。

・libcamera-vid

カメラ: ラズベリー・パイで動画を撮影するためのコマ

マンド

・ffmpeg

撮影した動画をFTP用にエンコードするコ

マンド: 実験ではlibcamera-v4l2で撮影した

- サーベイ・チュートリアル論文
 - ICN の研究動向
 - Cefore チュートリアル
 - 実験・性能評価
- URL
 - https://search.ieice.org/bin/summary.php?id=e107-b_1_139



IEICE TRANS. COMMUN., VOL.E107-B, NO.1 JANUARY 2024

139

INVITED PAPER

A Survey of Information-Centric Networking: The Quest for Innovation

Hitoshi ASAEDA^{†a)}, Senior Member, Kazuhisa MATSUZONO[†], Yusaku HAYAMIZU[†], Htet Htet HLAING[†], and Atsushi OOKA[†], Members

SUMMARY Information-Centric Networking (ICN) is an innovative technology that provides low-loss, low-latency, high-throughput, and high-reliability communications for diversified and advanced services and applications. In this article, we present a technical survey of ICN functionalities such as in-network caching, routing, transport, and security mechanisms, as well as recent research findings. We focus on CCNx, which is a prominent ICN protocol whose message types are defined by the Internet Research Task Force. To facilitate the development of functional code and encourage application deployment, we introduce an open-source software platform called Cefore that facilitates CCNx-based communications. Cefore consists of networking components such as packet forwarding and in-network caching daemons, and it provides APIs and a Python wrapper program that enables users to easily develop CCNx applications on Cefore. We introduce a Mininet-based Cefore emulator and lightweight Docker containers for running CCNx experiments on Cefore. In addition to exploring ICN features and implementations, we also consider promising research directions for further innovation.

key words: *Information-Centric Networking (ICN), Content-Centric Networking (CCN), CCNx, Cefore*

1. Introduction

Data are of critical importance for communications, and there is a need for advanced network architectures that can

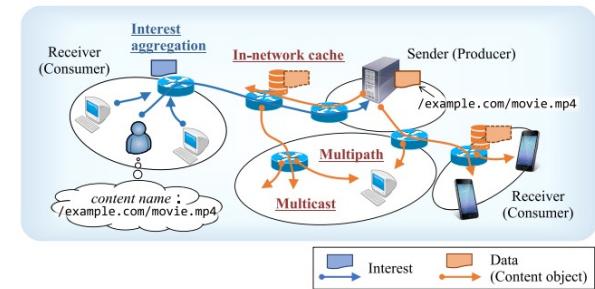


Fig. 1 ICN concept.

load estimation mismatches between users and servers.

Edge computing, in which data are processed and information is provided by servers located near end users, has also attracted significant attention. Edge computing reduces latency and improves the quality of communication services. It is expected that the fusion of cloud and edge computing will enhance future Internet services. However, because such systems are still scarce, dependent, it is difficult to solve

Thank you.

Cefore 



<https://cefore.net/>

GitHub 



<https://github.com/cef>