National Institute of
Information and Communications Technology

Cefore を用いた情報指向ネットワークの実践
～ ネットワーク構築からアプリケーション応用まで ～

速水祐作　　松園和久　　朝枝仁

国立研究開発法人情報通信研究機構（NICT）

2024年9月13日

2024年電子情報通信学会ソサイエティ大会企画セッション（情報指向ネットワーク技術特別研究専門委員会）

謝辞 本研究の一部は，JSPS科研費 24K20768 により得られたものである.

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
- 基礎編: 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
- 基礎編: 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 ライブラリ
  - Cefpyco を用いた基本通信
- 応用編・実践編: 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

National Institute of
Information and Communications Technology

事前準備: Ubuntu / RaspberryPi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 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 をダウンロードしてから、GUI化のため sudo apt install ubuntu-desktop する

NOTE: Ubuntu 20.04/22.04 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。

Ubuntu 設定 -- Ubuntu 20.04 or 22.04

- 必要なライブラリのダウンロード

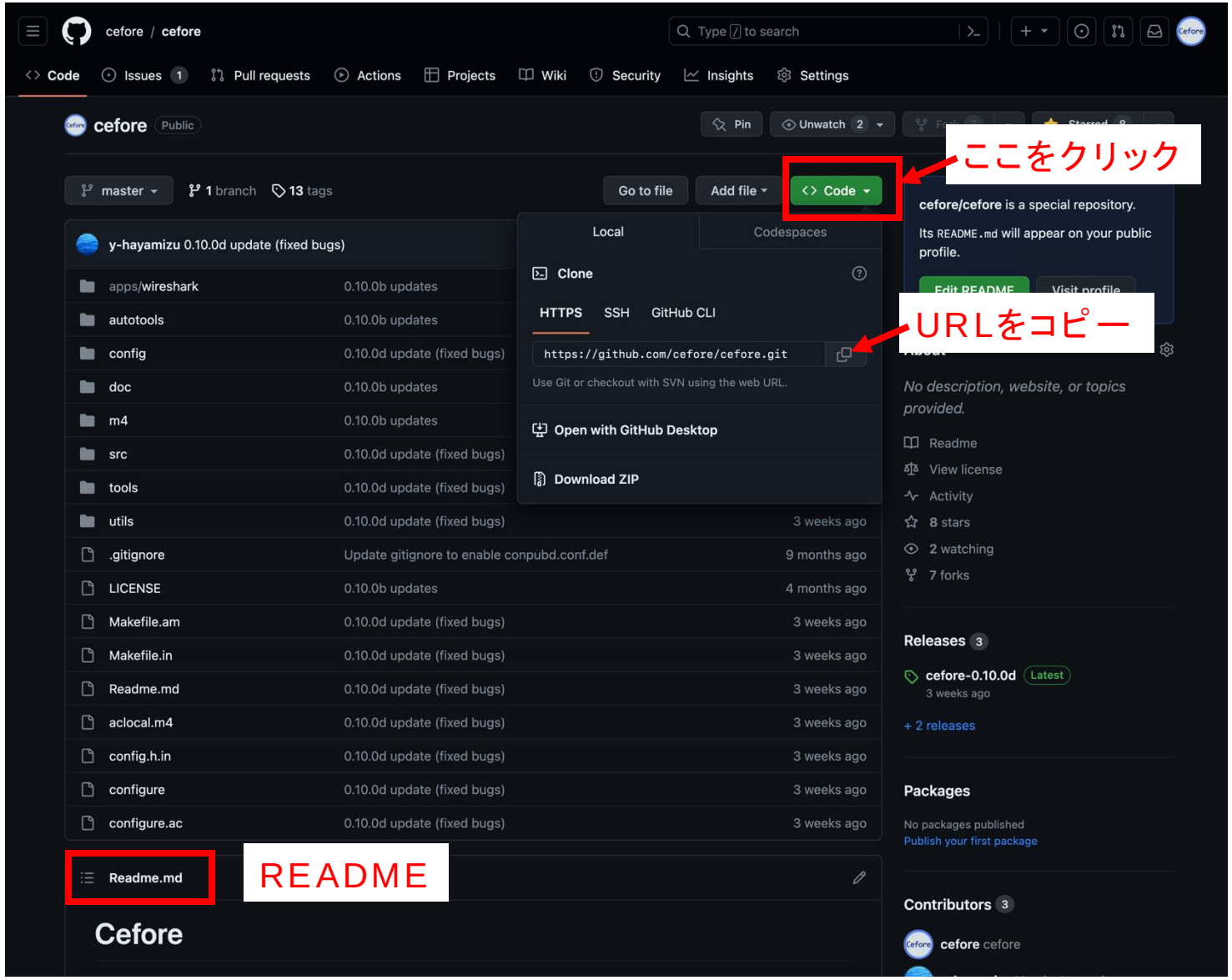
```
cefore:~$ sudo apt update -y
cefore:~$ sudo apt upgrade -y
cefore:~$ sudo apt install git gcc make ffmpeg net-tools emacs vim nano
cefore:~$ sudo apt update
```

- 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]



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

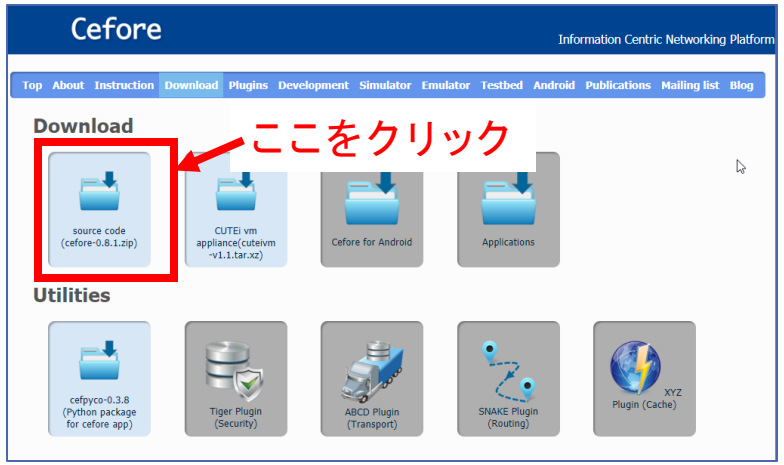
```
cefore:~$ git clone https://github.com/cefore/cefore.git
```

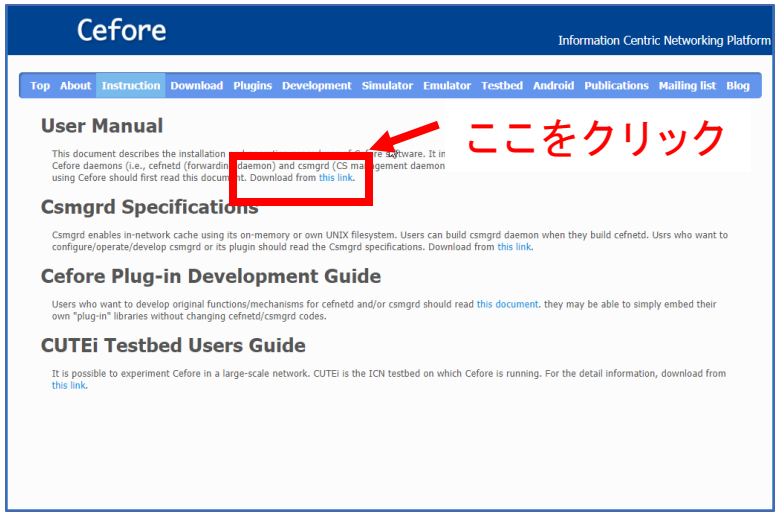
https://github.com/cefore/cefore

Ubuntu 設定 -- cefore.net からダウンロード [オプション]

https://cefore.net/から
ソースコードとユーザマニュアルをダウンロード

- ソースコード
  - Download
  > source code
  (cefore-0.10.0g.zip)
- ユーザマニュアル
  - Instruction
  > User Manual





Ubuntu 設定 -- Cefore インストール

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

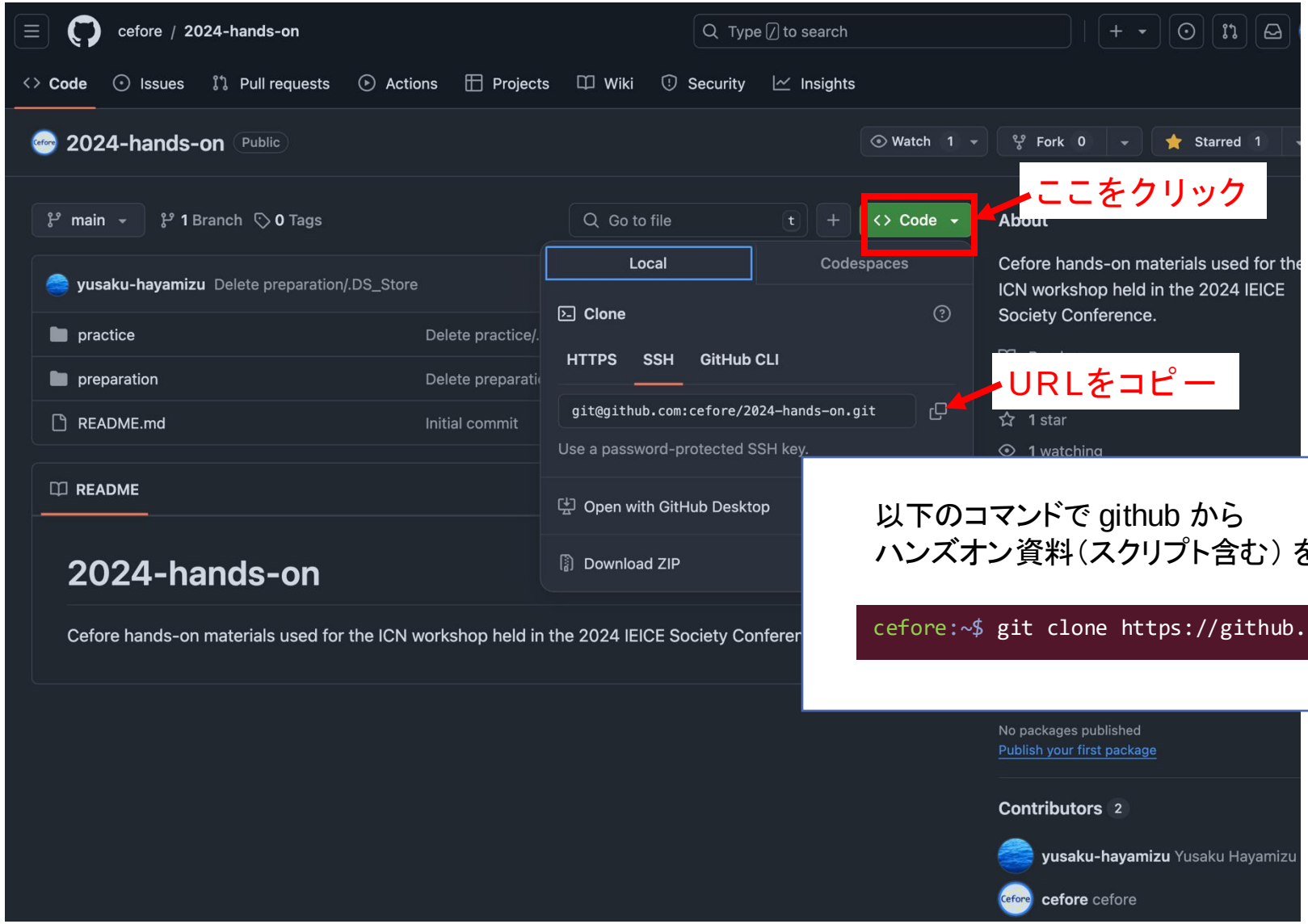
```
$ sudo apt-get install libssl-dev automake
```

- 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



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
```

※ 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 への接続

- Wi-Fi AP の情報

```
SSID:  Cefore-AP2-5G-WPA3
Pass:  icn-ken2024
IP:    192.168.102.0/24 (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
  - ssh icntest2023@pi1.local
    - user = icntest2023
    - password = **icn-ken2024**

```
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
```

[参考] Raspberry Pi 設定 -- 基本設定

- micro SDカード挿入
  - Rasberry Pi OS (bullseye) が既に install 済み（Raspberry Pi Imagerを利用）
    - https://www.raspberrypi.com/software/
- カメラモジュール接続 & テスト

```
$ libcamera-vid
```

- Internet 接続
  - Country 設定 = JP
  - Time Zone 設定（時刻が大きくずれていないか要確認）
- ライブラリのダウンロード

```
$ sudo apt install git gcc ffmpeg
```

- Cefore のインストール
  - Ubuntu と同様
- 接続確認
  - ping/SSH test

[参考] Raspi 設定 -- Cefore インストール

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

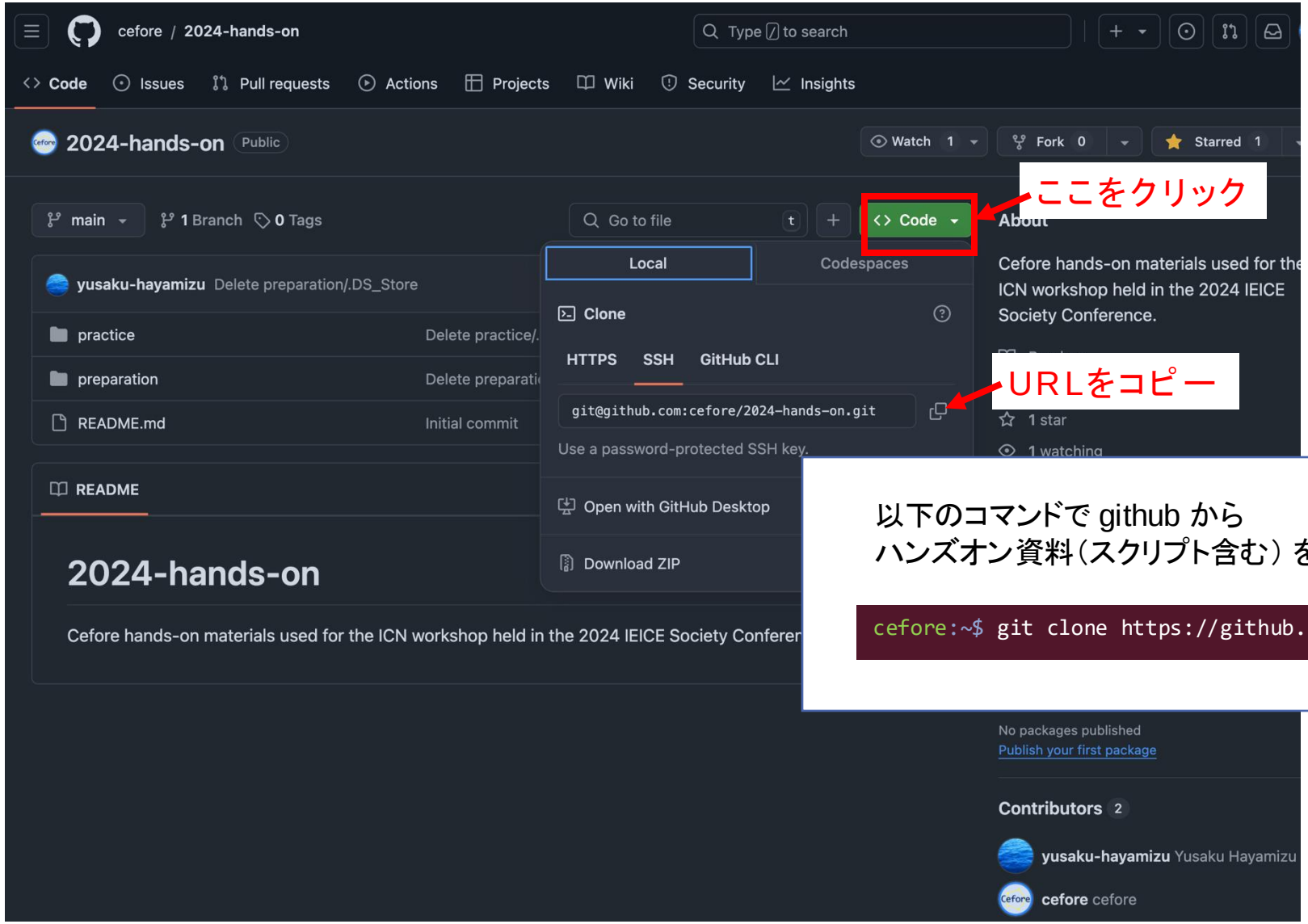
```
$ sudo apt-get install libssl-dev automake
```

- Installing Cefore

```
$ cd cefore-0.10.0b
$ 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

[参考] Raspberry Pi設定 -- Cefore ハンズオン資料のダウンロード



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
```

[参考] Raspberry Pi 設定 -- WiFi AP化

参考: https://www.mikan-tech.net/entry/raspi-wifi-ap

- 各 team のID に基づいて、hostapd.conf の piX の X を変更
  - 例 team IDが1 の場合

```
…

ssid=piX -> pi1
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=icn-ken2024
macaddr_acl=0
ignore_broadcast_ssid=0
```

- ./preparation/enable-wifi-ap.bash を実行

```
$ ifconfig
…
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.0.1  netmask 255.255.255.0  broadcast 10.0.0.255
      inet6 fe80::b4df:d61e:2dc9:cfb9  prefixlen 64  scopeid 0x20<link>
      ether dc:a6:32:bf:69:13  txqueuelen 1000  (Ethernet)
      RX packets 3819  bytes 353690 (345.4 KiB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 1653  bytes 224167 (218.9 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- 動作確認
  - Raspi OS で ifconfig を使って wifi AP のアドレスが設定できているか
  - WiFi で SSID = pi1のネットワークが表示されているか
  - raspi – Ubuntu 間のIP接続確認 (ping 10.0.0.1)
  - ssh cefore@10.0.0.1（password は **icn-ken2024** ）

Ubuntu が VMの場合、まずホストOS（macOS/Windows）で上記 pi1 に WiFi 接続した後、VMをブリッジモードのネットワーク設定で起動

National Institute of
Information and Communications Technology

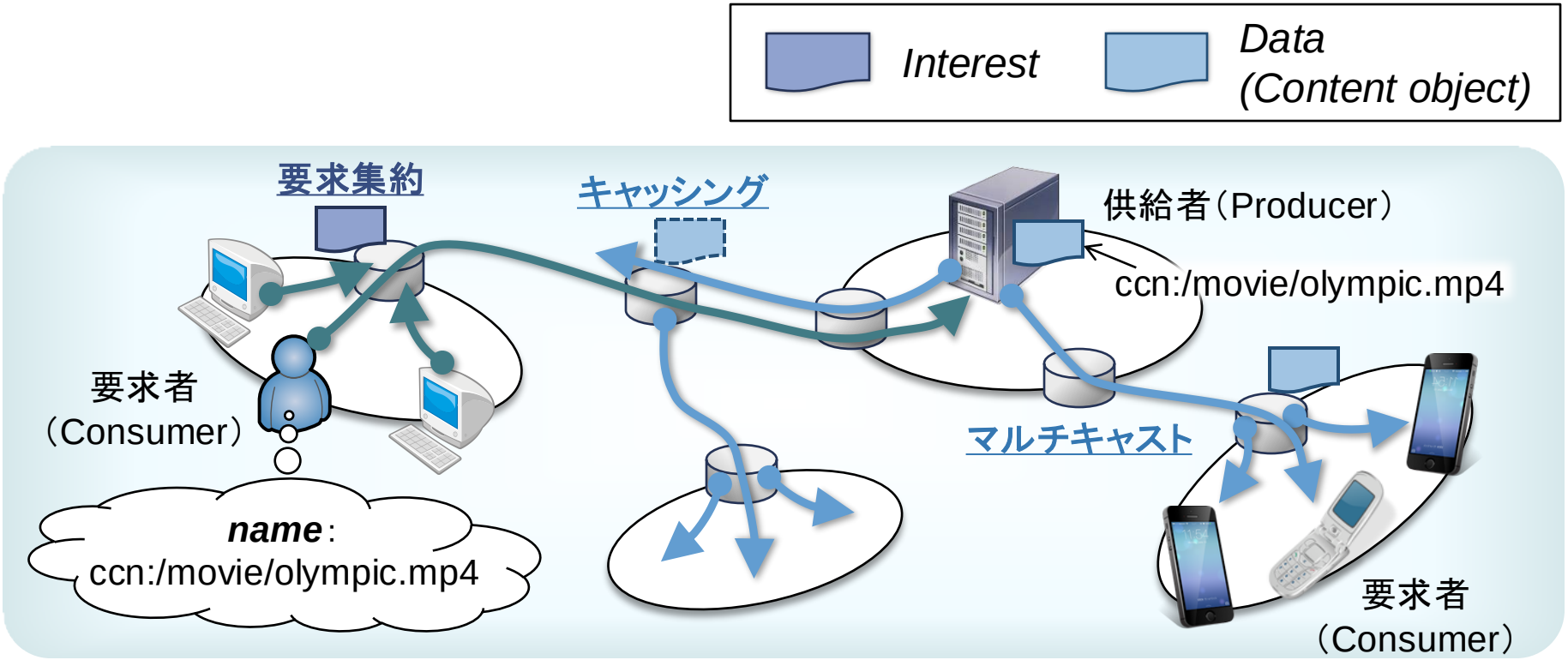
基礎編: ICN / Cefore の概要と基本的な通信

National Institute of
Information and Communications Technology

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)

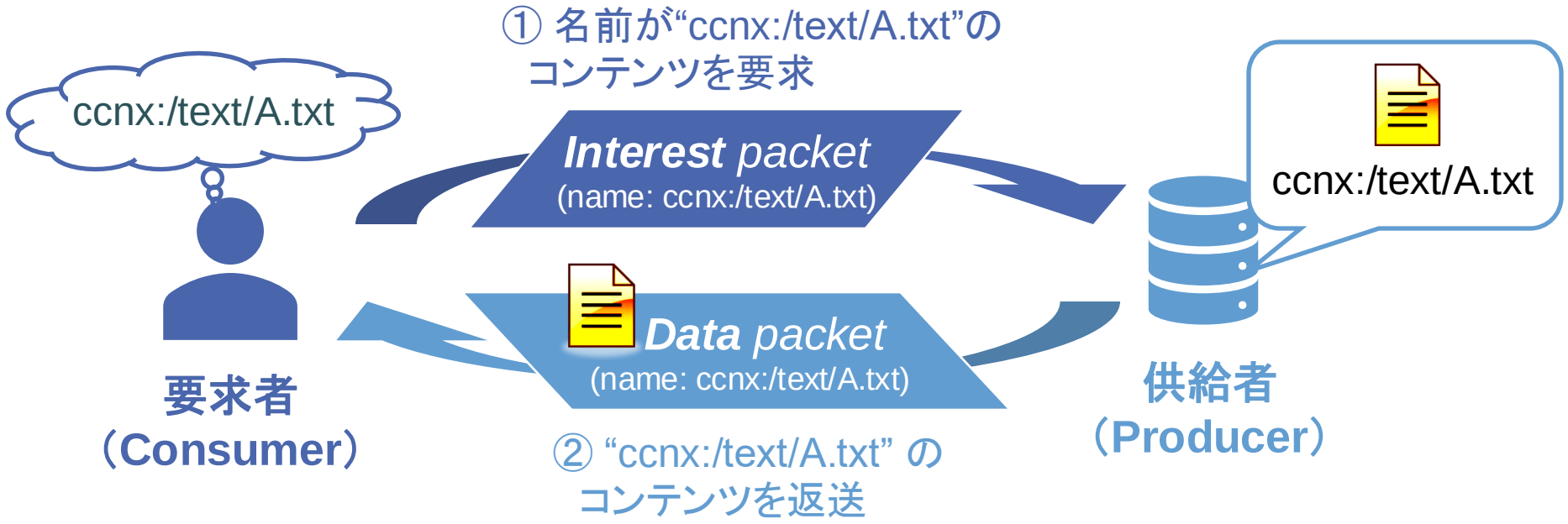
- ホスト中心ではなくコンテンツ中心のネットワークアーキテクチャ
  - IP アドレスではなくコンテンツ名を使用
- コンテンツを効率的に配布・取得するための仕組みをサポート



ICN の概略図

コンテンツ名に基づく通信

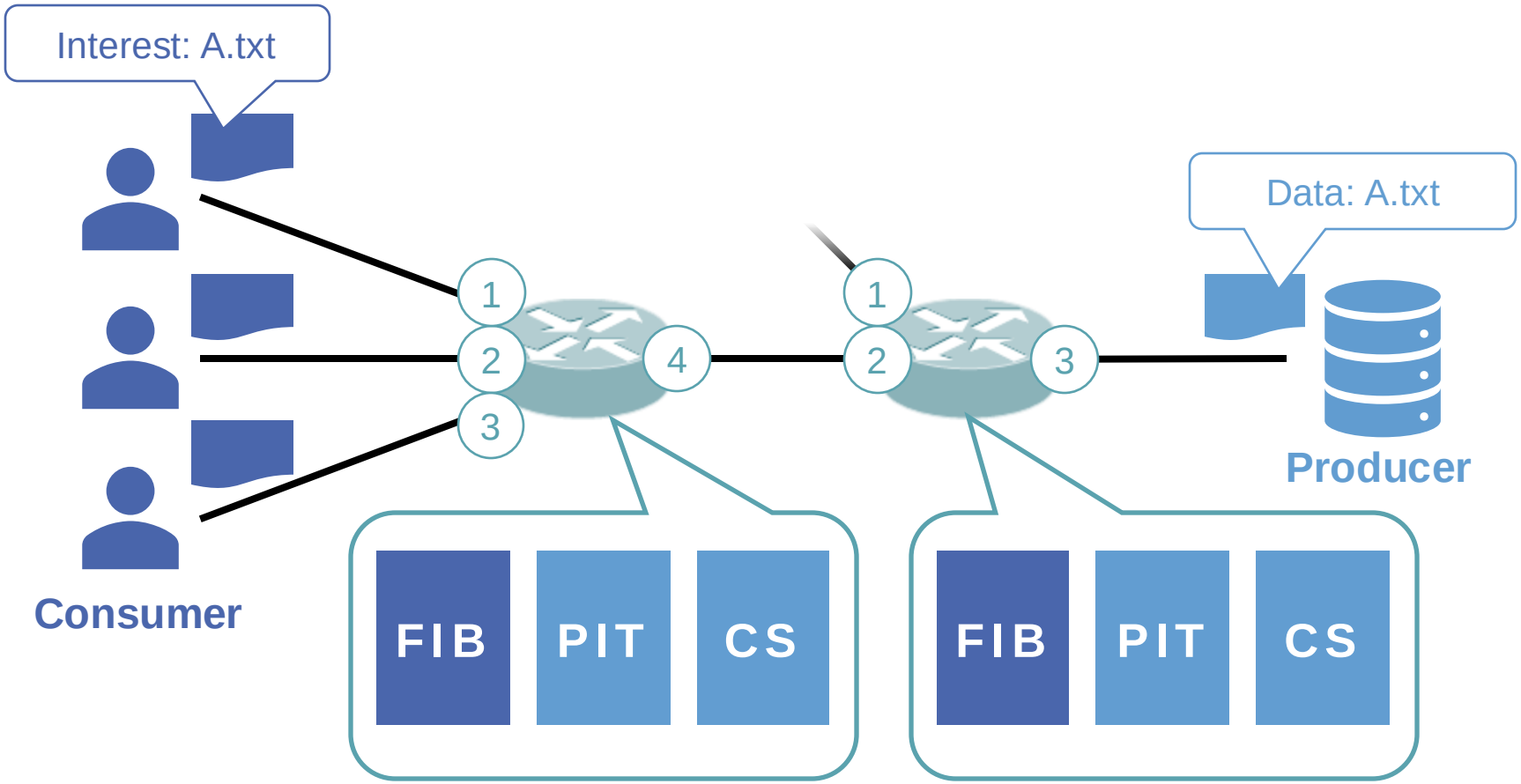
- **消費者（Consumer）と供給者（Producer）で通信**
  - IP アドレスのような場所（通信相手）の情報は不要
- **2種類のパケットを用いて通信**
  - **Interest**: コンテンツを要求するためのパケット
  - **Data/Content Object**: コンテンツを返送するためのパケット



CCN の通信例

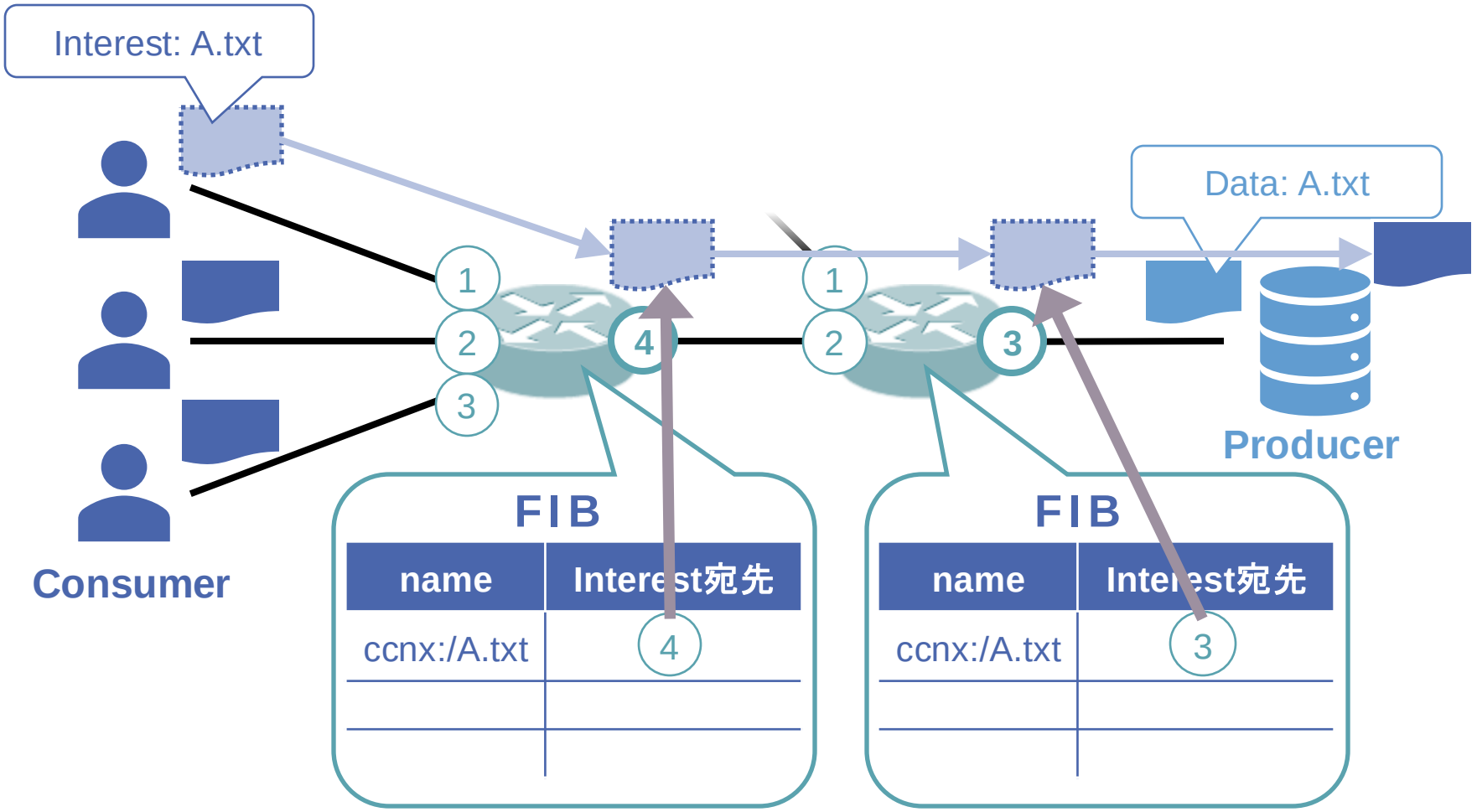
- CCN ルータはどのように通信を効率化するのか？

→例：3人の Consumer が ccnx:/A.txt を要求



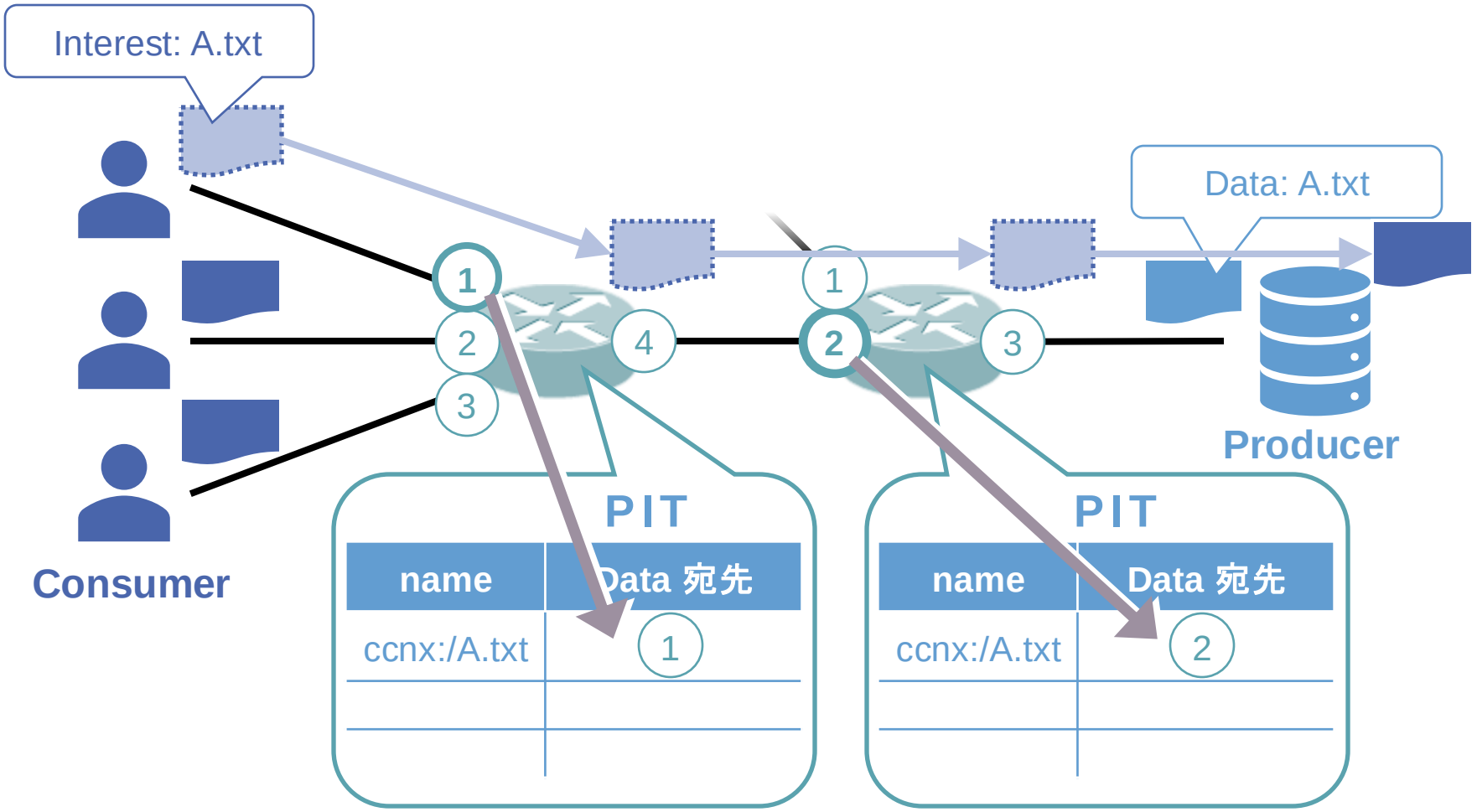
（1）Interest 転送

- 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)** に従って Interest パケットを転送する



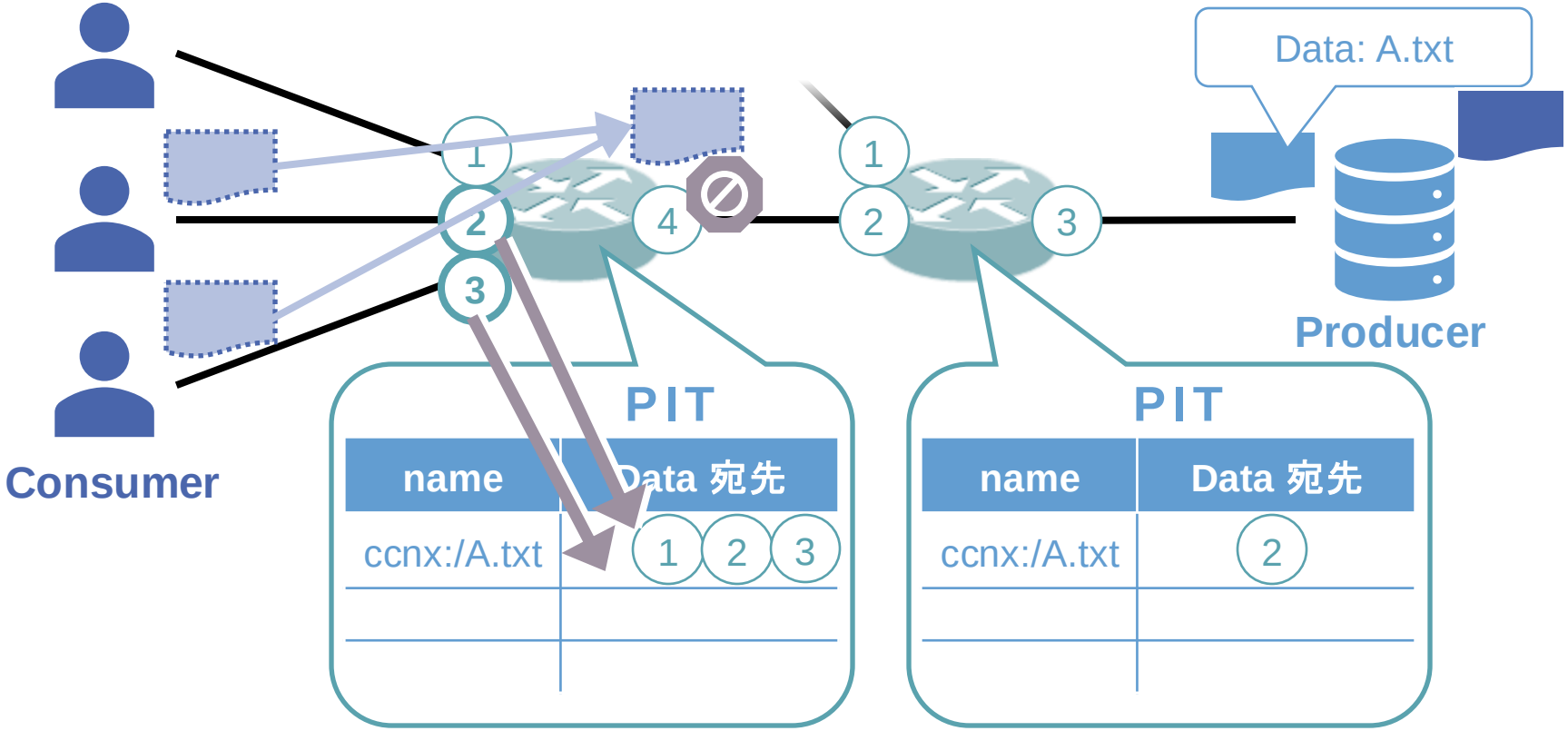
（2）要求待ち Interest の記憶

- 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



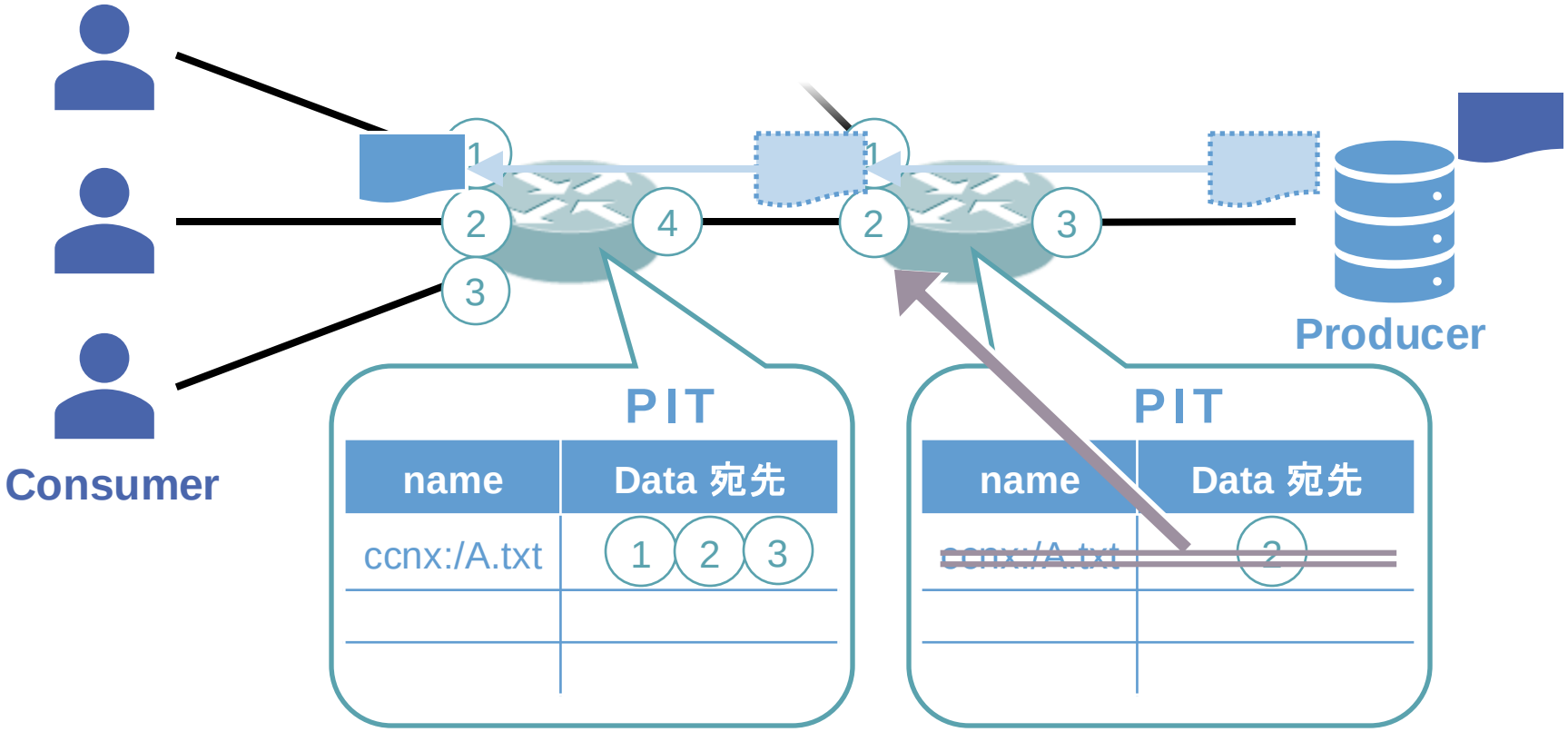
（3）重複 Interest の集約

- 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



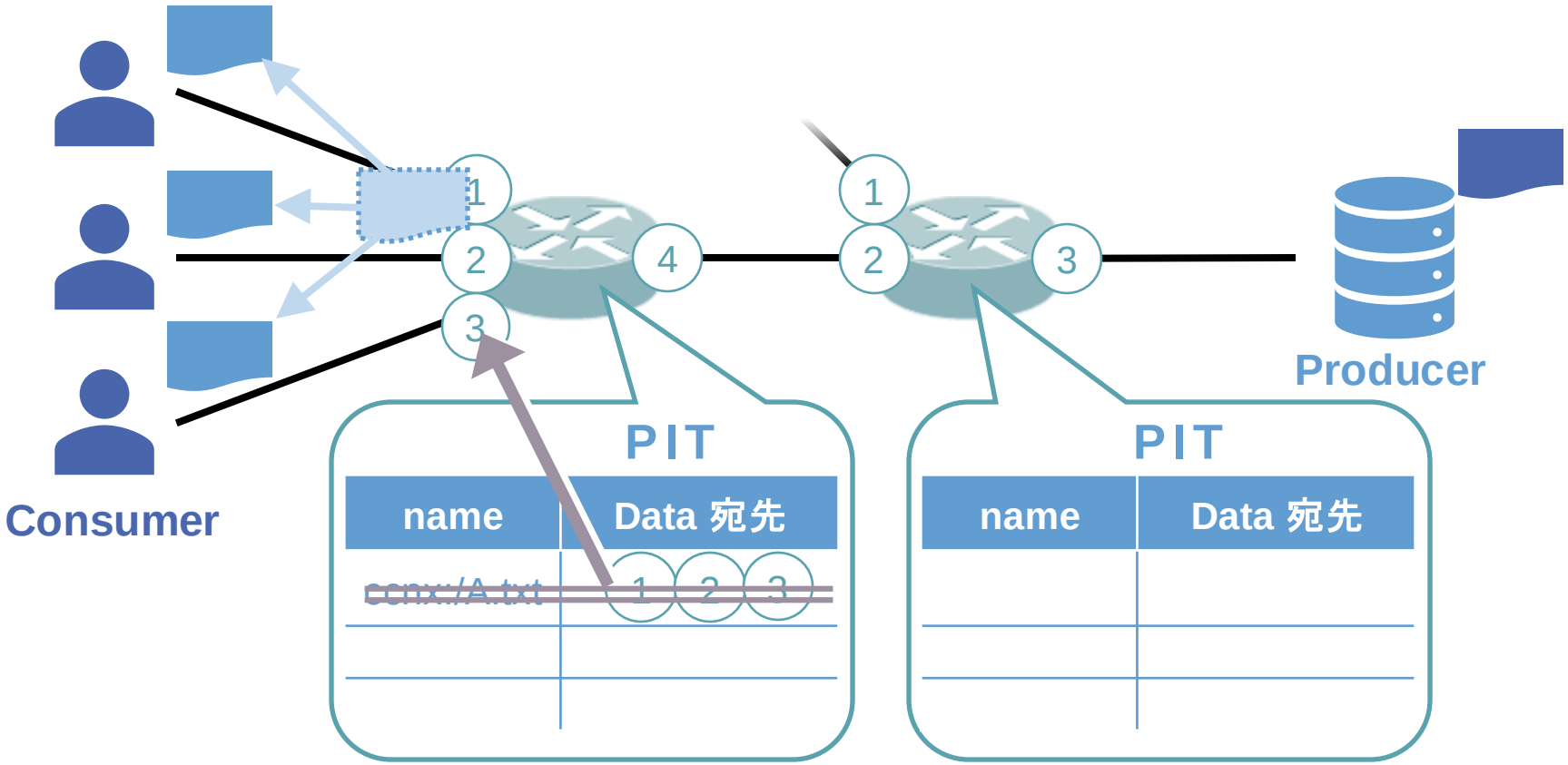
（4）Data パケットの返送

- 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



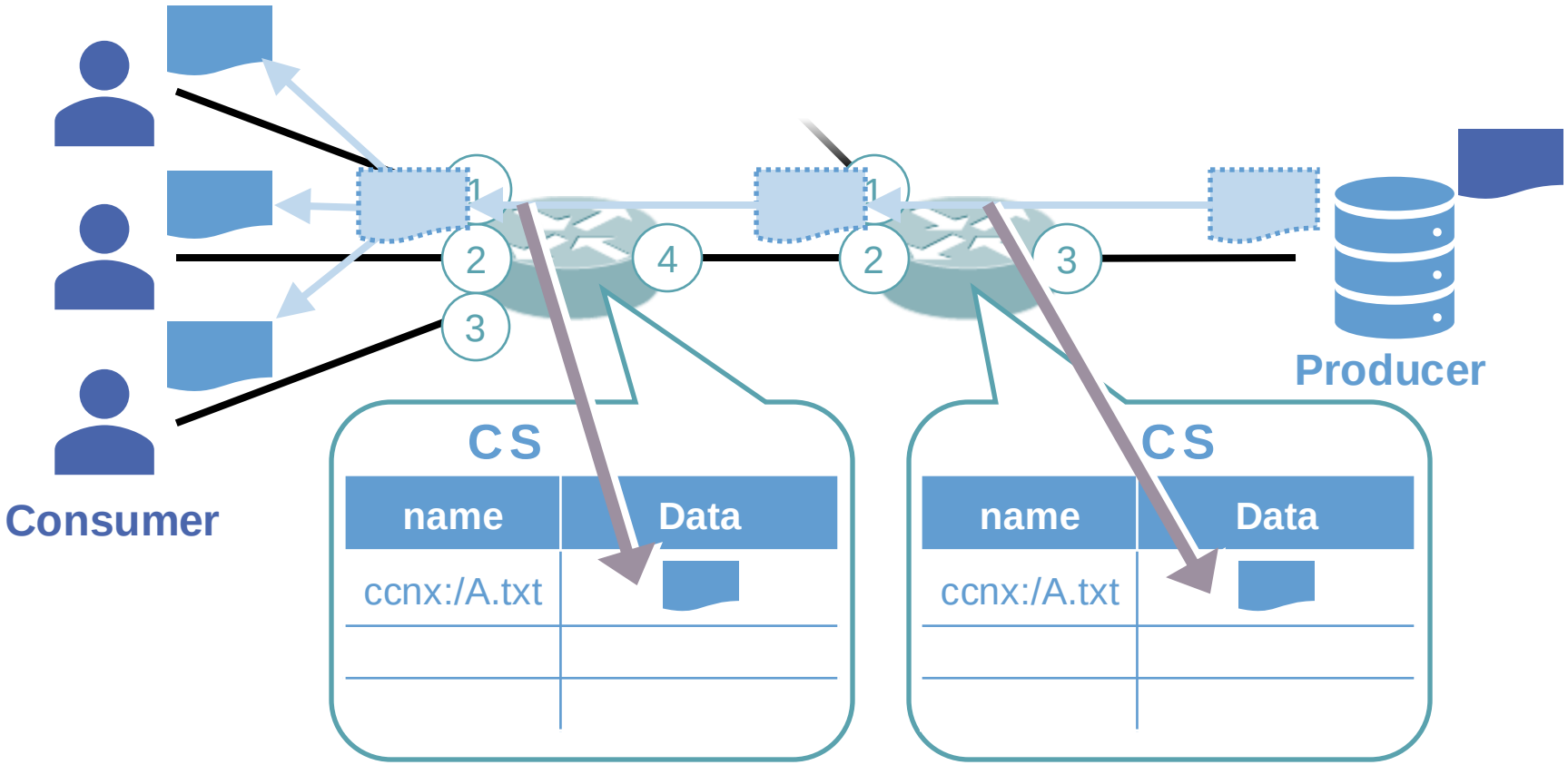
（5）Data パケットのマルチキャスト

- 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**



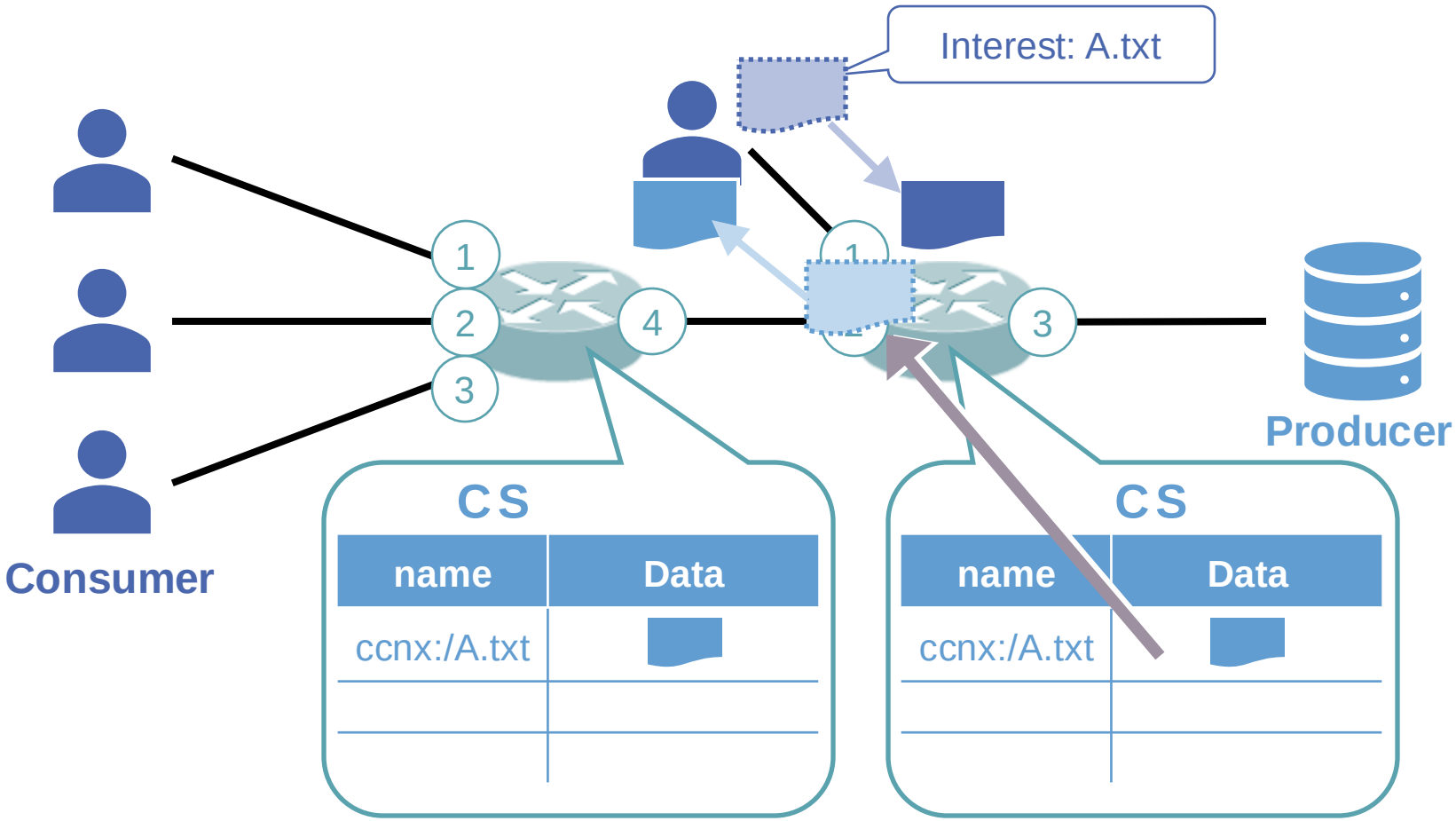
（6）Data パケットのキャッシュ

- 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)** にキャッシュする



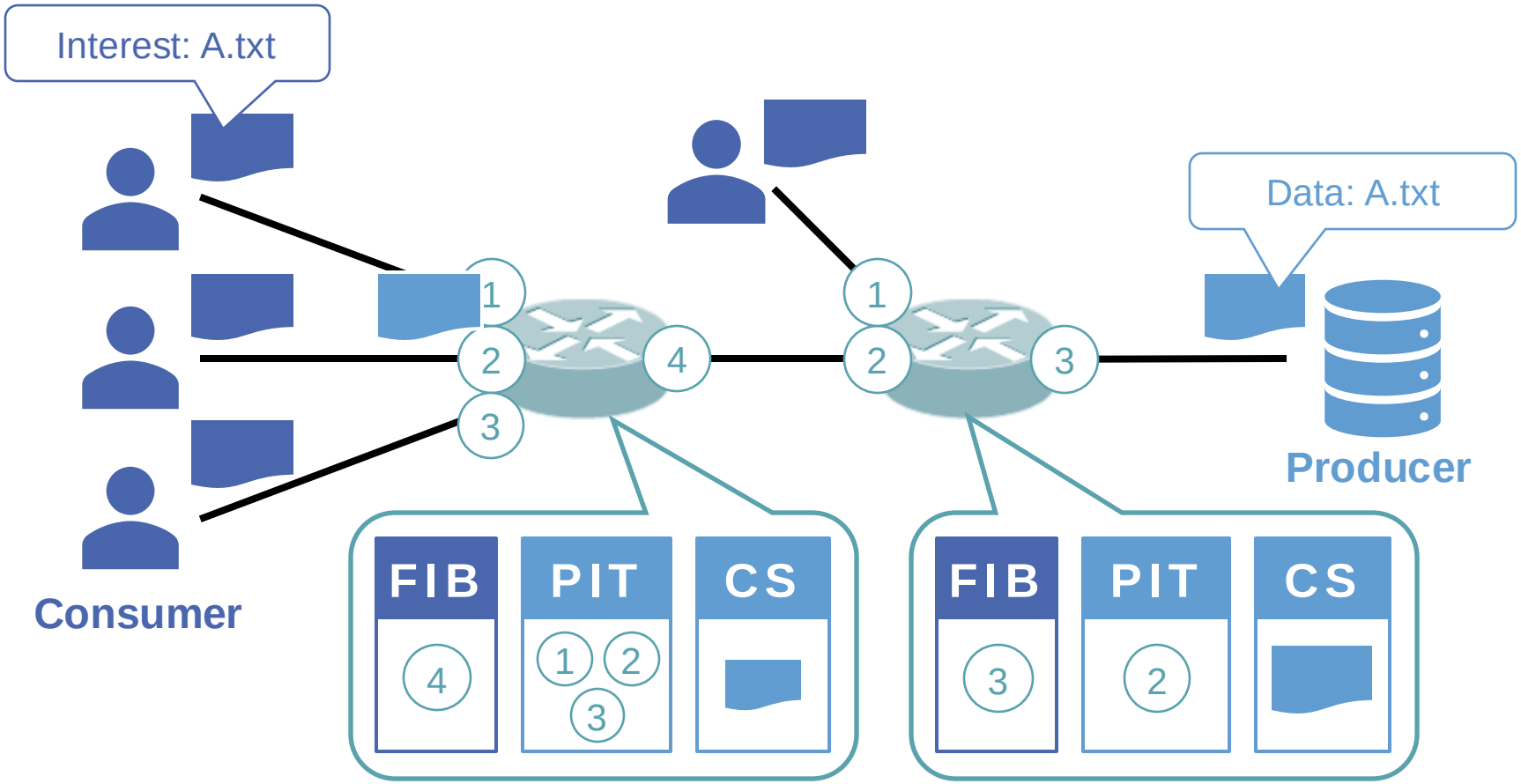
（7）ルータによるキャッシュ応答

- 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かる**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



National Institute of
Information and Communications Technology

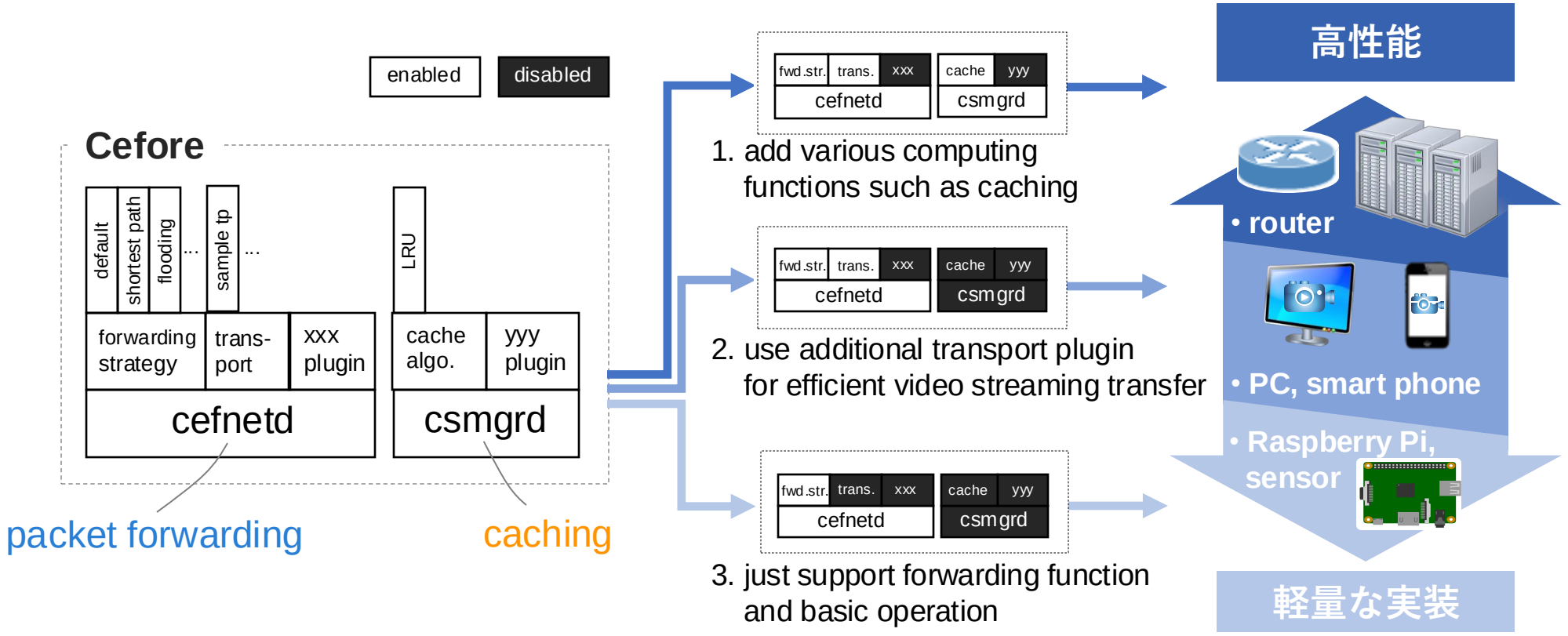
ICN のオープンソース実装
Cefore（セフォール）

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 が規定する標準化文書 (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）
  - 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

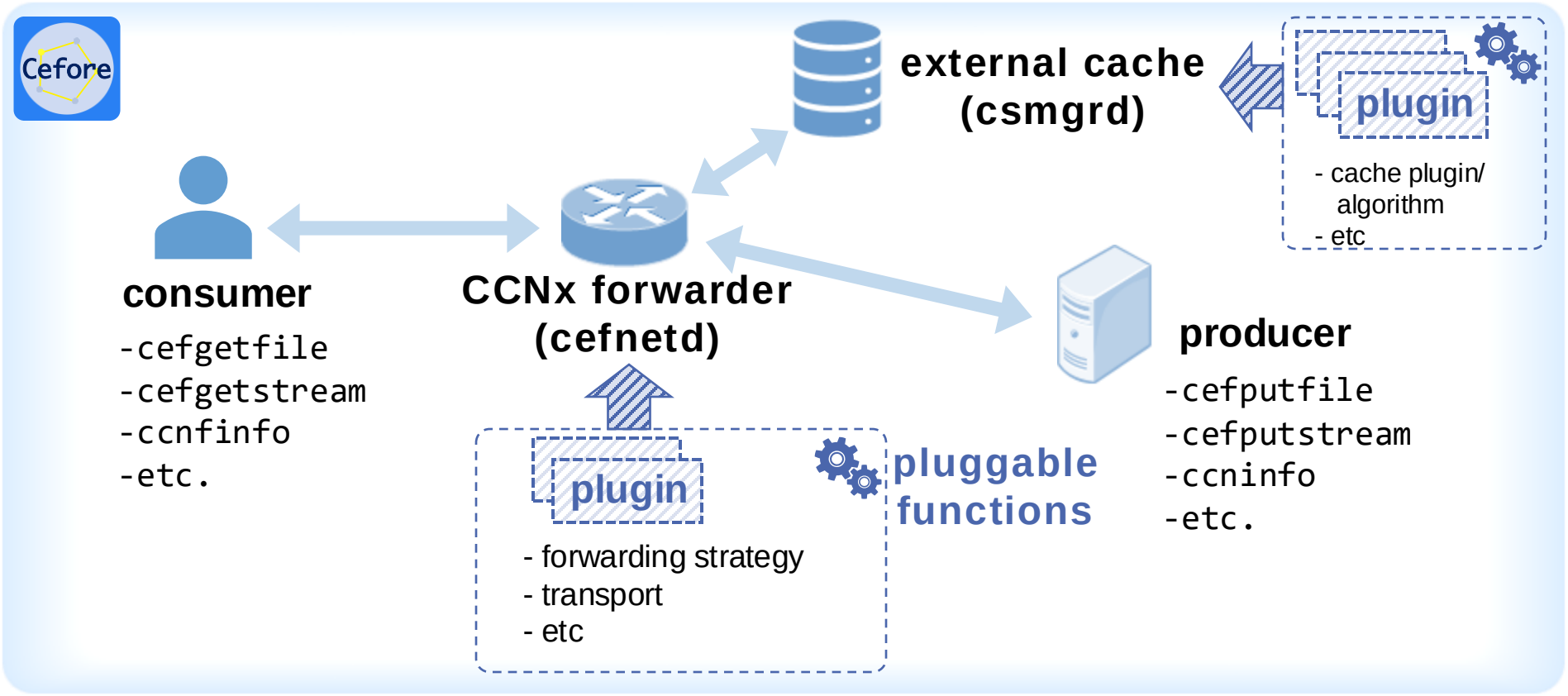
- 軽量かつ汎用的な CCN ソフトウェア実装



- 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



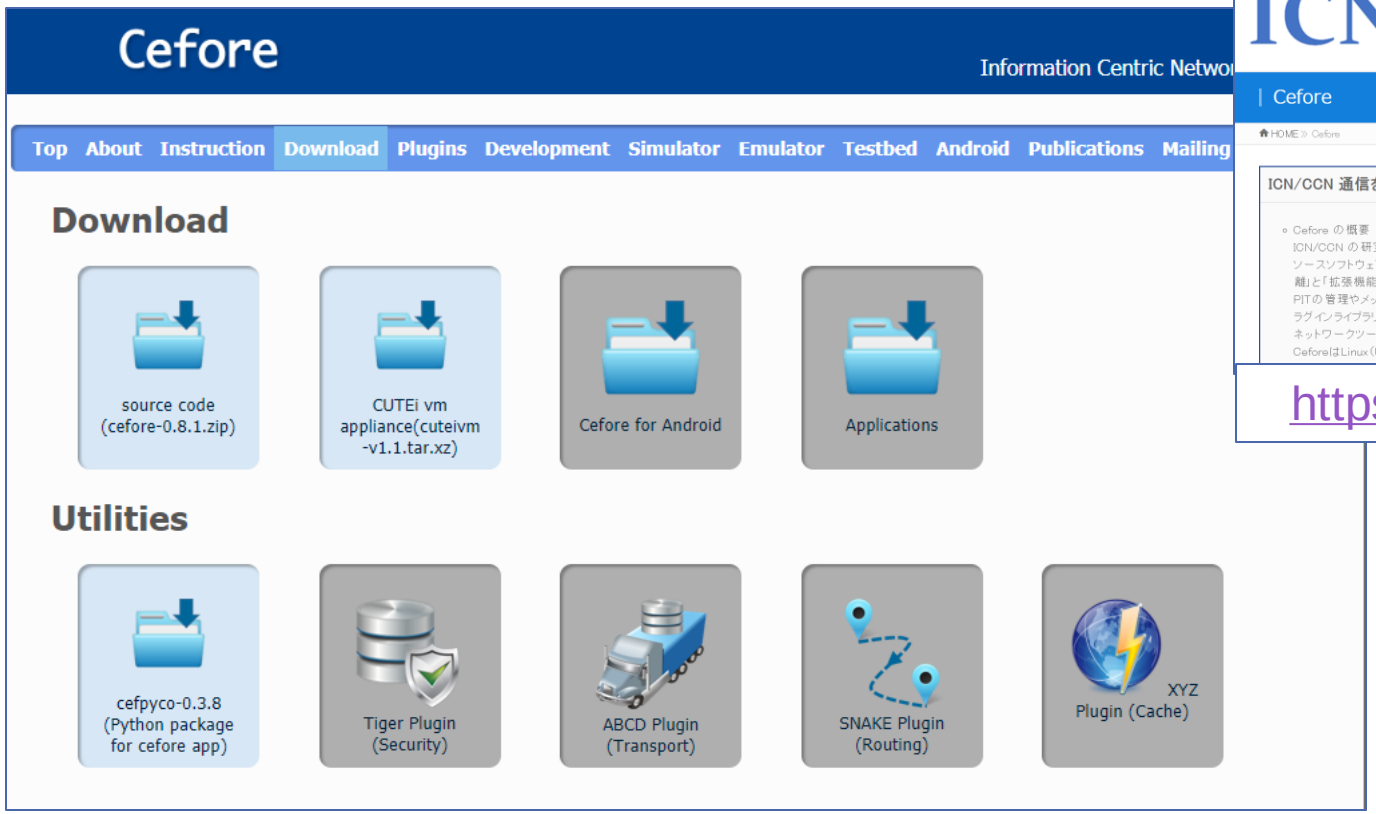
Ceforeの仕様

- 開発言語：C言語
- OS
  - Linux（ubuntu 20.04 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

- 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



https://www.ieice.org/~icn/cefore

https://cefore.net/

Cefore

https://cefore.net/

GitHub

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

C言語版

```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("ccn:/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行→4行

Python版

```python
import cefpyco

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

National Institute of
Information and Communications Technology

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

practice

Cefore デーモン (cefnetd/csmgrd) の起動・停止 ※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

- 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)

- `cefroute`
  - Insertion
    - `cefroute add ccnx:/aaa udp pi3.local`
  - Deletion
    - `cefroute del ccnx:/aaa udp pi3.local`
- Alternative: preparing an FIB configuration file
  - `/usr/local/cefore/cefnetd.fib`
    - cefnetd automatically loads this file when starting its process

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

- Routing Protocol
  - TBA

+ 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 (csgmrd)
    - 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

- 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 pi3.local
```

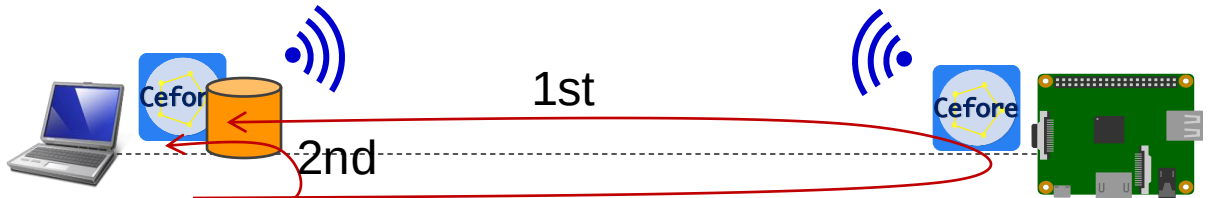
  - データ要求

```
$ 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、キャッシュによる効果を体感
- 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
$ cefroute add ccnx:/hello udp pi3.local
$ cefstatus
$ cefgetfile ccnx:/hello/100KB-file -s 1 -f ./output-1st.txt
  # Consumer のローカルキャッシュにデータが保存される
$ cefgetfile ccnx:/hello/100KB-file -s 1 -f ./output-2nd.txt
```



consumer
(cefnetd w/ localcache)

producer
(cefnted, csmgrd)

```
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
```

National Institute of
Information and Communications Technology

基礎編: 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 COmpact 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

C言語版

```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行→4行

Python版

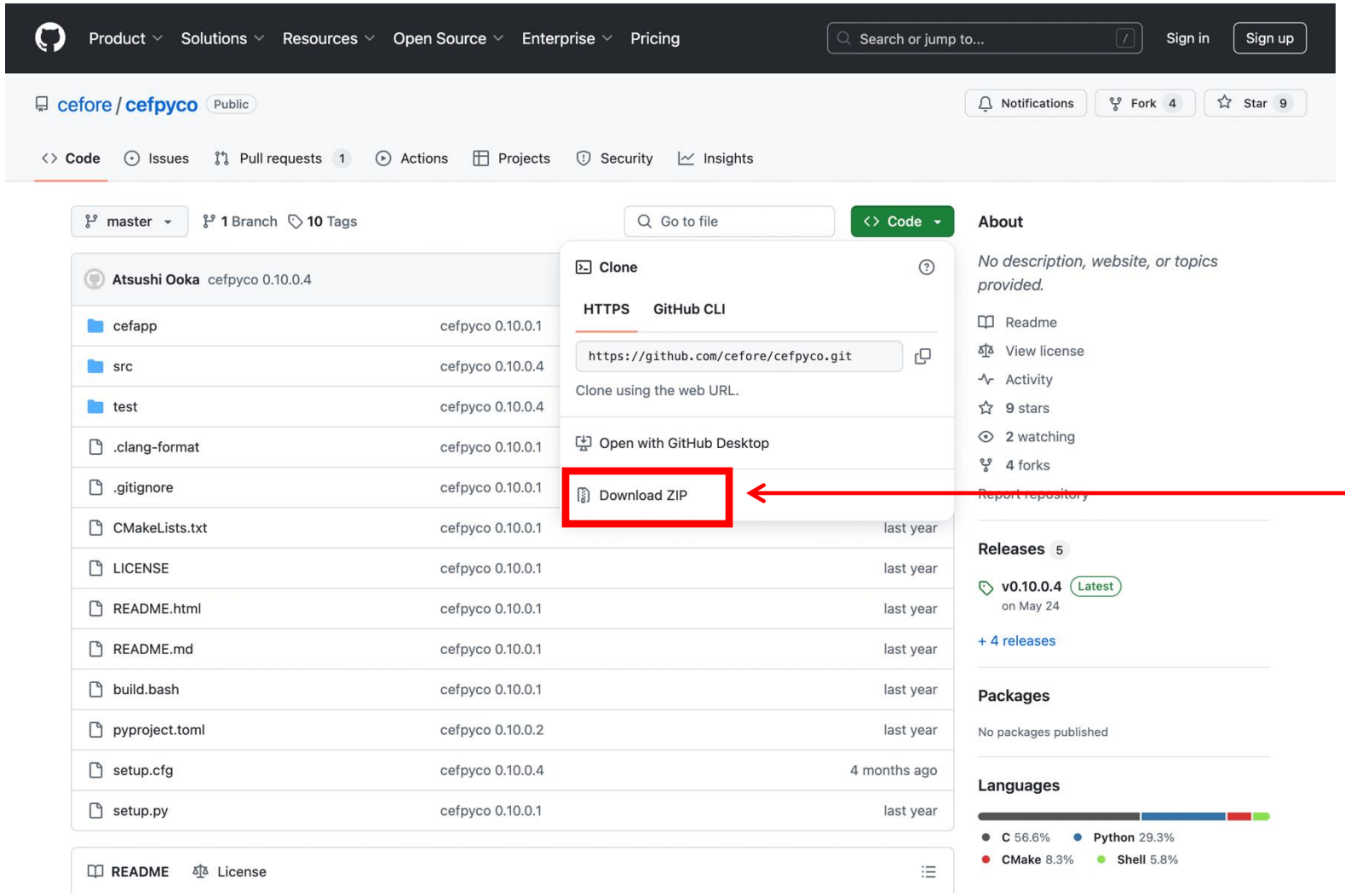
```python
import cefpyco

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

National Institute of
Information and Communications Technology

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



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)
```

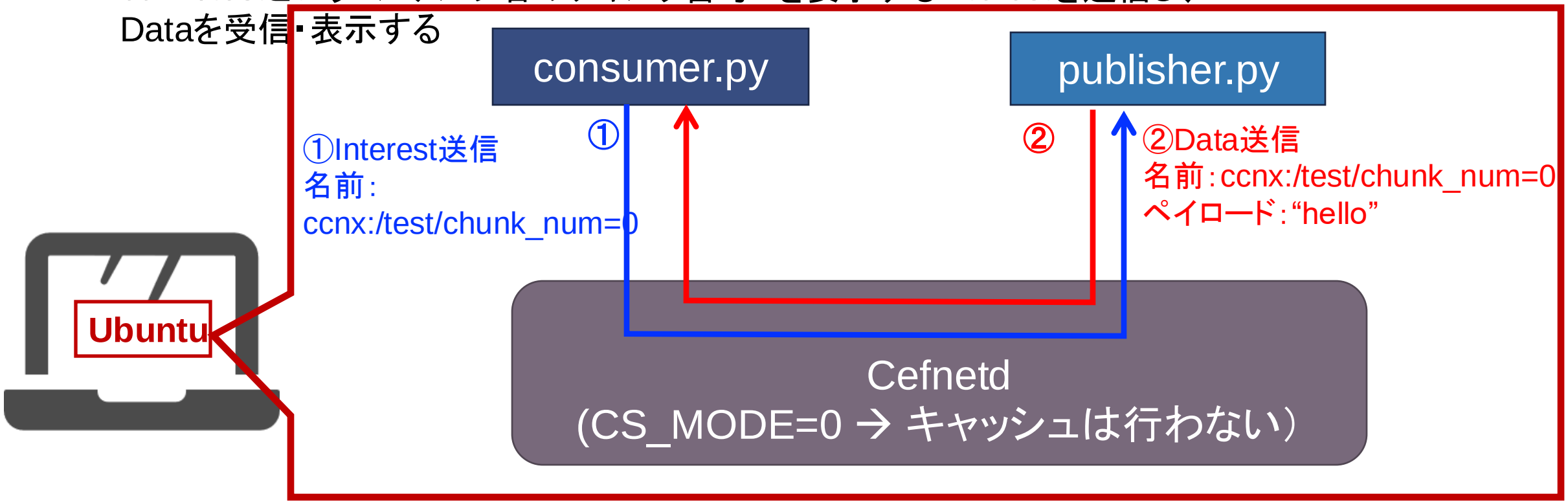
エラーが表示されないことを確認

National Institute of
Information and Communications Technology

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

ターミナル1

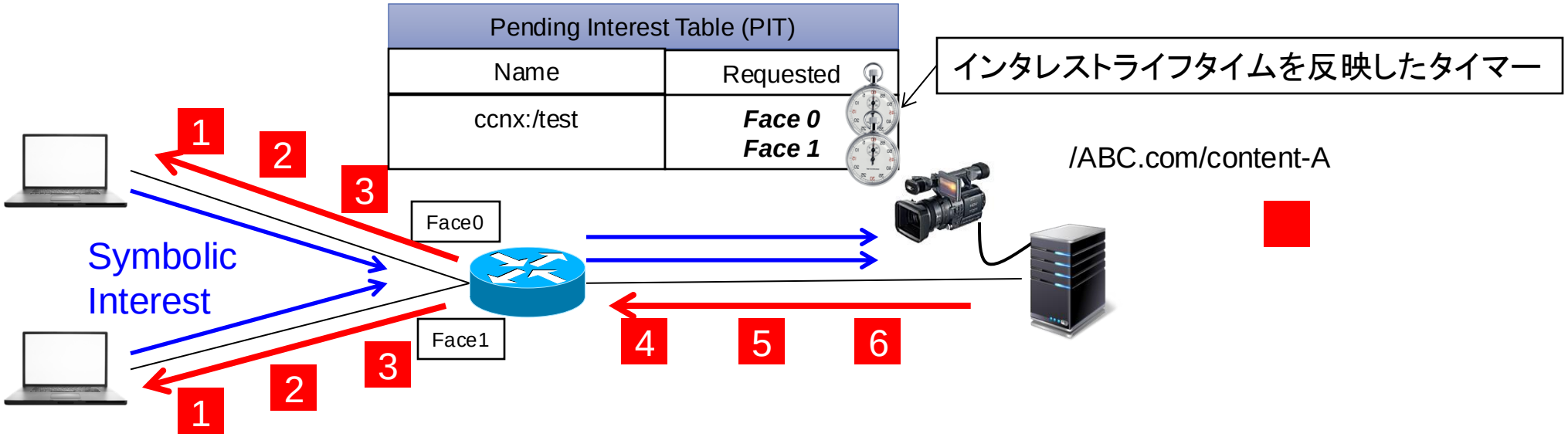
```
username:~/cefpyco-master$ cefnetdstart
username:~/cefpyco-master$ python3 publisher.py
```

ターミナル2

```
username:~/cefpyco-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）
```

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ンク番号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:
        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

ループ

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 Interest
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 Interest 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

ターミナル1

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

ターミナル2

```
username:~/cefpyco-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）
```

（参考）CCNxPacketInfoのプロパティ

プロパティ名	型	説明
is_succeeded, is_failed	bool	パケット受信の成否フラグ
is_interest, is_data	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）
name	string	URI形式(ccnx:/~)の名前
name_len	int	URI形式の名前の長さ(name TLV長ではない)
chunk_num	int	チャンク番号
payload	bytes	（Dataパケットの場合）コンテンツのデータ
payload_s	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）
payload_len	int	（Dataパケットの場合）コンテンツのデータのバイト長
version	int	受信パケットのバージョン値
type	int	受信パケットのタイプ値
actual_data_len	int	受信したパケットのヘッダを含むバイト長
end_chunk_num	int	コンテンツの最後のチャンク番号（指定時のみ有効）

（参考）CefpycoHandle API

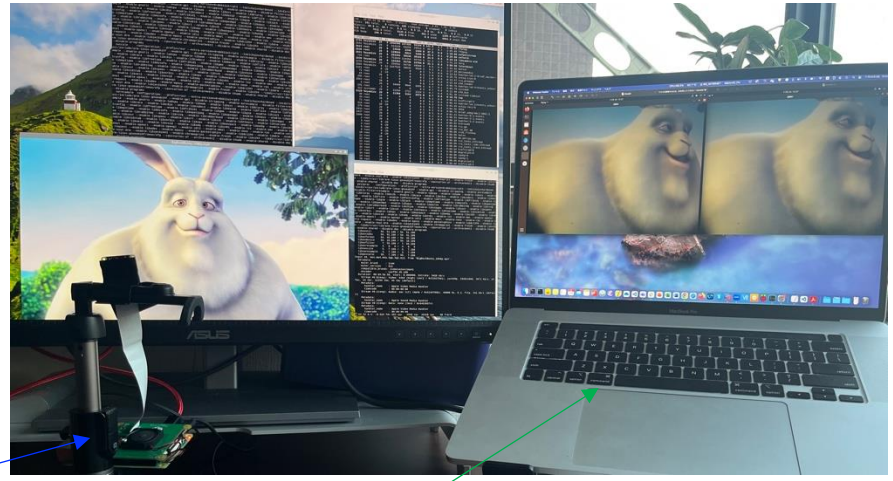
CefpycoHandle メソッド名	引数・返り値 (key=valueはデフォルト値のある省略可能引数)	説明
begin	• ceforedir=None: cefnetd.confの入ったディレクトリパス • portnum=9896: cefnetd のポート番号	cefnetdへの接続を開始する。with構文を利用する場合は自動で呼ばれる。
end	無し	cefnetdへの接続を修了する。with構文を利用する場合は自動で呼ばれる。
send_interest	• name: コンテンツ名 (ccnx:/…) • chunk_num=0: チャンク番号（負の数の場合はチャンク番号無し） • 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	• 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 • expirty=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	• 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	• 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	• 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。

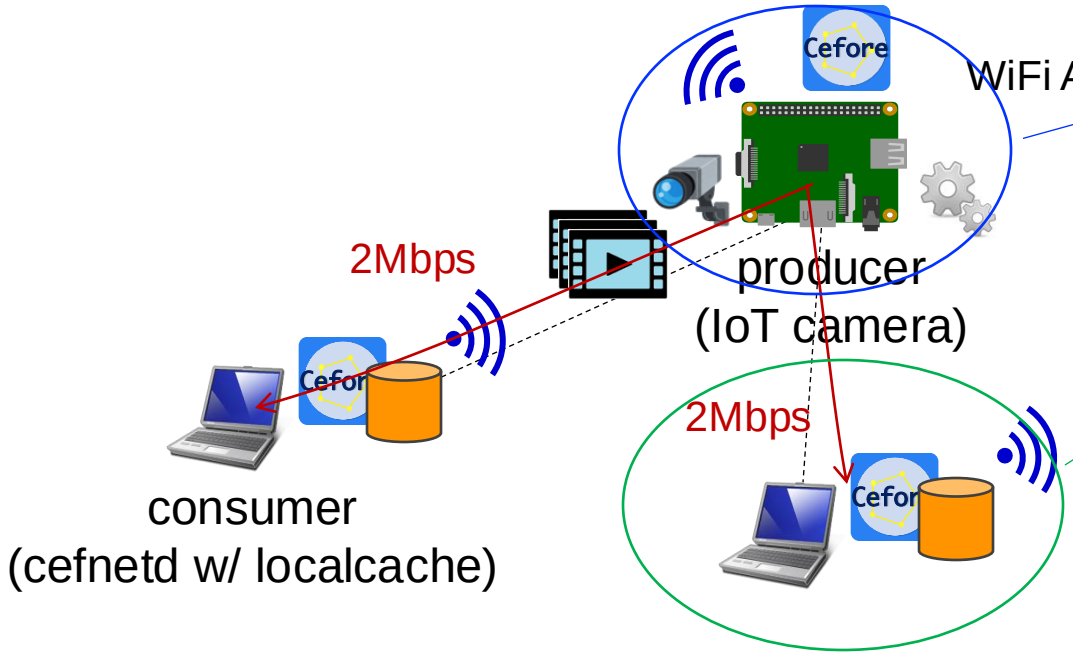
National Institute of
Information and Communications Technology

応用編・実践編:
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

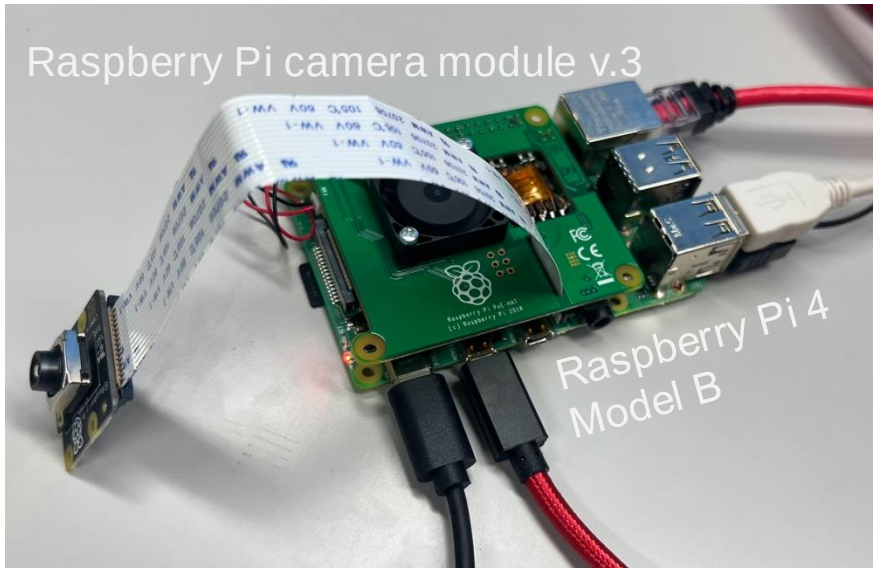
1. ラズパイを用いた ICN ライブ配信
   - ラズパイでは、publisher アプリが撮影した動画をCCNxパケットに変換してデータを送信
   - 参加者は consumer アプリとして動画を受信するリクエスト（Symbolic Interest）を送信して、データストリームを受信
   - 受信したデータをプレイヤーで再生





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

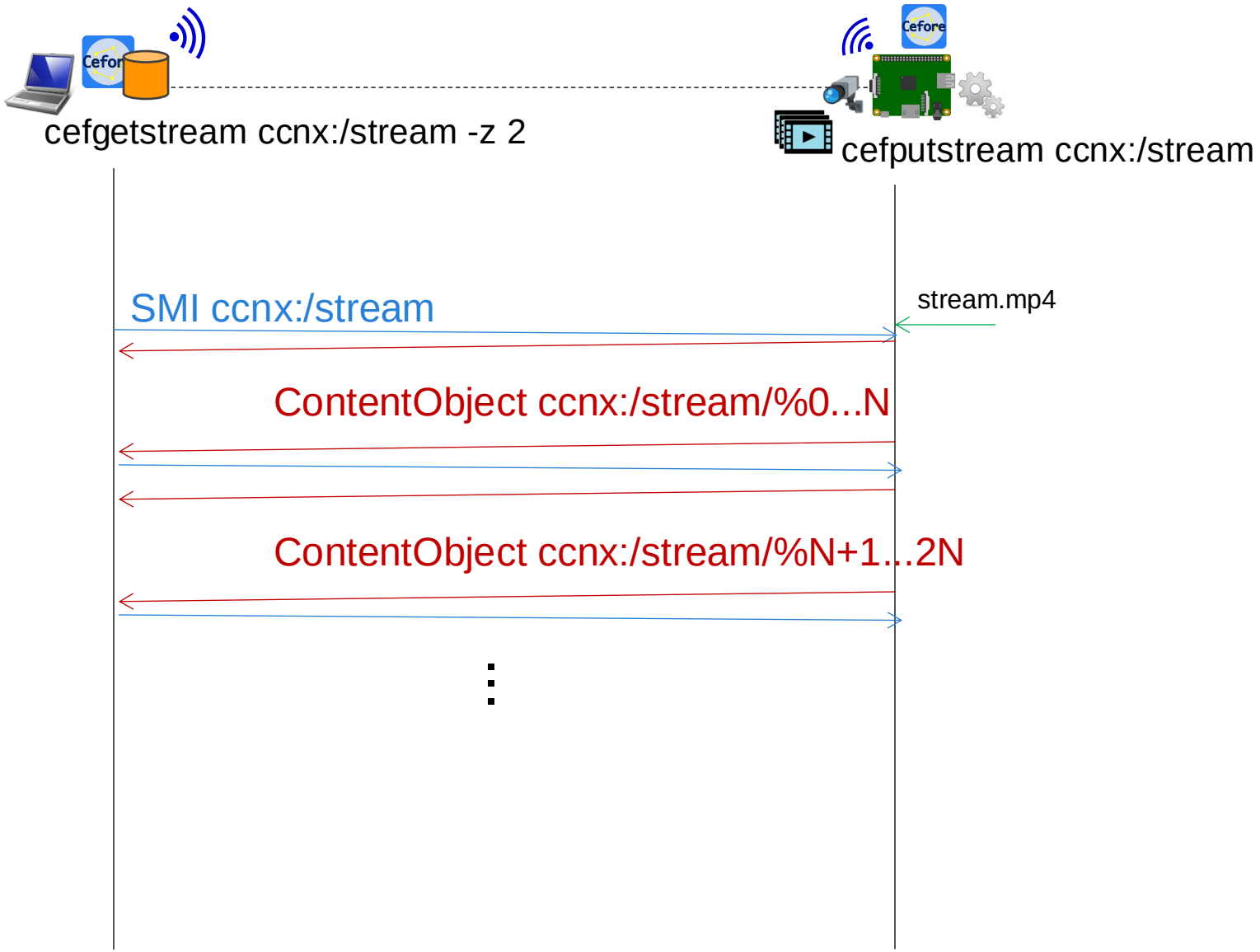


- ライブエンコーディング設定*

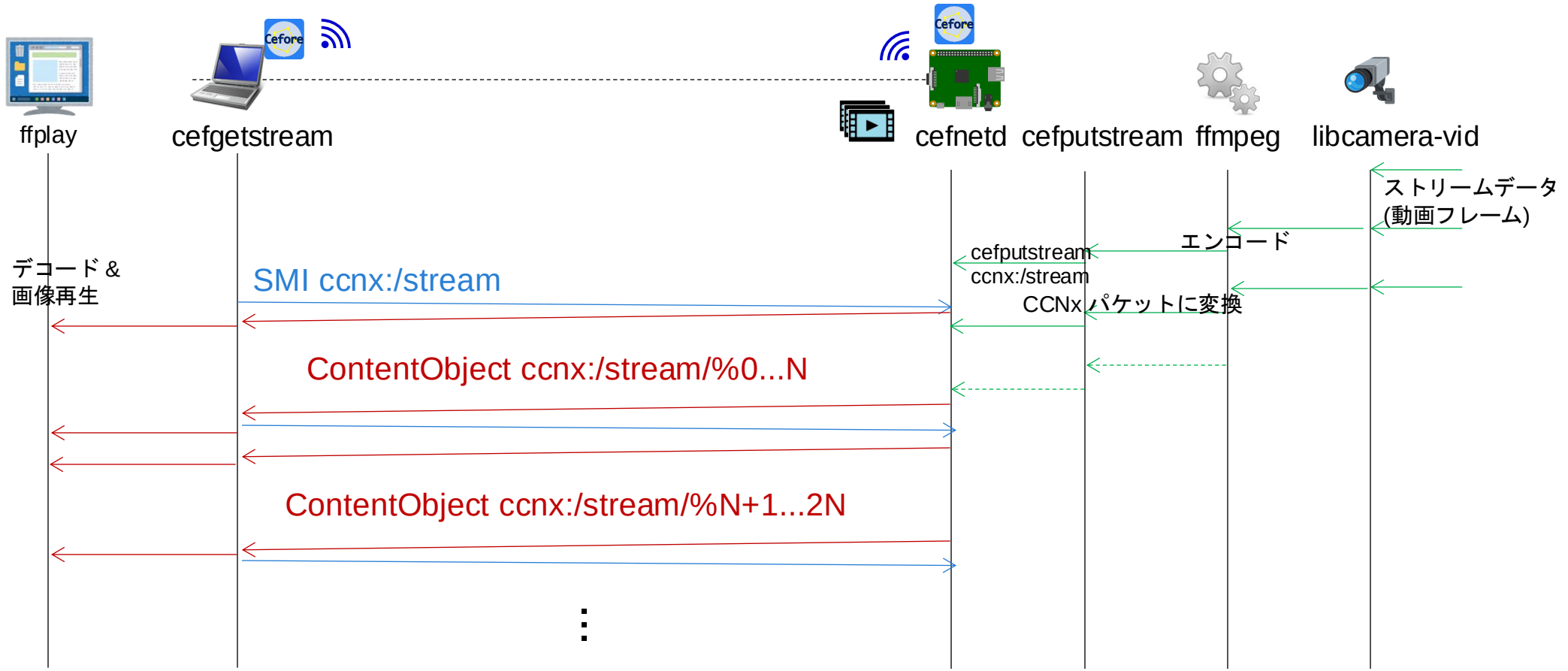
取り込み解像度 / フレームレート	最小ビットレート設定（Mbps）	最大ビットレート設定（Mbps）
2160p（4K）、30 fps	8 Mbps	35 Mbps
1440p、30 fps	5 Mbps	25 Mbps
1080p、30 fps	3 Mbps	8 Mbps
240p ～ 720p、30 fps	3 Mbps	8 Mbps

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

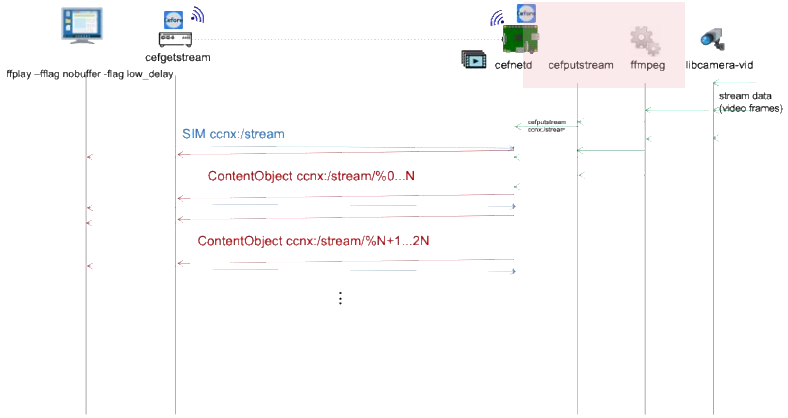
通信シーケンス（概要）


cefgetstream ccnx:/stream -z 2
cefputstream ccnx:/stream
SMI ccnx:/stream
stream.mp4
ContentObject ccnx:/stream/%0...N
ContentObject ccnx:/stream/%N+1...2N

通信シーケンス（詳細）


ffplay
cefgetstream
cefnetd
cefputstream
ffmpeg
libcamera-vid
ストリームデータ
(動画フレーム)
エンコード
cefputstream
ccnx:/stream
CCNx パケットに変換
デコード＆
画像再生
SMI ccnx:/stream
ContentObject ccnx:/stream/%0...N
ContentObject ccnx:/stream/%N+1...2N

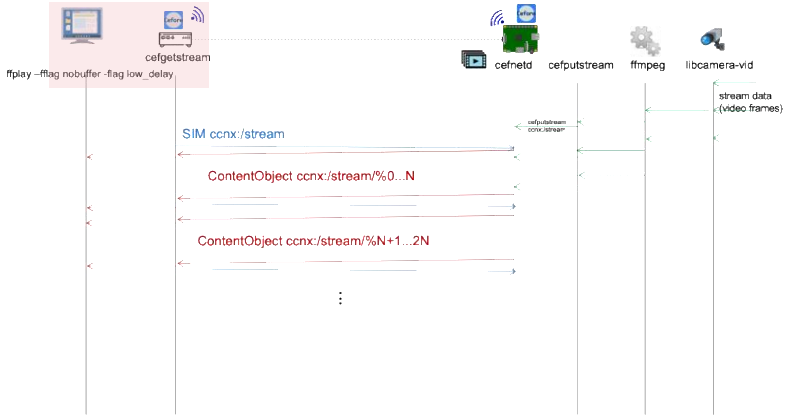
Publisher 側のプログラム



- ffmpeg
  - ffmpeg -i - -vcodec mpeg4 -b:v 2M -acodec aac -f mpegts -
    - -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

- cefputstream
  - cefputstream ccnx:/stream -r 2 -b 1400 -t 3600 -e 3600
    - cefore 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

- ラズパイの操作（複数人で実行する場合は各班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 mpegts - | 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
  - cefgetstream で受信したContentObject を ffplay でリアルタイムにデコード & 再生

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

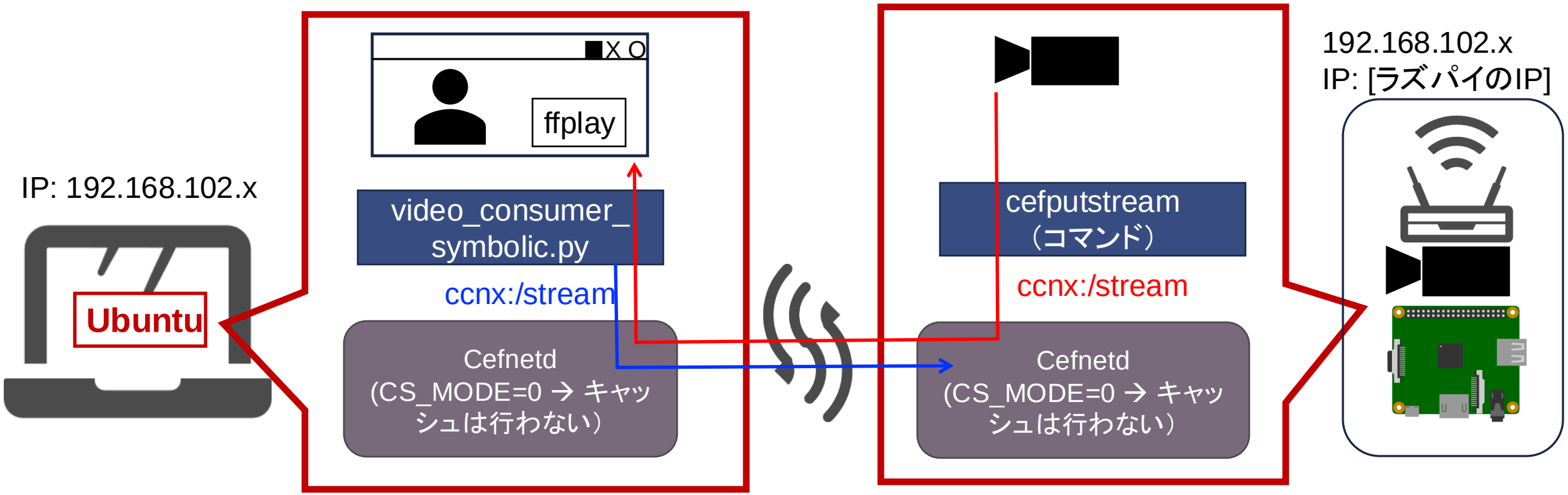
  - playback.bash を実行するのと同じ

National Institute of
Information and Communications Technology

Cefpycoを用いたリアルタイムビデオ受信・再生用Consumerアプリ

ゴール

- カメラ付きラズベリーパイから生成されているビデオデータを受信し、映像を表示させる
  - **ラズペリーパイ側の設定は終了している前提（別資料参照）**
- video_consumer_symbolic.py
  - Symbolic Interestを送信し、リアルタイムに生成されているビデオデータを受信し、標準出力する
    （標準出力されたビデオデータは、映像再生ソフト（ffplay)に入力され、再生される）



video_consumer_symbolic.py

- Symbolic Interestを送信し、リアルタイムに生成されているビデオデータを受信し、標準出力する
  （標準出力されたビデオデータは、映像再生ソフト（ffplay)に入力され、再生される）

```
import cefpyco

with cefpyco.create_handle() as handle:
    while True:
        handle.send_symbolic_interest(“ccnx:/test”)
        info = handle.receive()
        if info.is_succeeded and (info.name == “ccnx:/test”):
            print(“Recv-Success: recv-chunkNum=”, info.chunk_num, “ payload_len=”,
info.payload_len, sep=’’, file=sys.stderr)
            sys.stdout.buffer,write(info.payload)
```

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

受信データを標準出力

video_consumer_symbolic.py

実行手順

- ラズベリーパイ（Publisher）側

```
username:~/$ libcamera-vid --width 1280 --height 720 --bitrate 4096000 -n -t 0 -o
/dev/stdout | ffmpeg -I - -vcodec mpeg4 -b:v 4M -acodec aac -f mpegts - | cefputstream
ccnx:/stream -r 4 -b 1400 -t 3600 -e 3600
```

- Ubuntu（Consumer）側
  - FIB設定
    - /usr/local/cefore/cefnetd.fib ファイルに下記を追加
      - ccnx:/stream udp [ラズパイのIP]
  - キャッシュ設定の確認
    - /usr/local/cefore/cefore.confファイルにて、CS_MODE=0 となっているか確認
  - 下記のコマンドを実行

```
username:~/cefpyco-master$ python3 video_consumer_symbolic.py | ffplay -fflags nobuffer
-flags low_delay -
```

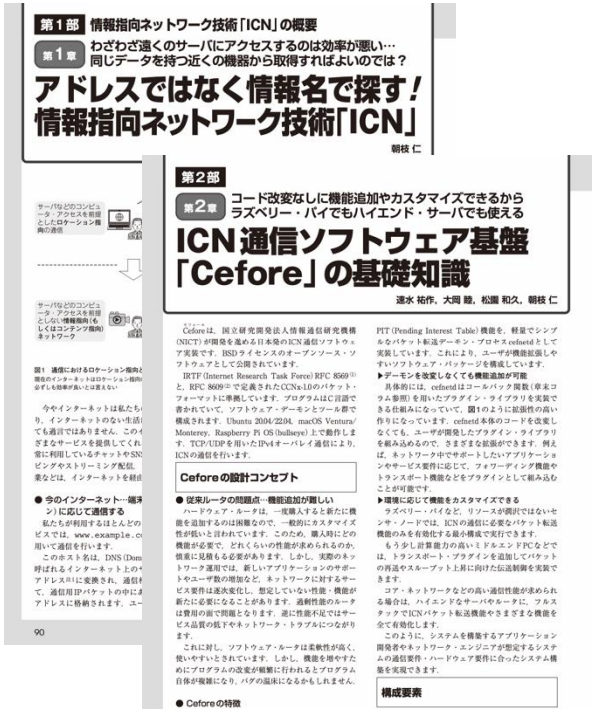
（適切に動作した場合、ffplayにてラズベリーパイのカメラ映像が描写される）

まとめ

- 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



- 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/

EB 招待論文

- 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

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 for 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

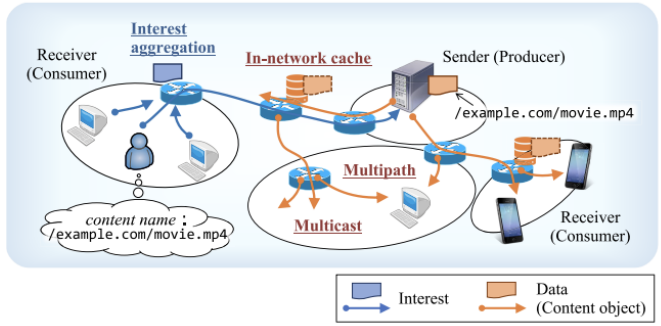


Fig. 1 ICN concept.

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

National Institute of
Information and Communications Technology

Thank you.

Cefore

https://cefore.net/

GitHub

https://github.com/cefore