国立研究開発法人
情報通信研究機構

第19回ICN研究会ワークショップ

Practice-B
Cefore・cefpyco を用いた通信

2021年 8月26日（木）・ 8月27日（金）

目次

Cefore を用いた通信

1. cefnetd の起動・停止
2. csmgrd の起動・停止
3. 設定ファイルの説明
   - cefnetd.conf
   - cefnetd.fib
   - csmgrd.conf
4. 設定ファイルの変更
   - ルーティングテーブルの設定
   - キャッシュ利用設定
5. ファイルのアップロードとダウンロード

cefpyco を用いた通信

1. python から cefnetd への接続
2. Data パケットの送信
3. Interest パケットの送信
4. パケットの受信
5. 簡易 Consumer アプリ
6. 簡易 Producer アプリ
7. CefApp 解説

付録：インストールマニュアル

※ Docker を使用せず自身の PC 上に環境を構築する場合

1. Cefore のインストール
2. cefpyco のインストール

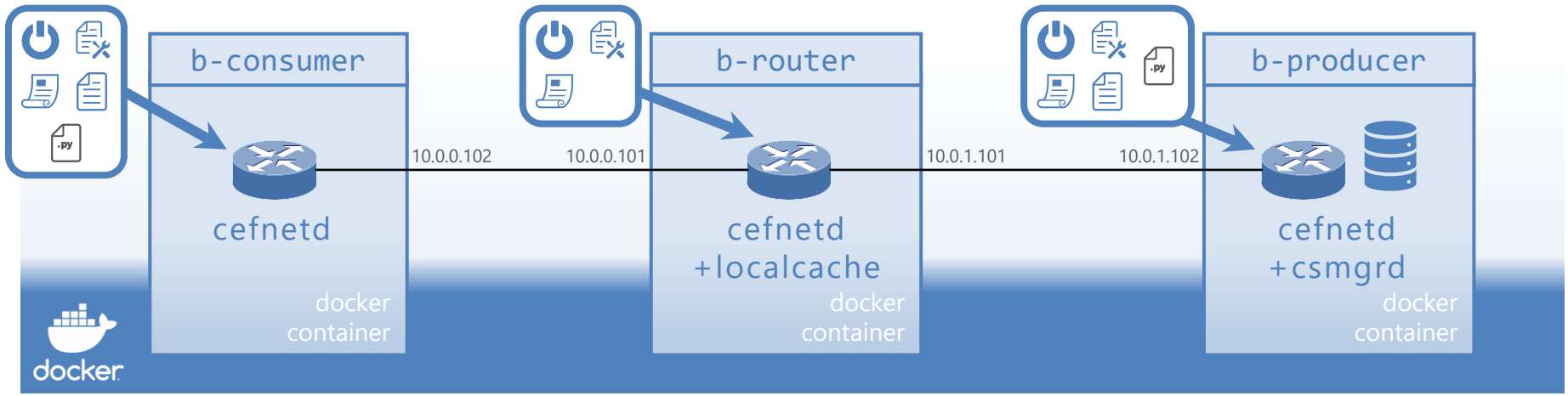
スライドの種類

表記無し	Practice	Advanced
説明スライド	実践スライド （実際に手を動かす）	高度な内容のスライド （今回使用しない参考情報）

Practice-B 全体像

目標

Cefore・cefpyco を使って
自分でネットワークを作って通信できるようになること

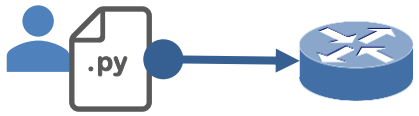


Cefore を用いた通信

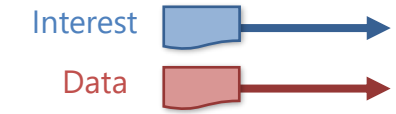
- cefnetd/csmgrd の起動・停止
- cefnetd/csmgrd のステータス表示
- 設定ファイルの理解と変更
  - cefnetd.conf
  - cefnetd.fib/cefroute
  - csmgrd.conf
- ファイル送受信
  - 送信: cefputfile
  - 受信: cefgetfile
- ストリーミング送受信
  - 送信: cefputstream
  - 受信: cefgetstream
- ネットワーク管理: ccninfo

cefpyco を用いた通信

cefnetd への接続



パケットの送受信

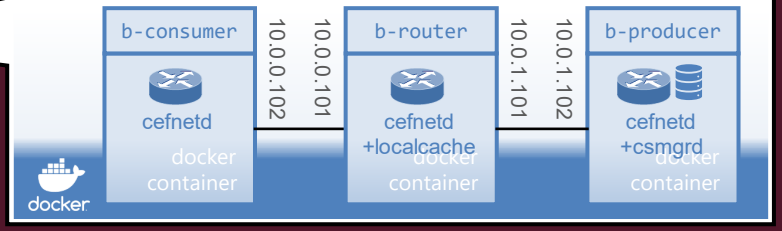


Practice-B 用の環境構築

- Practice-A でダウンロードした 2021-hands-on を利用
- 2021-hands-on/practice-B に移動して setup.bash を実行

```
cefore:~$ cd 2021-hands-on/practice-B
cefore:~/2021-hands-on/practice-B$ ./setup.bash
[INFO] Build docker images.
router uses an image, skipping
consumer uses an image, skipping
Building producer
[+] Building 0.2s (14/14) FINISHED
 => [internal] load build definition from Dockerfile                    0.0s
...(中略)
 => => naming to docker.io/cefore/practice-b                            0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[INFO] Compose docker containers.
Creating network "upward" with driver "bridge"
Creating network "downward" with driver "bridge"
Creating b-router   ... done
Creating b-consumer ... done
Creating b-producer ... done
[INFO] Check docker containers are up.
CONTAINER ID   IMAGE               COMMAND       CREATED         STATUS                  PORTS     NAMES
xxxxxxxxxxxx   cefore/practice-b   "/bin/bash"   1 second ago   Up Less than a second             b-consumer
xxxxxxxxxxxx   cefore/practice-b   "/bin/bash"   1 second ago   Up Less than a second             b-router
xxxxxxxxxxxx   cefore/practice-b   "/bin/bash"   1 second ago   Up Less than a second             b-producer
[SUCCESS] Completed.
```

下図の環境（ただし cefnetd 除く）を生成



b-consumer, b-router, b-producer の３つのコンテナが生成されている

抜粋: Cefore の機能・ツール一覧

機能	形態	説明
cefnetd	daemon	フォワーディングデーモン
cefnetdstart	utility	フォワーディングデーモン起動ユーティリティ
cefnetdstop	utility	フォワーディングデーモン停止ユーティリティ
cefstatus	utility	cefnetdのstatus標準出力ユーティリティ
cefroute	utility	FIB操作ユーティリティ
ccninfo/cefinfo	tool	ネットワーク管理ツール
cefputfile	tool	任意のファイルを name 付きコンテンツに変換しcefnetdへ入力する
cefgetfile	tool	cefnetdを介して取得したコンテンツをファイルとして出力する
cefgetchunk	tool	指定された name 付きコンテンツを取得し、ペイロードを標準出力する
cefputstream	tool	標準入力を name 付きコンテンツに変換しcefnetdへ入力する
cefgetstream	tool	cefnetdを介して取得したコンテンツを標準出力する
csmgrd	daemon	コンテンツストア管理デーモン (CS_MODE=2)
csmgrdstart	utility	csmgrd起動ユーティリティ
csmgrdstop	utility	csmgrd停止ユーティリティ
csmgrstatus	utility	csmgrdのstatus標準出力ユーティリティ

抜粋: 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。

抜粋: CcnPacketInfo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）

国立研究開発法人
情報通信研究機構

Cefore を用いた通信

Cefore を用いた通信 | cefpyco を用いた通信 | 付録：インストールマニュアル

Practice 手順

1. cefnetd の起動・停止
2. csmgrd の起動・停止
3. 設定ファイルの説明
   (3-1) cefnetd.conf
   (3-2) cefnetd.fib
   (3-3) csmgrd.conf
4. 設定ファイルの変更
   (4-1) ルーティングテーブルの設定
   (4-2) キャッシュ利用設定
5. ファイルのアップロードとダウンロード
   (5-1) アップロード（キャッシュ利用）
   (5-2) ダウンロード

(1) cefnetd の起動・停止

① b-consumer にログイン

```
cefore:~/2021-hands-on/practice-B$ ./login-consumer.bash
root@b-consumer:/cefore#
```

② cefnetd を起動・起動確認

```
root@b-consumer:/cefore# cefnetdstart
root@b-consumer:/cefore# cefstatus
Version    : 1
Port       : 9896
Rx Frames  : 0
Tx Frames  : 0
Cache Mode : None
Faces : 6
  faceid =   4 : IPv4 Listen face (udp)
  faceid =   0 : Local face
  faceid =   5 : IPv6 Listen face (udp)
  faceid =  16 : Local face
  faceid =   6 : IPv4 Listen face (tcp)
  faceid =   7 : IPv6 Listen face (tcp)
FIB(App) :
  Entry is empty
FIB :
  Entry is empty
PIT(App) :
  Entry is empty
PIT :
  Entry is empty
```

ポート番号

送受信フレーム数

キャッシュモード（後で説明）

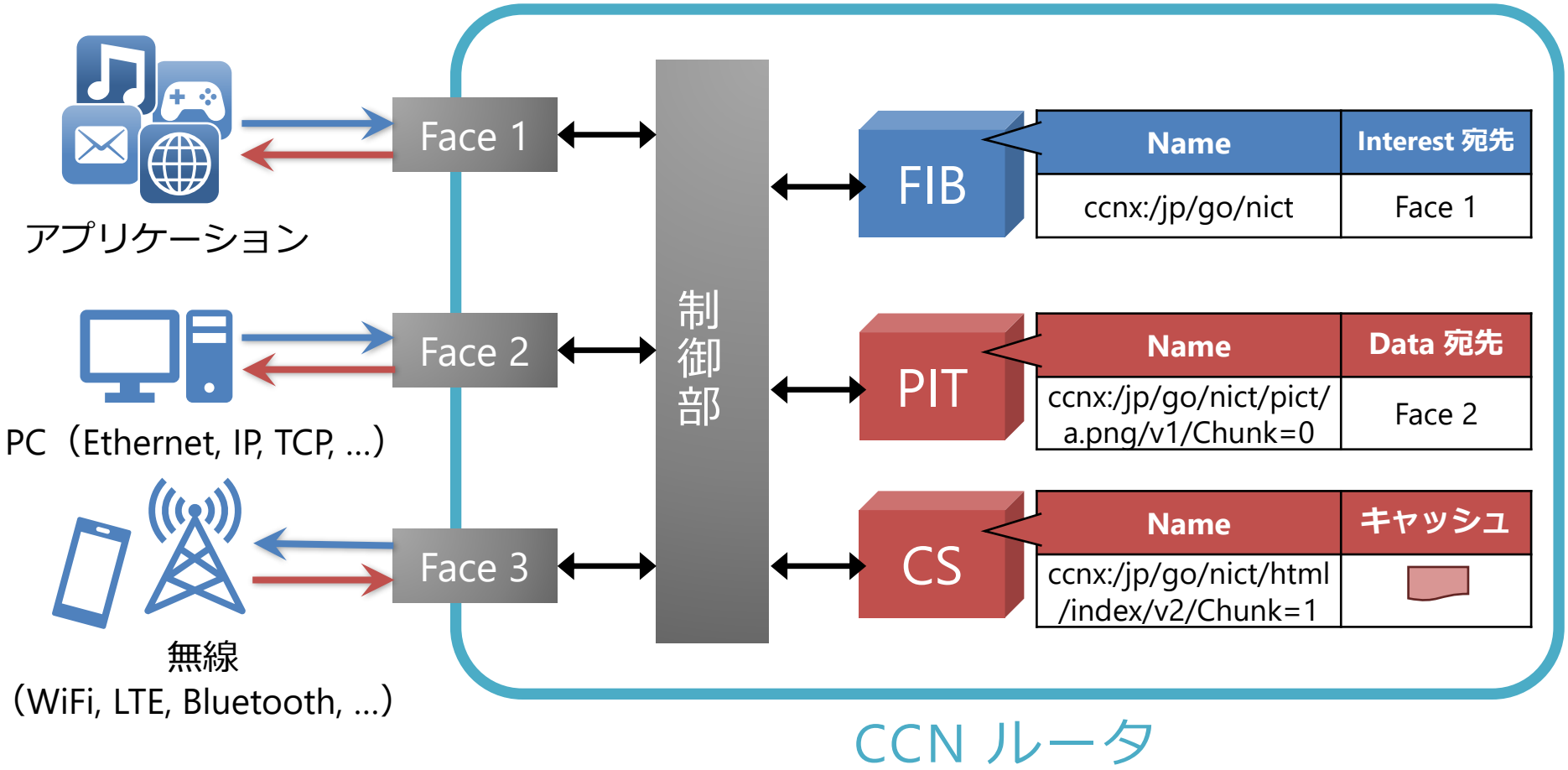
Face 情報

FIB 情報

PIT 情報

再掲：ICN ルータモデル

- Face：外部と接続する論理的なインターフェース
- 制御部：名前検索を行い FIB・PIT・CS を参照・更新



(1) cefnetd の起動・停止

③　cefnetd を停止・停止確認

```
root@b-consumer:/cefore# cefnetdstop
root@b-consumer:/cefore# cefstatus
2021-08-26 12:34:56.789 [cefctrl] ERROR: cef_client_connect (connect:No such file or directory)
2021-08-26 12:34:56.789 [cefctrl] ERROR: Failed to connect to cefnetd.
```

cefnetd が起動していないので
エラーが表示される

(2) csmgrd の起動・停止

① csmgrd を起動・起動確認

```
root@b-consumer:/cefore# csmgrdstart
root@b-consumer:/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
```

② csmgrd を停止・停止確認

```
root@b-consumer:/cefore# csmgrdstop
root@b-consumer:/cefore# csmgrstatus ccnx:/

Connect to 127.0.0.1:9799
ERROR : connect to csmgrd cefstatus
```

(3) 設定ファイルの説明

① /usr/local/cefore の中身を確認

```
root@b-consumer:/cefore# ls /usr/local/cefore
cefnetd.conf   cefnetd.fib   cefnetd.key   csmgr_fsc_615   csmgrd.conf
default-private-key   default-public-key   plugin   plugin.conf
```

ファイル名	説明
cefnetd.conf	cefnetd の設定ファイル
cefnetd.fib	cefnetd の FIB エントリの設定ファイル
csmgrd.conf	csmgrd の設定ファイル
• cefnetd.key • ccore-public-key • default-public-key • default-private-key	Interest と Data の Validation に使用する公開鍵と秘密鍵の設定ファイル、およびデフォルトで使用する公開鍵と秘密鍵
• plugin.conf • plugin/	プラグインの設定ファイルとディレクトリ （プラグイン使用時のみ使用）

今回はcefnetd.conf・cefnetd.fib・csmgrd.confを設定

※環境変数$CEFORE_DIRを変更してインストールした場合は"$CFEORE_DIR/cefore"下に存在

(3-1) cefnetd.conf

■設定ファイルcefnetd.confの内容

```
root@b-consumer:/cefore# cat /usr/local/cefore/cefnetd.conf
#
# cefnetd.conf
#

#
# Node Name is specified in URI format.
#   ex) abc.com/tokyo/router-a
#
#NODE_NAME=""

# Operational Log Level
#  0: Error only
#  1: Warning and Error
#  2: Info, Warning, and Error
#
#CEF_LOG_LEVEL=0

#
# Port number used by cefnetd.
# This value must be higther than 1024 and lower than 65536.
#
#PORT_NUM=9896
...
```

"#" で始まる行はコメント行

インストール直後の雛形ではすべての
パラメータがコメントアウトされている
（雛形のコメントに書かれている値は
各パラメータのデフォルト値）

cefnetd.conf の主なパラメータ

- 「parameter=value」の書式で記述する
  - 例: キャッシュ無しモードからcsmgrd使用モードに変更する場合
    - CS_MODE=2
- キャッシュを使用する場合に設定すべきパラメータ

パラメータ	説明	デフォルト	値の範囲・意味
CS_MODE	CSの動作モード	0	0：CSを使用しない 1：cefnetdのローカルキャッシュ 2：csmgrd
BUFFER_CAPACITY	cefnetdの最大Dataバッファサイズ	30000	0 ≦ n < 65536
CSMGR_NODE	cefnetdが接続するcsmgrdのIPアドレス	localhost	
CSMGR_PORT	cefnetdが接続するcsmgrdのTCPポート番号	9799	1024 < p < 65536

cefnetd.conf の詳細パラメータ①

■cefnetd ネットワーク設定

パラメータ	説明	デフォルト	値の範囲・意味
PORT_NUM	cefnetdが使用するポート番号（単一のPC上でcefnetdを複数起動する場合等に設定）	9896	1024 < p < 65536
LOCAL_SOCK_ID	UNIXドメインソケットのID文字列（単一のPC上でcefnetdを複数起動する場合等に設定）	0	
NODE_NAME	cefnetd のノード名、無指定の場合は IP アドレスを使用	-	

■cefnetd のローカルキャッシュ設定

- CS_MODE=1 の場合にのみ使用

パラメータ	説明	デフォルト	値の範囲・意味
LOCAL_CACHE_CAPACITY	キャッシュ容量（単位：Data数）	65535	1 < n < 8M
LOCAL_CACHE_INTERVAL	期限切れコンテンツチェック間隔（秒）	60	1 < n < 86400 (2 hours)
LOCAL_CACHE_DEFAULT_RCT	Dataのデフォルトのキャッシュ期限 （Recommended Cache Time; RCT） （Data が RCT を指定している場合はそちらを優先） （単位：ミリ秒）	600 (10分)	0 < n < 3600

cefnetd.conf の詳細パラメータ②

■cefnetd テーブルエントリ設定

パラメータ	説明	デフォルト	値の範囲・意味
PIT_SIZE	最大PITエントリ数	2048	1 < n < 65536
FIB_SIZE	最大FIBエントリ数	1024	1 < n < 65536
PIT_SIZE_APP	最大PITエントリ数 (アプリ用)	64	1 < n < 1025
FIB_SIZE_APP	最大FIBエントリ数 (アプリ用)	64	1 < n < 1M
BUFFER_ CACHE_TIME	cefnetd のバッファへの保持時間（ミリ秒）	10000 (10秒)	0 < n
REGULAR_INTEREST_ MAX_LIFETIME	通常の Interest （Symbolic でないもの） のライフタイム（PIT エントリの生存時間）の最大値（秒）。パケットに記載された値よりも優先される	2	0 < n
SYMBOLIC_INTEREST_ MAX_LIFETIME	Symbolic Interest のライフタイム（PIT エントリの生存時間）の最大値（秒）。パケットに記載された値よりも優先される	4	0 < n

cefnetd.conf の詳細パラメータ③

■cefnetd 転送戦略設定

パラメータ	説明	デフォルト	値の範囲・意味
INTEREST_ RETRANSMISSION	PIT 消失前に Interest が再送されたときの挙動。以下の二種類から選択 • RFC8569: 同じ Face（PIT登録済みの Face）から来た場合は再送と見なして転送し、それ以外の場合は破棄する（集約する） • SUPPRESSIVE: 常に破棄する（集約する）	RFC8569	左記
FORWARDING_ INFO_STRATEGY	FIB に複数転送先が設定されている場合の戦略。戦略は以下の二種類 • 0: 任意の一つ（最初に登録されたもの）に転送 • 1: すべてに転送	0	左記
SYMBOLIC_ BACKBUFFER	Symbolic Interest (SMI) の挙動に関するパラメータ ※詳細：SMI はリアルタイムストリーミング用なので、最後に転送した Data のチャンク番号を覚えておき、基本的にはそれよりチャンク番号が大きい（＝最新の）もののみ転送して、チャンク番号が小さい（＝過去の）ものの転送を抑制する。しかし、 NW 環境が不安定な場合は順番が前後することもありうるので、このパラメータで指定した値だけ遡ったチャンク番号の Data の転送は許容する。確実に順番通りに来る場合は0でいいが、不安定な環境では値を大きくするとよい。	100	0 ≦ n

cefnetd.conf の詳細パラメータ④

■ログ設定

パラメータ	説明	デフォルト	値の範囲・意味
CEF_LOG_LEVEL	出力ログの詳細度（[0] Error のみ表示、[1] Warning+Error、 [2] Error+Watning+Info）	0	0 ≦ n ≦ 2
CEF_DEBUG_LEVEL	デバッグ用ログの詳細度（configure時に"--enable-debug"を指定する必要有）	0	0 ≦ n ≦ 3 （※n=3にするとパケットダンプまで表示される）

(3-2) cefnetd.fib

■ 設定ファイル cefnetd.fib の内容

```
root@b-consumer:/cefore# cat /usr/local/cefore/cefnetd.fib
#ccnx:/example udp 10.0.1.1
```

■ cefnetd.fib は静的なFIBエントリの設定ファイル

- 書式：name (udp|tcp) ip_address[:port] …
- 設定例
  - ccnx:/ udp 10.0.1.1
  - ccnx:/cinema tcp 10.0.2.1:8888 10.0.2.2:9999
  - ccnx:/news/today udp 10.0.3.1 10.0.3.2:8765 10.0.3.3:9876

  ※経路を複数指定した場合、デフォルトでは利用可能な最初のノードにのみ転送する。cefnetd.fib の FORWARDING_INFO_STRATEGY の値を変更すれば、全ノードへの転送も選択できる。

■ 動的なFIBエントリの設定はcefrouteで行う

- 追加: cefroute add name (udp|tcp) ip_address
- 削除: cefroute del name ip_address

(3-3) csmgrd.conf

- 書式やファイルの場所は cefnetd.conf と同じ
  - 「parameter=value」の形式で記述
  - "#"で始まる行はコメント
  - /usr/local/cefore/csmgrd.conf に配置

```
root@b-consumer:/cefore# cat /usr/local/cefore/csmgrd.conf
#
# csmgrd.conf
#

# Operational Log Level
#  0: Error only
#  1: Warning and Error
#  2: Info, Warning, and Error
#
#CEF_LOG_LEVEL=0

#
# Port number used by csmgrd.
# This value must be higher than 1024 and lower than 65536.
#
#PORT_NUM=9799
…
```

csmgrd.conf の主なパラメータ

パラメータ	説明	デフォルト	値の範囲・意味
CACHE_TYPE	csmgrdが使用するPlugin名称（文字列）	filesystem	• filesystem • memory （詳細は後述）
CACHE_INTERVAL	csmgrdの期限切れコンテンツチェック間隔 （単位：ミリ秒）	10,000 （10秒毎）	1,000 < n < 86,400,000 （1秒～24時間）
CACHE_DEFAULT_RCT	Dataのデフォルトのキャッシュ期限 （Recommended Cache Time; RCT） （Data が RCT を指定している場合はそちらを優先） （単位：ミリ秒）	600,000 （10分間）	1,000 < n < 3,600,000 （1秒～24時間）
CACHE_ALGORITHM	キャッシュ置換アルゴリズムライブラリ	libcsmgrd_lru	• None • libcsmgrd_lru • libcsmgrd_lfu • libcsmgrd_fifo
CACHE_PATH	ファイルシステムキャッシュのキャッシュ保存用ディレクトリ（ファイルシステムキャッシュ使用時は必須）	/usr/local/cefore	
CACHE_CAPACITY	キャッシュ容量（単位：Data数）	819,200	1 < n < 64 G

csmgrd.conf の詳細パラメータ①

パラメータ	説明	デフォルト	値の範囲
PORT_NUM	csmgrdが使用するポート番号	9799	1024 < p < 65536

ALLOW_NODE	• csmgrdへの接続を許可するホストのIPアドレス
	• リモートでのcsmgrdへの接続を許可する場合のみ設定（デフォルトではローカルホストのみ接続可能）
	• " ALL"と記述すると、全ての接続を許可
	• 「, （カンマ）」区切りで複数指定可能
	• 複数行に分けての指定も可能
	• サブネットを使用した指定も可能
	• 設定例
	  • ALLOW_NODE=192.168.1.1,192.168.1.2
	  • ALLOW_NODE=192.168.2.0/24
		localhost	

CEF_LOG_LEVEL	出力ログの詳細度（[0] Error のみ表示、 [1] Warning+Error、 [2] Error+Watning+Info）	0	0 ≦ n ≦ 2
CEF_DEBUG_LEVEL	デバッグ用ログの詳細度（configure時に"--enable-debug"を指定する必要有）	0	0 ≦ n ≦ 3
LOCAL_SOCK_ID	UNIXドメインソケットのID文字列（単一のPC上でcsmgrdを複数起動する場合等に設定）	0	

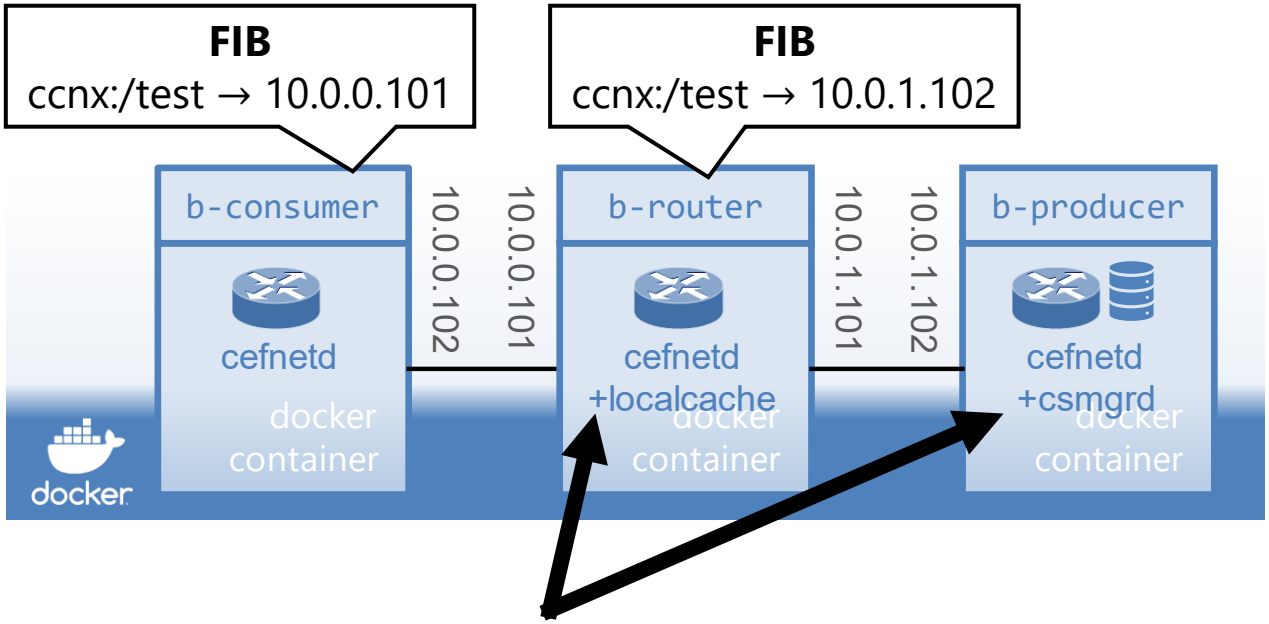
csmgrd.conf の詳細パラメータ②

パラメータ	説明	デフォルト	値の範囲
CACHE_ALGO_NAME_SIZE	想定される Data の平均ネーム長（単位：Byte）（キャッシュの使用メモリ領域の計算に使用）	256	100 < n < 8000
CACHE_ALGO_COB_SIZE	想定される Data の平均パケットサイズ（単位：Byte）（キャッシュの使用メモリ領域の計算に使用）	2048	500 < n < 64 K

(4) 設定ファイルの変更

(4-1) ルーティングテーブルの設定

- **cefnetd.fib を変更**して b-consumer から b-producer までの経路を設定しよう



(4-2) キャッシュ利用設定

- **cefnetd.conf の “CS_MODE“ を変更**してキャッシュを利用しよう
  - b-router は cefnetd のローカルキャッシュを使用
  - b-producer は csmgrd を使用
- **csmgrd.conf** でキャッシュの挙動を設定しよう

(4) 設定ファイルの変更方法

本 Practice では以下のいずれかの方法で設定ファイルを変更

A) コンテナ内の /usr/local/cefore のファイルを直接編集する

- ログインして vi, emacs, nano 等を使用するか、
  VisualStudioCode の docker 用プラグインで編集

B) 2021-hands-on/practice-B/bin/cefore-conf.xxxxx のファイルを変更

- それぞれ以下のコンテナの設定ファイルと対応する
  - cefore-conf.consumer ⇔ b-consumer
  - cefore-conf.router ⇔ b-router
  - cefore-conf.producer ⇔ b-producer
- 編集した後、 4_reflect-conf.bash を実行すると反映される

本 Practice では (B) の方法で説明

(4-1) ルーティングテーブルの設定

① **cefore-conf.consumer/cefnetd.fib** に以下を入力

```
ccnx:/test udp 10.0.0.101
```

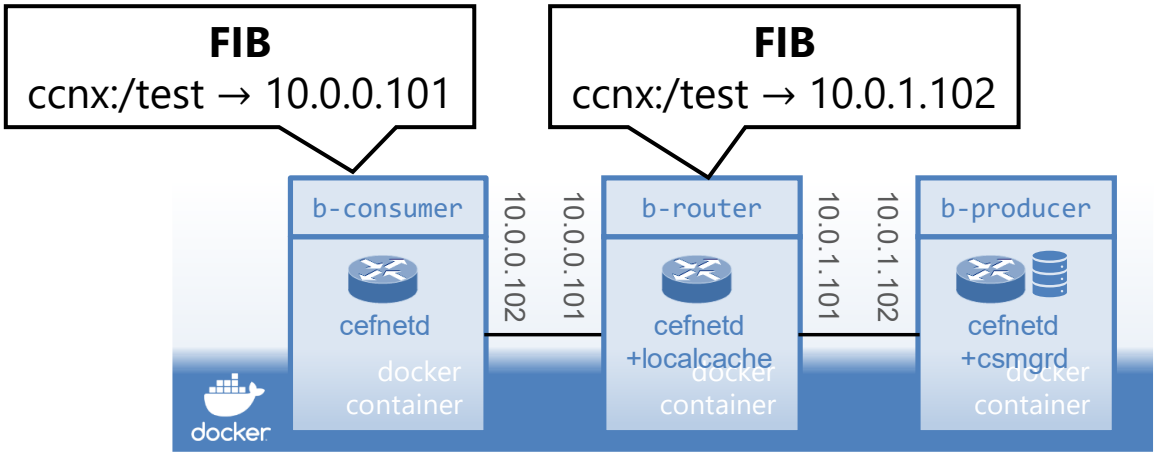
② **cefore-conf.router/cefnetd.fib** に以下を入力

```
ccnx:/test udp 10.0.1.102
```

③ 4_reflect-conf.bash を実行して設定をコンテナに反映

※ コンテナにログインしている場合は exit や Crtl-D でホストに戻る

```
root@b-consumer:/cefore# exit
cefore:~/2021-hands-on/practice-B$ ./4_reflect-conf.bash
[SUCCESS] Completed.
```



(4-1) ルーティングテーブルの設定

④ b-consumer にログインして cefnetd を起動、cefstatus を実行

```
cefore:~/2021-hands-on/practice-B$ ./login-consumer.bash
root@b-consumer:/cefore# cefnetdstart
root@b-consumer:/cefore# cefstatus
Version    : 1
Port       : 9896
Rx Frames  : 0
Tx Frames  : 0
Cache Mode : None
Faces : 7
  faceid =   4 : IPv4 Listen face (udp)
  faceid =   0 : Local face
  faceid =  16 : address = 10.0.0.101:9896 (udp)
  faceid =   7 : IPv6 Listen face (tcp)
  faceid =  17 : Local face
FIB(App) :
FIB : 1
  ccnx:/test
    Faces : 16 (-s-)
  Entry is empty
PIT :
  Entry is empty
```

Face と FIB エントリが追加されている

Practice

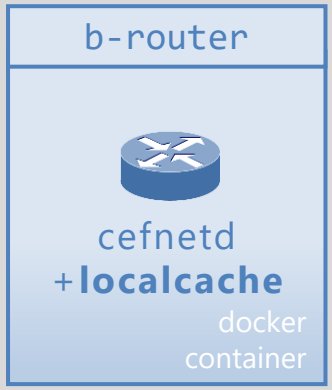
(4-2) キャッシュ利用設定

① cefore-conf.router/cefnetd.conf で CS_MODE=1 に設定

cefore-conf.router/cefnetd.conf

```
…
## Content Store used by cefnetd
#  0 : No Content Store
#  1 : Use cefnetd's Local cache
#  2 : Use external Content Store (use csmgrd)
#  3 : Use external Content Store (use conpubd)
#CS_MODE=0
CS_MODE=1
…
```

追加

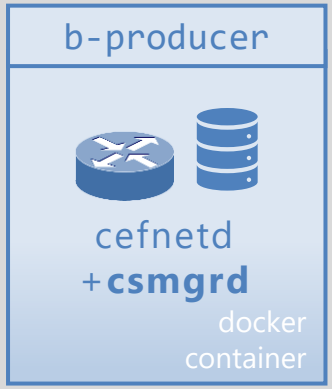


② cefore-conf.producer/cefnetd.conf で CS_MODE=2 に設定

cefore-conf.producer/cefnetd.conf

```
…
## Content Store used by cefnetd
#  0 : No Content Store
#  1 : Use cefnetd's Local cache
#  2 : Use external Content Store (use csmgrd)
#  3 : Use external Content Store (use conpubd)
#CS_MODE=0
CS_MODE=2
…
```

追加



(4-2) キャッシュ利用設定

③ cefore-conf.producer/csmgrd.conf でキャッシュ挙動変更（任意）

cefore-conf.producer/csmgrd.conf

```
…
#
#
# Type of CS space used by csmgrd.
#  filesystem : UNIX filesystem
#  memory     : Memory
#
#CACHE_TYPE=filesystem
CACHE_TYPE=memory

#
# Type of cache policy by cache plugin.
#
#CACHE_ALGORITHM=libcsmgrd_lru
CACHE_ALGORITHM=libcsmgrd_fifo
…
```

追加

追加

- filesystem:
  - ファイルにキャッシュデータを保存
  - CACHE_PATHパラメータで キャッシュディレクトリを変更可
- memory:
  - メモリ上にキャッシュデータを保存

- libcsmgrd_fifo: First-In First-Out
- libcsmgrd_lru: Least Recently Used
- libcsmgrd_lfu: Least Frequently Used
- None: 置換を行わない （一杯になったらキャッシュを停止）

④ 4_reflect-conf.bash を実行して設定をコンテナに反映

※ コンテナにログインしている場合はホストに戻る

```
cefore:~/2021-hands-on/practice-B$ ./4_reflect-conf.bash
[SUCCESS] Completed.
```

(4-2) 設定確認 (b-router)

⑤ b-router にログインして cefnetd を起動、cefstatus を実行

```
cefore:~/2021-hands-on/practice-B$ ./login-router.bash
root@b-router:/cefore# cefnetdstart
root@b-router:/cefore# cefstatus
Version    : 1
Port       : 9896
Rx Frames  : 0
Cache Mode : Localcache
  faceid =   4 : IPv4 Listen face (udp)
  faceid =   0 : Local face
  faceid =   5 : IPv6 Listen face (udp)
  faceid =  16 : address = 10.0.1.102:9896 (udp)
  faceid =   7 : IPv6 Listen face (tcp)
FIB(App) :
FIB : 1
  ccnx:/test
    Faces : 16 (-s-)
  Entry is empty
PIT :
  Entry is empty
```

localcache を使用するモードに変更されている

Face と FIB エントリが追加されている

(4-2) 設定確認 (b-producer)

⑥ b-producer にログインして、 **csmgrd → cefnetd の順に起動**

```
cefore:~/2021-hands-on/practice-B$ ./login-producer.bash
root@b-producer:/cefore# csmgrdstart
root@b-producer:/cefore# cefnetdstart
```

cefnetd から csmgrd に接続するため、
必ず先に csmgrd を起動する
(起動していないと以下のようなエラーが出る)

```
root@b-producer:/cefore# cefnetdstart
2021-08-26 12:34:56.789 [cefnetd] ERROR: cef_csmgr_stat_create (connect to csmgrd)
2021-08-26 12:34:56.789 [cefnetd] ERROR: Failed to init Content Store
2021-08-26 12:34:56.789 [cefnetd] ERROR: Failed to create cefnetd handle
2021-08-26 12:34:56.789 [cefnetd] ERROR: Stop
```

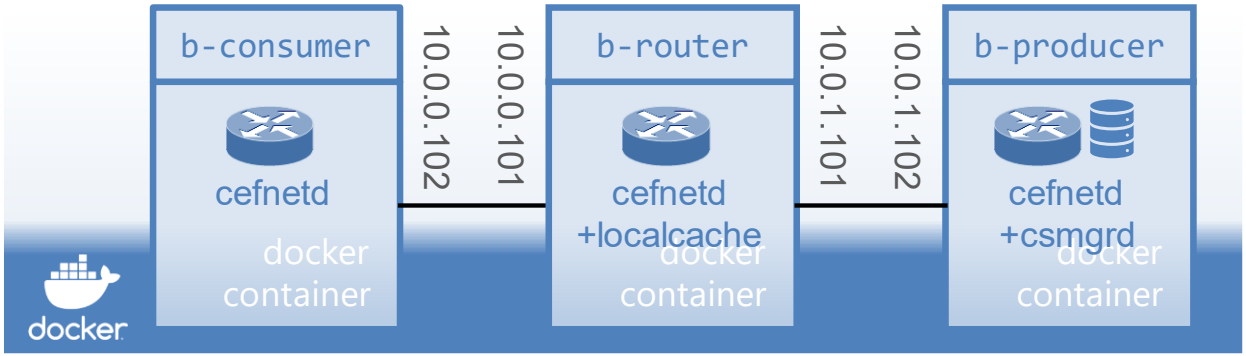
⑦ cefstatus で状態確認

```
root@b-producer:/cefore# cefstatus
Version    : 1
Port       : 9896
Rx Frames  : 0
Tx Frames  : 0
Cache Mode : Excache
Faces : 6
...
```

外部キャッシュ (csmgrd) を
使用するモードに変更されている

(5) ファイルのアップロードとダウンロード

ここまでの Practice で以下の CCN 環境が完成



実際にファイルのアップロードとダウンロードを試してみよう

(5-1) アップロード（キャッシュ利用）

- b-producer の csmgrd にファイルをアップロード（キャッシュ）する
- csmgrstatus でキャッシュされたファイルを確認する

(5-2) ダウンロード

- b-consumer から b-router 経由でファイルをダウンロードする
- ccninfo でキャッシュの所在を確認する

(5-1) アップロード（キャッシュ利用）

① b-producer にログインする

② ファイル hello.txt を作成する（内容は任意）

③ cefputfile で ccnx:/test/hello.txt を Data パケットとして送り出す

```
cefore:~/2021-hands-on/practice-B$ ./login-producer.bash
root@b-producer:/cefore# echo hello > hello.txt
root@b-producer:/cefore# cefputfile ccnx:/test/hello.txt
[cefputfile] Start
[cefputfile] Parsing parameters ... OK
[cefputfile] Init Cefore Client package ... OK
[cefputfile] Conversion from URI into Name ... OK
[cefputfile] Checking the input file ... OK
[cefputfile] Connect to cefnetd ... OK
[cefputfile] URI         = ccnx:/test/hello.txt
[cefputfile] File        = hello.txt
[cefputfile] Rate        = 5.000 Mbps
[cefputfile] Block Size  = 1024 Bytes
[cefputfile] Cache Time  = 300 sec
[cefputfile] Expiration  = 3600 sec
[cefputfile] Start creating Content Objects
[cefputfile] Unconnect to cefnetd ... OK
[cefputfile] Terminate
[cefputfile] Tx Frames   = 1
[cefputfile] Tx Bytes    = 6
[cefputfile] Duration    = 0.005 sec
[cefputfile] Throughput = 13150 bps
```

(5-1) アップロード（キャッシュ利用）

④ csmgrstatus でキャッシュ状況を確認する

```
root@b-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       : 1

[0]
  Content Name : ccnx:/test/hello.txt
  Content Size : 6 Bytes
  Access Count : 0
  Freshness    : 292 Sec
  Elapsed Time : 6 Sec
```

コンテンツ名
コンテンツサイズ
アクセス数
キャッシュが削除される期限
キャッシュされてからの経過時間

(5-2) ダウンロード

① b-consumer にログインする

② cefgetfile で ccnx:/test/hello.txt 宛に Interest パケットを送る

③ ファイルの内容を確認する

```
cefore:~/2021-hands-on/practice-B$ ./login-consumer.bash
root@b-consumer:/cefore# cefgetfile ccnx:/test/hello.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:/test/hello.txt
[cefgetfile] Start sending Interests
[cefgetfile] Complete
[cefgetfile] Unconnect to cefnetd ... OK
[cefgetfile] Terminate
[cefgetfile] Rx Frames = 1
[cefgetfile] Rx Bytes  = 6
[cefgetfile] Duration  = 0.000 sec
[cefgetfile] Jitter (Ave) = 0 us
[cefgetfile] Jitter (Max) = 0 us
[cefgetfile] Jitter (Var) = 0 us
root@b-consumer:/cefore# cat hello.txt
hello
```

ファイルがダウンロードできている

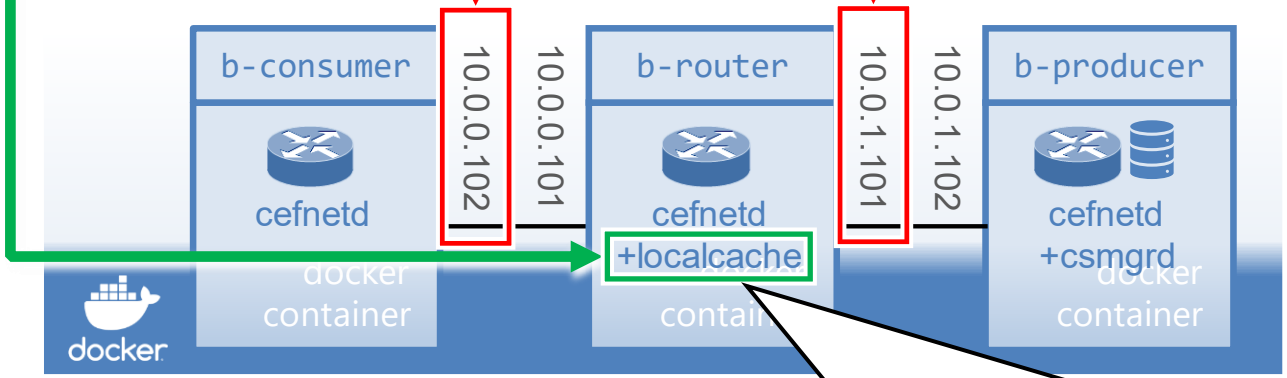
(5-2) ダウンロード

④ ccninfo でキャッシュの所在を確認する

```
root@b-consumer:/cefore# ccninfo ccnx:/test/hello.txt -c
ccninfo to ccnx:/test/hello.txt with HopLimit=32, SkipHopCount=0, Flag=0x0001, Request ID=51103 and node
ID=10.0.0.102

response from 10.0.1.101: NO_ERROR, time=0.514000 ms

route information:
 1 10.0.0.102: 0.214 ms
 2 10.0.1.101: 0.168 ms

cache information:   prefix                size    cobs    interests   start-end    cachetime    lifetime
 1 c                 ccnx:/test/hello.txt  0 KB    1       0           0-0          0 secs       0 secs
```

※出力結果整形済

経路情報（2ノードで発見）



localcache からキャッシュ情報を取得
（統計情報が無いため一部の情報が欠けている）

(5-2) ダウンロード

⑤ csmgrd の キャッシュ情報を ccninfo で確認する

```
root@b-consumer:/cefore# exit
cefore:~/2021-hands-on/practice-B$ ./producer.bash
root@b-producer:/cefore# ccninfo ccnx:/test/hello.txt -c
ccninfo to ccnx:/test/hello.txt with HopLimit=32, SkipHopCount=0, Flag=0x0001, Request ID=40361 and node
ID=10.0.1.102

response from 10.0.1.102: NO_ERROR, time=4.741000 ms

route information:
 1 10.0.1.102: 4.684 ms

cache information:   prefix               size    cobs    interests   start-end    cachetime    lifetime
 1 c                 ccnx:/test/hello.txt  0 KB    1       1           0-0          16 secs      281 secs
```

csmgrd は統計情報を保持しており、そこから応答では統計情報を確認できる
- prefix: コンテンツ名
- size: コンテンツサイズ
- cobs: チャンク数
- interests: 要求された数（先程の cefgetfile で 1 になっている）
- start-end: キャッシュされているチャンク番号の最初と最後
- cachetime: キャッシュされてからの経過時間
- lifetime: キャッシュから削除されるまでの残り時間

補足：Practice 自動実行モード

- practice-B/bin/answers 下に解答データを用意
- 解答データを用いた自動実行スクリプトも提供
  - Practice 「(4) 設定ファイルの変更方法」の設定ファイル適用

```
cefore:~/2021-hands-on/practice-B$ ./4_reflect-con.bash -a
[INFO] Answer mode.
[SUCCESS] Completed.
```

  - Practice 「(5) ファイルのアップロードとダウンロード」の実行

```
cefore:~/2021-hands-on/practice-B$ ./5_file-upload-and-download.bash
[INFO] * SETUP PROCESS STARTS.
[INFO] (4-1)-(4) Start cefnetd at b-consumer.
[INFO] [b-consumer] cefnetdstart
[INFO] [b-consumer] cefstatus
Version    : 1
Port       : 9896
Rx Frames  : 0
Tx Frames  : 0
Cache Mode : None
...
[INFO] (4-2)-(5) Start cefnetd at b-router.
[INFO] [b-router] cefnetdstart
[INFO] [b-router] cefstatus
Version    : 1
...
... (略、cefnetd/csmgrd 起動からファイルのアップロード・ダウンロードまでを自動実行)
...
[SUCCESS] * DOWNLOAD PROCESS COMPLETED.
```

Practice Advanced

発展課題

- cefgetfile/cefputfileのオプションを確認しよう
  - ユーザマニュアル6.1節、6.2節
  - デフォルトでは5分しかキャッシュされないので、コンテンツ期限(expiry, eオプション)やキャッシュ時間(cache time, tオプション)を変えてみよう
    - cmsgrstatus ccnx:/ の Freshness や ccninfo の lifetime で変化を確認できる
  - 大きなサイズのファイルを作ってアップロード速度やダウンロード速度を計測してみよう
    - cefputfileはアップロードレートが調整可能(rオプション)
    - cefgetfileは取得パイプライン数が調整可能(sオプション)
- cefgetchunkで複数のチャンクから成るコンテンツの特定のチャンクだけ取得してみよう
  - ユーザマニュアル6.3節
- docker-compose.yml を変更して自分で好きなネットワーク・トポロジーを作り、CCN 網を構築して通信してみよう

国立研究開発法人
情報通信研究機構

cefpyco を用いた通信

Cefore を用いた通信 | cefpyco を用いた通信 | 付録：インストールマニュアル

Practice 手順

① cefnetd への接続
- create_handle()メソッド

② Data パケットの送信
- send_data(name, payload , chunk_num)メソッド

③ Interest パケットの送信
- send_interest(name, chunk_num)メソッド

④ パケットの受信
- receive()メソッド

⑤ 簡易 Consumer アプリの作成

⑥ 簡易 Producer アプリの作成
- register(name)メソッド

⑦ サンプルアプリ CefApp の紹介

Practice

①cefnetdへの接続

1. practice-B/bin/cefpyco 下にファイル test1.py を作成

practice-B/bin/cefpyco/test1.py

```
#!/usr/bin/env python3

import cefpyco

with cefpyco.create_handle() as handle:
    pass # ブロック開始時にcefnetdへ接続、終了時に切断
```

2. b-producer にログインし、 cefnetd を起動して実行
（エラーが無ければ正常）

```
cefore:~/2021-hands-on/practice-B$ ./login-producer.bash
root@b-producer:/cefore# cefnetdstart
root@b-producer:/cefore# cd bin/cefpyco
root@b-producer:/cefore/bin/cefpyco# python3 test1.py
[cefpyco] Config directory is /usr/local/cefore
```

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.")
```

②Dataパケットの送信

1. python ファイル test2.py の内容を確認

practice-B/bin/cefpyco/test2.py

```
#!/usr/bin/env python3

import cefpyco

with cefpyco.create_handle() as handle:
    # ccnx:/testというコンテンツ名・チャンク番号0で
    # helloというテキストコンテンツをDataパケットとして送信
    handle.send_data("ccnx:/test", "hello", 0, cache_time=7200000)
```

2. csmgrd・cefnetdを起動して実行、動作確認

```
root@b-producer:/cefore/bin/cefpyco# csmgrdstart
root@b-producer:/cefore/bin/cefpyco# cefnetdstart
root@b-producer:/cefore/bin/cefpyco# python3 test2.py
[cefpyco] Configure directory is /usr/local/cefore
root@b-producer:/cefore/bin/cefpyco# csmgrstatus ccnx:/
...
*****   Cache Status Report         *****
Number of Cached Contents      : 1

[0]
  Content Name : ccnx:/test
  Content Size : 5 Bytes
  Access Count : 0
  Freshness    : 7194 Sec
  Elapsed Time : 4 Sec
```

csmgrdにDataがアップロードされる

③Interestパケットの送信

1. python ファイル test3.py を確認

practice-B/bin/cefpyco/test3.py

```
#!/usr/bin/env python3

import cefpyco

with cefpyco.create_handle() as handle:
    # ccnx:/testというコンテンツの0番目のチャンクを
    # 要求するInterestパケットを送信
    handle.send_interest("ccnx:/test", 0)
```

2. （test2.py実行後に）test3.pyを実行、動作確認

```
root@b-producer:/cefore/bin/cefpyco# python3 test3.py
[cefpyco] Configure directory is /usr/local/cefore
root@b-producer:/cefore/bin/cefpyco# csmgrstatus ccnx:/
...
*****   Cache Status Report         *****
Number of Cached Contents      : 1
[0]
  Content Name : ccnx:/test
  Content Size : 5 Bytes
  Access Count : 1
  Freshness    : 0168 Sec
  Elapsed Time : 1030 Sec
```

Access Count が 1 だけ増加

④パケットの受信

■ 以下のコードでパケットを受信可能

```
import cefpyco

with cefpyco.create_handle() as handle:
    handle.send_interest("ccnx:/test", 0) # Dataパケットを受信したい場合
    # handle.register("ccnx:/test") # Interestパケットを受信したい場合
    info = handle.receive()
```

■ receive()メソッド

- 実行後、約4秒の間 パケットを待ち受ける
  - 成功するまで受信したい場合はwhileループ等が必要
- CcnPacketInfoオブジェクトを返す
  - 受信の成否やInterest/Dataに依らない
  - プロパティ一覧は次ページで説明

CcnPacketInfo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）

⑤簡易Consumerアプリの作成

- 以下の内容のpythonファイルconsumer.pyを作成
  - チャンクが１個しかないコンテンツccnx:/testを受信
  - 受信に成功するまでreceiveメソッドを繰り返す
- **穴埋め問題：**
  - **受信したCcnPacketInfoを見て受信に成功したか否かを検証するif文を完成させよう**
  - 解答ファイルを practice-B/bin/answers/cefpyco 下に用意

practice-B/bin/cefpyco/consumer.py

```
#!/usr/bin/env python3

from time import sleep
import cefpyco

with cefpyco.create_handle() as handle:
    while True:
        handle.send_interest(                    )
        info = handle.receive()
        if info.is      and (info.name ==                ) and (info.chunk_num ==  ):
            print("Success")
            print(info)
            break
        sleep(1)
```

⑤簡易Consumerアプリの作成：解答

- 以下の内容のpythonファイルconsumer.pyを作成
  - チャンクが１個しかないコンテンツccnx:/testを受信
  - 受信に成功するまでreceiveメソッドを繰り返す
- **穴埋め問題：**
  - **受信したCcnPacketInfoを見て受信に成功したか否かを検証するif文を完成させよう**
  - 解答ファイルを practice-B/bin/answers/cefpyco 下に用意

practice-B/bin/cefpyco/consumer.py

```
#!/usr/bin/env python3

from time import sleep
import cefpyco

with cefpyco.create_handle() as handle:
    while True:
        handle.send_interest("ccnx:/test", 0)
        info = handle.receive()
        if info.is_data and (info.name == "ccnx:/test") and (info.chunk_num == 0):
            print("Success")
            print(info)
            break
        sleep(1)
```

consumer.pyの動作例

```
root@b-producer:/cefore/bin/cefpyco# csmgrdstart
root@b-producer:/cefore/bin/cefpyco# cefnetdstart
root@b-producer:/cefore/bin/cefpyco# echo hello > test
root@b-producer:/cefore/bin/cefpyco# cefputfile ccnx:/test
[cefputfile] Start
...
[cefputfile] Throughput = 11819 bps
root@b-producer:/cefore/bin/cefpyco# python3 consumer.py
[cefpyco] Configure directory is /usr/local/cefore
Success
Info: Succeeded in receiving Data packet with name 'ccnx:/test' (#chunk: 0) and payload
'b'hello¥n'' (6 Bytes)
```

ccnx:/testという名前の
コンテンツの0番目のチャンクの
取得に成功

⑥簡易Producerアプリの作成

- 以下の内容のpythonファイルproducer.pyを作成
  - **Interestを受信するにはregisterメソッドを用いる**
    - register(prefix): cefnetdがInterestを受信した際に、Interestの名前とprefixが一致する場合はアプリに送るように登録するためのメソッド
- **穴埋め問題：**
  - **"ccnx:/test"宛のInterestを受け取ったら"hello"という文字列を返すifブロックを完成させよう**
  - 解答ファイルを practice-B/bin/answers/cefpyco 下に用意

practice-B/bin/cefpyco/consumer.py

```
#!/usr/bin/env python3

import cefpyco

with cefpyco.create_handle() as handle:
    handle.register("ccnx:/test")
    while True:
        info = [redacted]
        if info.is_[redacted] and (info.name == [redacted]) and (info.[redacted]):
            handle.send_[redacted])
            # break # Uncomment if publisher provides content once
```

⑥簡易Producerアプリの作成：解答

- 以下の内容のpythonファイルproducer.pyを作成
  - **Interestを受信するにはregisterメソッドを用いる**
    - register(prefix): cefnetdがInterestを受信した際に、Interestの名前とprefixが一致する場合はアプリに送るように登録するためのメソッド
- **穴埋め問題：**
  - **"ccnx:/test"宛のInterestを受け取ったら"hello"という文字列を返すifブロックを完成させよう**
  - 解答ファイルを practice-B/bin/answers/cefpyco 下に用意

practice-B/bin/cefpyco/consumer.py

```
#!/usr/bin/env python3

import cefpyco

with cefpyco.create_handle() as handle:
    handle.register("ccnx:/test")
    while True:
        info = handle.receive()
        if info.is_interest and (info.name == "ccnx:/test") and (info.chunk_num == 0):
            handle.send_data("ccnx:/test", "hello", 0)
            # break # Uncomment if publisher provides content once
```

producer.pyの動作例

- b-producer の csmgrd.conf を無効化（CS_MODE=0に変更）
- 2つの端末を立ち上げて以下を実行

端末 1

```
cefore:~/2021-hands-on/practice-B$ ./login-producer.bash
root@b-producer:/cefore# cd bin/cefpyco
root@b-producer:/cefore/bin/cefpyco# cefnetdstart
root@b-producer:/cefore/bin/cefpyco# python3 producer.py
[cefpyco] Configure directory is /usr/local/cefore
```

コンテンツ配布開始

端末 2

```
cefore:~/2021-hands-on/practice-B$ ./login-producer.bash
root@b-producer:/cefore# cd bin/cefpyco
root@b-producer:/cefore/bin/cefpyco# python3 consumer.py
[cefpyco] Configure directory is /usr/local/cefore
Success
Info: Succeeded in receiving Data packet with name 'ccnx:/test' (#chunk: 0) and payload
'b'hello'' (5 Bytes)
```

csmgrd無しで取得に成功

トラブルシューティング

- ProducerがInterestを受け取れない
  - cefnetdのバッファやcsmgrdのキャッシュにヒットしている可能性がある
    - /usr/local/cefore/cefnetd.confで"CS_MODE=0"としてcsmgrdを停止する
    - cefnetdの場合はしばらく待つ

Advanced

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。

サンプルアプリ CefApp

- cefpyco/cefappディレクトリ内の２つのアプリ
  - cefappconsumer.py: Consumerアプリ
  - cefappproducer.py: Producerアプリ

- 特徴
  - 先ほどの簡易アプリと異なり、複数チャンクから成るコンテンツの送受信に対応
  - 入出力はインライン・標準入出力・ファイルの３種類
  - cefappconsumerはパイプライン処理を実装

- 詳細な使用法はREADME 5章参照

CefApp動作例

端末１（producer）

```
root@b-producer:/cefore# cd /cefore/cefpyco/cefapp
root@b-producer:/cefore/cefpyco/cefapp# ./cefappproducer.py ccnx:/test hello
[cefpyco] Configure directory is /usr/local/cefore
[cefapp] Receiving Interest...

[cefapp] Receive request for meta info
[cefapp] Wait for Interest...(1/2)
[cefapp] Wait for Interest...(2/2)
[cefapp] Succeed to send.
```

コンテンツ配布開始

端末２（consumer）

```
root@b-producer:/cefore# cd /cefore/cefpyco/cefapp
root@b-producer:/cefore/cefpyco/cefapp#./cefappconsumer.py ccnx:/test
[cefpyco] Configure directory is /usr/local/cefore
[cefapp] Succeed to receive.
hello
```

コンテンツ取得に成功

Practice Advanced

発展課題

- 複数チャンクの送受信を試してみよう
  - send_interestやsend_dataをチャンク数に応じて複数回実行するしてみよう
  - end_chunk_numを指定して取得すべきチャンク数だけ要求を出すようにしてみよう
- cefgetfile/cefputfileと同等の機能を作ってみよう
  - ファイルを読み書きして送受信できるようにしよう
  - 大きなファイルに対応するためのパイプライン処理を作ってみよう
    - CefAppのコード等を参考
- ceforeのツールやcefpyco/c_src/cefpyco.cを参考に、C言語でアプリを作ってみよう

国立研究開発法人
情報通信研究機構

付録：インストールマニュアル

Docker を使用せずに手動で環境構築を行う方法

Cefore を用いた通信 → cefpyco を用いた通信 → **付録：インストールマニュアル**

Cefore のインストール

① ソースコードとマニュアルのダウンロード

② ビルドとインストール

③ インストールされる機能について

- デフォルトでインストールされる機能
- インストール時にオプション指定が必要な機能

④ バッファチューニング

①ダウンロード (cefore.net の場合)

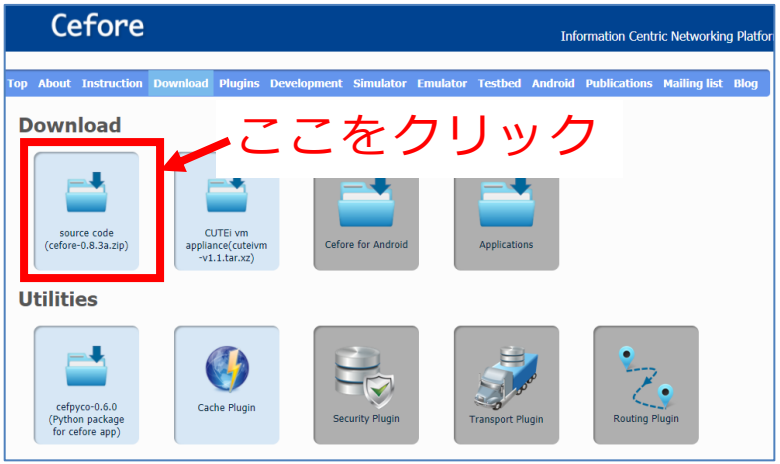
https://cefore.net/から
ソースコードとユーザマニュアルをダウンロード

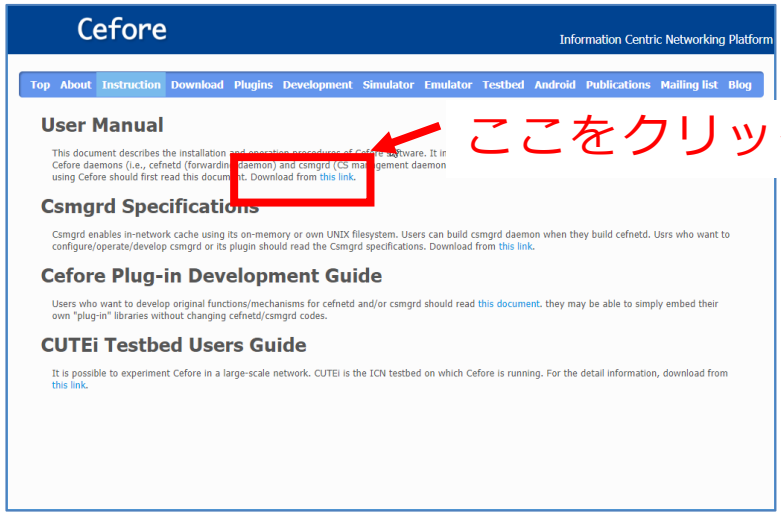
■ソースコード

- Download
  > source code
  (cefore-0.8.3a.zip)

■ユーザマニュアル

- Instruction
  > User Manual





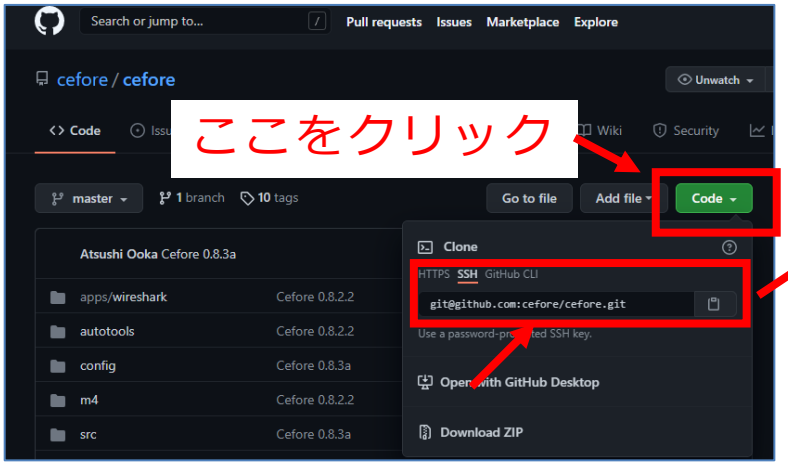
①ダウンロード (github.com の場合)

https://github.com/cefore/cefore から ソースコードをクローン

■ソースコード

- Code
  > HTTPS または SSH の アドレスをコピー

■自身の環境で git clone




```
$ git clone git@github.com:cefore/cefore.git
```

ここをコピー

②ビルドとインストール(Ubuntu)

■ライブラリのインストール

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

■Ceforeのビルド

```
$ unzip cefore-0.8.3a.zip                          # 任意のディレクトリでアーカイブを解凍
$ cd cefore-0.8.3a
$ autoconf
$ automake
$ ./configure --enable-csmgr --enable-cache        # csmgr、ローカルキャッシュを有効化
$ make
$ sudo make install                                # /usr/local/bin, sbin にインストールされる
$ sudo ldconfig                                    #（必要に応じて実行）
```

②ビルドとインストール(MacOS)

■ ライブラリのインストール（homebrew使用時）

```
$ brew install openssl automake
```

■ Ceforeのビルド

```
$ unzip cefore-0.8.3a.zip                    # 任意のディレクトリにアーカイブを解凍
$ cd cefore-0.8.3a
$ autoconf
$ automake
$ export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin:$PATH"
$ ./configure --enable-csmgr --enable-cache
  opssl_header_path=/usr/local/opt/openssl/include/
  LDFLAGS='-L/usr/local/opt/openssl/lib'
  CPPFLAGS='-I/usr/local/opt/openssl/include/'
$ make
$ sudo make install
```

一行で入力して実行
※長いので打ち間違いに注意
（macportsの場合は
"/usr/local/opt/openssl"を
"/opt/local"に置き換える）

■ ~/.bash_profileに以下を追加

```
export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin:$PATH"
```

トラブルシューティング: make

- autoconfに失敗する
  - →aclocalを実行する
    - brewでaclocalが失敗する場合は"brew doctor; brew brune"を試す
- Mac
  - cefctrlやcsmgrdが見つからないというエラーが出る
    - →PATHに/usr/local/sbinが入っているか確認する
  - configureに失敗する
    - →オプションを打ち間違いしていないか確認する
      - × openssl_header_path ○opssl_header_path
      - × LDFLAGS=`-L…` (バッククオート)
      - ○ LDFLAGS='-L…' （シングルクオート）
    - configureのオプションを.bash_profileに追加すれば毎回の入力を省略可能
      - .bash_profile に以下を書き込む（要端末再起動）

```
$ export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin/:$PATH"
$ export LDFLAGS="-L/usr/local/opt/openssl/lib"
$ export CPPFLAGS="-I/usr/local/opt/openssl/include"
$ export opssl_header_path="/usr/local/opt/openssl/include"
```

③インストールされる機能（デフォルト）

機能	形態	説明
cefnetd	daemon	フォワーディングデーモン
cefnetdstart	utility	フォワーディングデーモン起動ユーティリティ
cefnetdstop	utility	フォワーディングデーモン停止ユーティリティ
cefstatus	utility	cefnetdのstatus標準出力ユーティリティ
cefroute	utility	FIB操作ユーティリティ
cefputfile	tool	任意のファイルを name 付きコンテンツに変換しcefnetdへ入力する
cefgetfile	tool	cefnetdを介して取得したコンテンツをファイルとして出力する
cefgetchunk	tool	指定された name 付きコンテンツを取得し、ペイロードを標準出力する
cefputstream	tool	標準入力を name 付きコンテンツに変換しcefnetdへ入力する
cefgetstream	tool	cefnetdを介して取得したコンテンツを標準出力する

③インストールされる機能（要オプション）

■configure実行時にオプション指定が必要な機能

- 例: csmgrdとローカルキャッシュを有効化する場合
  - `./configure --enable-csmgr --enable-cache`
- `configure`変更後は`make`を再実行

機能	形態	option	説明
（cefnetdローカルキャッシュ）	function	`--enable-cache`	cefnetdのローカルキャッシュを有効化(CS_MODE=1)
csmgrd	daemon	`--enable-csmgr`	コンテンツストア管理デーモン (CS_MODE=2)
csmgrdstart	utility	`--enable-csmgr`	csmgrd起動ユーティリティ
csmgrdstop	utility	`--enable-csmgr`	csmgrd停止ユーティリティ
csmgrstatus	utility	`--enable-csmgr`	csmgrdのstatus標準出力ユーティリティ
csmgrecho	tool	`--enable-csmgr`	csmgr接続確認ツール
cefping	tool	`--enable-cefping`	ネットワーク管理ツールcefping
Sample Transport	plugin	`--enable-samptp`	Sample Transport プラグイン
NDN Plugin	plugin	`--enable-ndn`	NDNプラグイン

インストールディレクトリの指定方法

- 環境変数"$CEFORE_DIR"でインストール先を指定可能
  - "$CEFORE_DIR"のデフォルトは"/usr/local"
  - daemon機能は"$CEFORE_DIR/sbin"
  - utilityとtool機能は"$CEFORE_DIR/bin"
  - 設定ファイルは"$CEFORE_DIR/cefore"

- インストールディレクトリを変更した場合は、configure実行前にautoconfとautomakeを再実行

④バッファチューニング

■ Linux OS

```
$ sudo sysctl -w net.core.rmem_default=10000000
$ sudo sysctl -w net.core.wmem_default=10000000
$ sudo sysctl -w net.core.rmem_max=10000000
$ sudo sysctl -w net.core.wmem_max=10000000
```

■ Mac OS

```
$ sudo sysctl -w net.local.stream.sendspace=2000000
$ sudo sysctl -w net.local.stream.recvspace=2000000
```

■ PC再起動時にパラメータが初期化されるので、
再実行しやすいようスクリプト化するのを推奨

トラブルシューティング: csmgrd

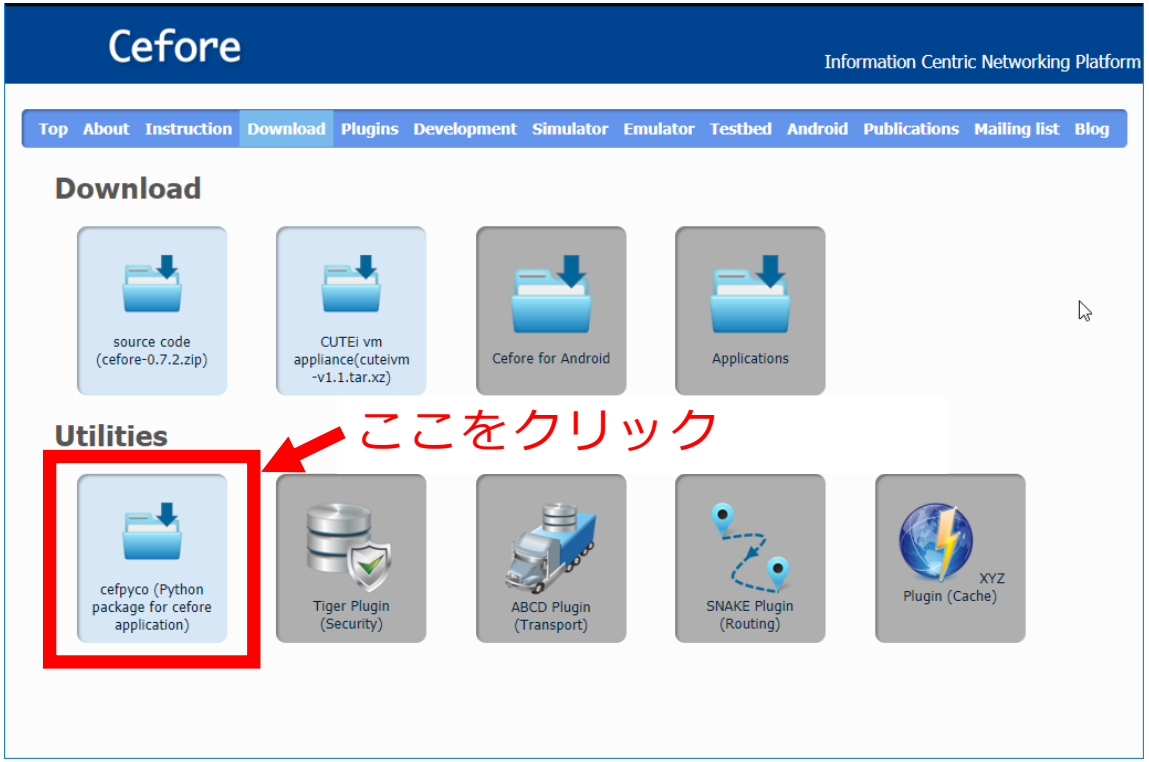
- csmgrdstartコマンドが見つからない。
  - →configure実行時に"--enable-csmgr"を付けているか確認する。
    - オプション指定にミスがあると無視されるので、打ち間違いに要注意。
- configureのオプション指定を変更すると、makeに失敗する。
  - "make clean"を実行してからmakeをやり直す。
- csmgrdstart実行時に" [csmgrd] ERROR: libcsmgrd_plugin.so: cannot open shared object file: No such file or directory"と表示される。
  - →Ubuntuの場合、"sudo ldconfig"を実行する。
  - →MacOSの場合、以下を実行する（~/.bash_profileにも要追記）。
    - export PATH="/usr/local/sbin:/usr/local/opt/openssl/bin:$PATH"
- cefnetd・csmgrdが起動しない
  - "sudo cefnetdstop -F" を実行してから再起動する。
  - 「インストール④バッファチューニング」を行ったかどうか確認する。
    - バッファチューニングはPCを再起動すると設定が初期化されるので要注意。

cefpyco のインストール

- ダウンロード
- ライブラリのインストール
- ビルドとインストール

cefpycoのダウンロード

- https://cefore.net/download > Utilities > cefpyco
  - マニュアル（README.html）付属*1



*1: 2018/8/30現在は日本語マニュアルのみ

ライブラリのインストール

■ライブラリのインストール（Ubuntu）

- python3 環境を推奨

```
$ sudo apt-get install cmake python-pip3
$ sudo pip3 install setuptools click numpy
```

■ライブラリのインストール（Mac）

```
$ brew install cmake pyenv
$ pyenv install 3.7.3       # 任意のバージョンを指定
$ pyenv global 3.7.3        # pyenv local … なら実行ディレクトリでのみ有効化
$ eval "$(pyenv init -)"
$ pyenv versions            # インストール version の確認
$ sudo easy_install pip     # macports では py-pip
$ python –m pip install setuptools click numpy
$ echo 'eval "$(pyenv init -)"' >> ~/.bash_profile  # 任意
```

ビルドとインストール

■cefpycoのビルド

```
$ unzip cefpyco-0.6.2.zip        # 任意のディレクトリにアーカイブを解凍
$ cd cefpyco
$ cmake .                        # "."のつけ忘れに注意
$ sudo make install
$ cd
$ sudo python3                   # pythonでインポートして起動確認
>> import cefpyco                # エラーが起きなければ成功
>> (Ctrl-D)                      # Ctrl-D を押して終了
```

■作成したアプリの実行にも sudo が必要なので要注意