国立研究開発法人
情報通信研究機構

第9回ICN研究会ワークショップ

Ceforeチュートリアル

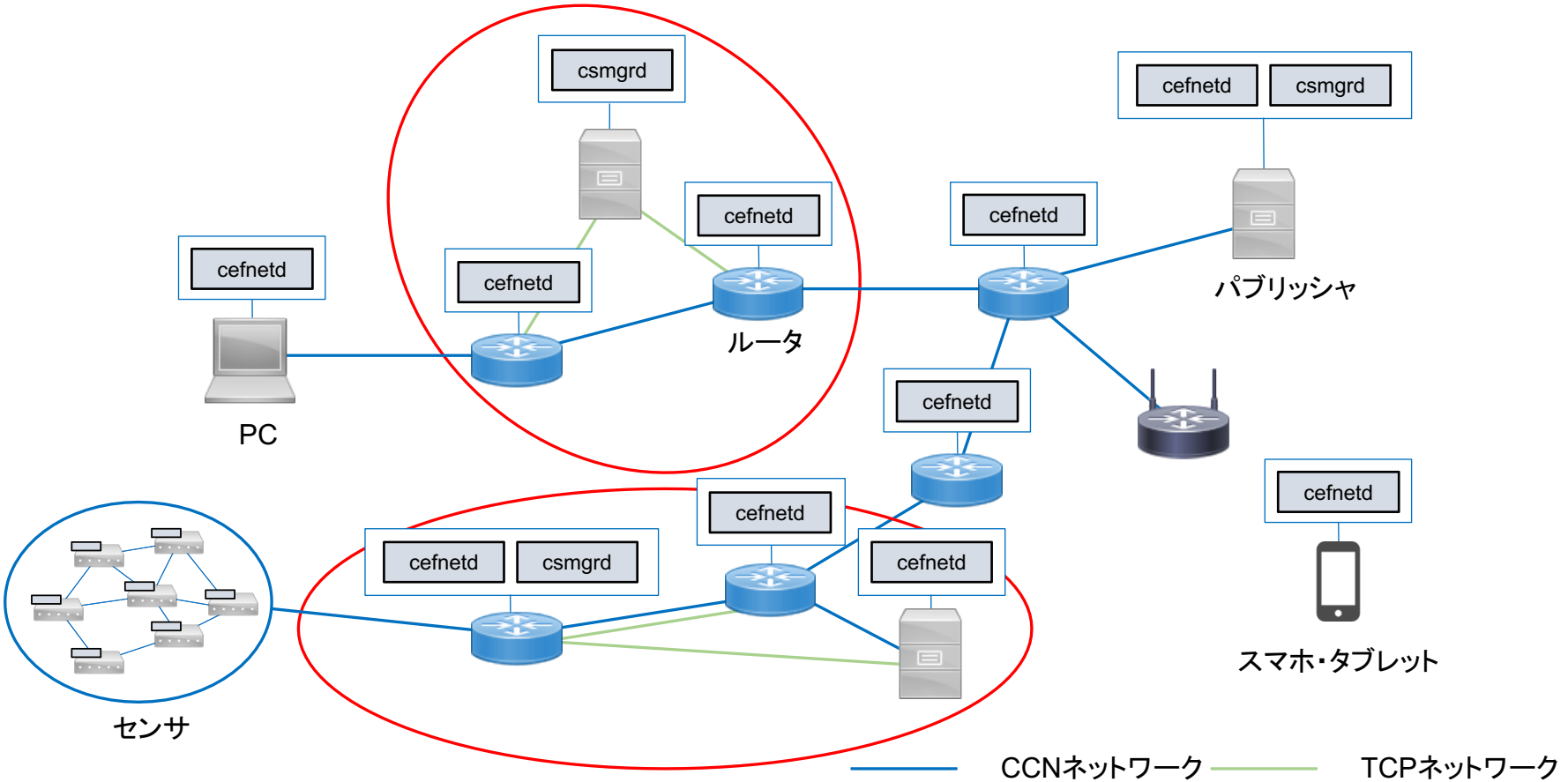
2017年8月25日（金）

Cefore

- 軽量かつ汎用的なCCNベースソフトウェアプラットフォーム
  - 言語：C、OS：Linux（ubuntu 14.04 or 16.04）、MacOS、Raspbian、 Android
  - 最小機能構成（cefnetd（後述））
    - フォワーディング機能と基本オペレーションツールのみを具備
      - (1) FIB、(2) PIT、(3) InterestおよびObject転送
      - Content Storeは具備していない
  - ユースケースに合わせた機能構成の実現
    - リソースの乏しいセンサーノードでは最小機能構成で起動
    - 最小機能以外はPluginもしくは外部機能として実装
    - Plugin実装として柔軟かつ自由に機能拡張を組み込み
      - キャッシュ、モビリティ、トランスポート、NDNパケット転送
      - 使用しない機能はビルドしない
    - 外部機能実装
      - csmgrd （後述）とのTCPソケット通信

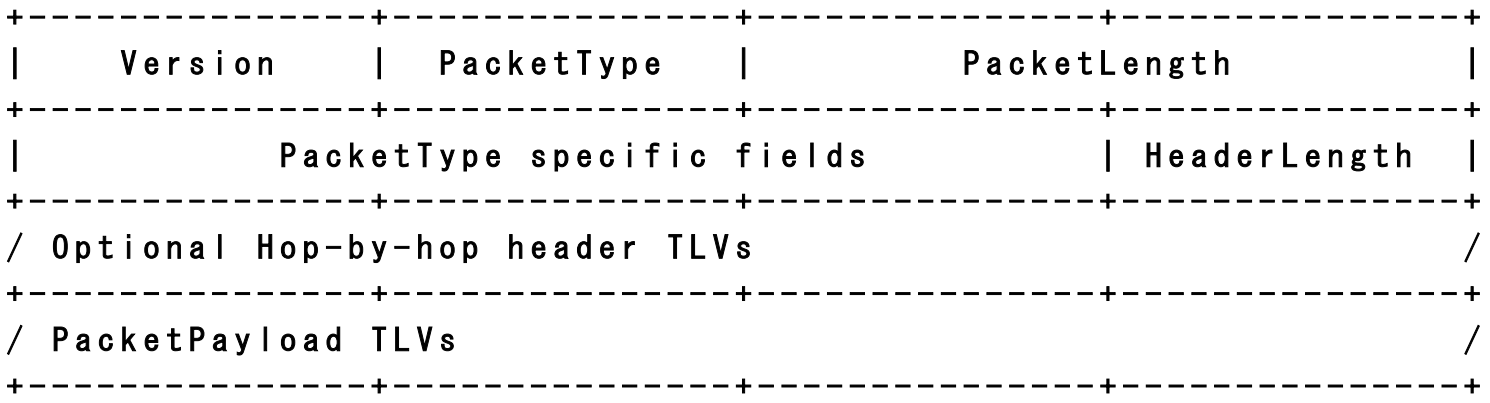
Ceforeの機能構成

- フォワーディングデーモン（cefnetd）は全てのノードで稼働
- コンテンツストア管理デーモン（csmgrd）は、コンテンツストアが必要な場合のみ稼動させる
  - 下図の例では赤枠内のcefnetdでコンテンツストアを共有する



Ceforeヘッダー＆パケットTLV

- CCNx-1.0のパケットフォーマットに準拠
  - CCNx-1.0のヘッダー構成
    - 8オクテットの固定ヘッダー
    - 最大247バイトのOptional Hop-by-hopヘッダー
  - Optional Hop-by-hopヘッダー
    - ホップ毎に処理可能なオプション領域
    - Cefore独自のプロトコル拡張はこのオプションヘッダーに配置



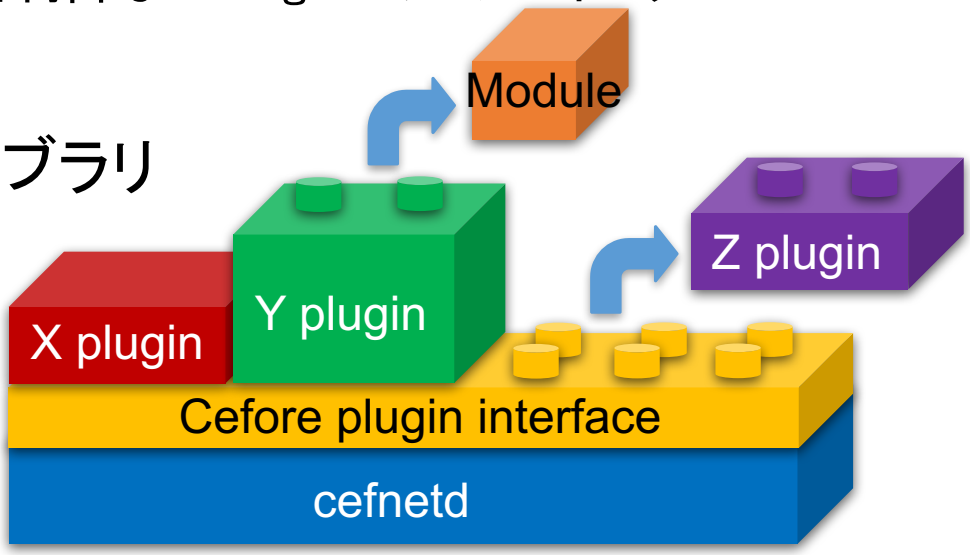
CCNx Messages Format

[出典] CCNx Messages in TLV Format (draft-irtf-icnrg-ccnxmessages-04)

cefnetd / csmgrd

cefnetd

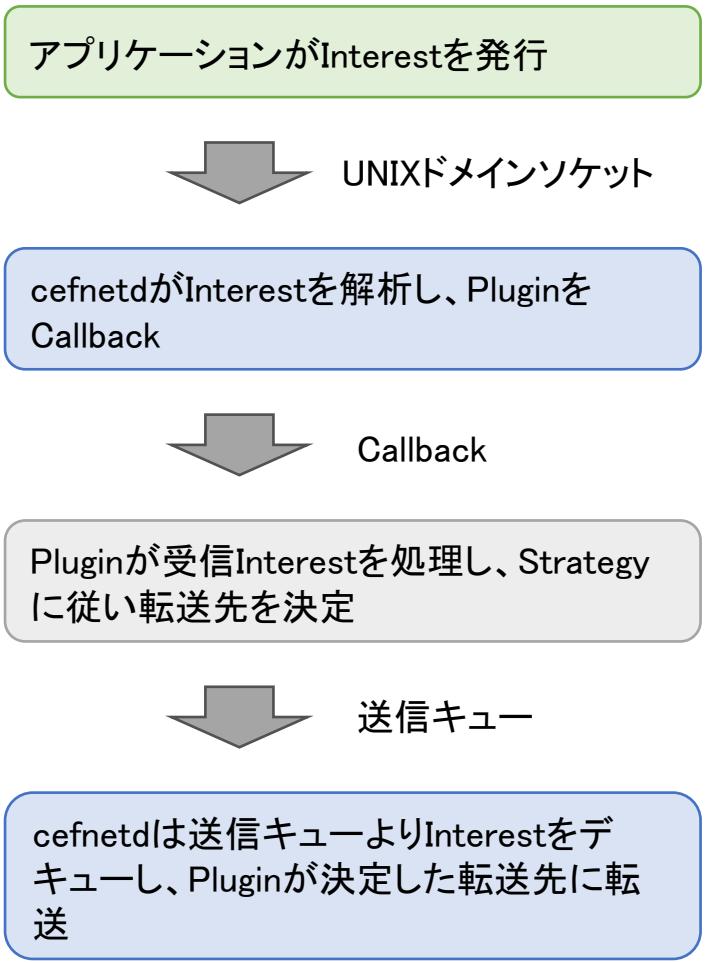
- Ceforeの土台となるシンプルなフォワーディングデーモンであり、全ノード(送受信者・ルーター)に組み込む
- 実装機能
  - フォワーディング機能
  - Staticルーティング機能
  - Plugin interface
    - キャッシュ、モビリティ、経路制御などPluginのためのインタフェース
  - Security機能(未定)
- コンパイルに必要となるライブラリ
  - OpenSSL(ver.1.0.2l)



cefnetd – Configuration

- cefnetdの設定ファイルは以下の3種類
  - cefnetd.conf ・・・ cefnetdに関する各種パラメータ設定
  - cefnetd.fib ・・・ cefnetd起動時のFIB
    - FIBはcefnetd起動後にcefrouteユーティリティでも操作可能
  - plugin.conf ・・・ Cefore Plugin全般に関するパラメーター設定
- 設定ファイルは以下のディレクトリに配置する
  - 非スーパーユーザ
    - “$CEFORE_USER_DIR/cefore”
    - 環境変数“CEFORE_USER_DIR”のデフォルトは“$HOME”
  - スーパーユーザ
    - “$CEFORE_DIR/cefore”
    - 環境変数“CEFORE_DIR”のデフォルトは“/usr/local”
  - 各Plugin固有の設定ファイルは、上記ディレクトリ配下のpluginディレクトリに配置
- 設定ファイルの行頭が“#”の行は、コメント行として扱われる

通信概要 – Interest送信



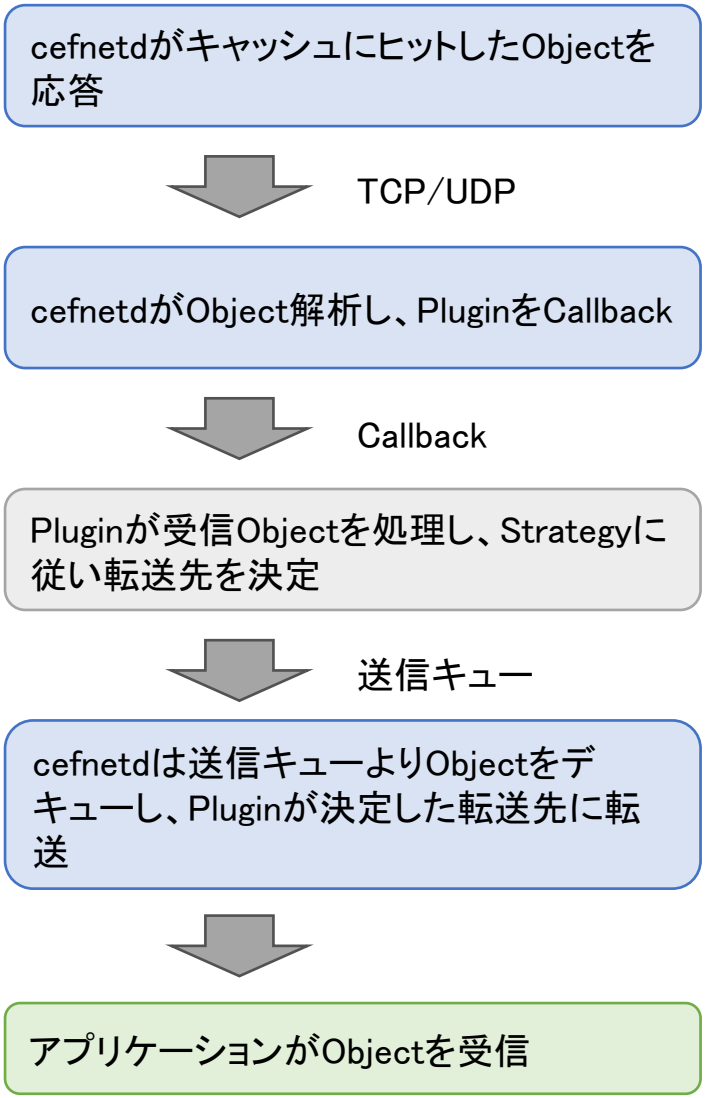
- Interestを発行し、UNIXドメインソケットを介してcefnetdへ
- 特定のTransport機能を使用する場合、アプリケーション種別に応じたTransportを選択

- 受信したInterestのOptional Hop-by-hopヘッダーを解析
- PIT更新
- 指定されたPluginをサポートしている場合はCallback（Plugin呼び出し）
- Callbackする際、FIBから検索した転送先をPluginに渡す

- Optional Hop-by-hopヘッダーの解析後、Nameに従いInterestを処理
- 転送が必要な場合、Strategyに従い転送先を決定
- 転送するInterestは送信キュー（共有メモリ）にインキュー

- 転送先はPluginのStrategyがFIBの検索結果から決定

通信概要 – Content Object送信



- PITに従いキャッシュヒットしたObjectを転送

- 受信したObjectのOptional Hop-by-hopヘッダーを解析
- 指定されたPluginをサポートしている場合はCallback
- Callbackする際、PITから検索した転送先をPluginに渡す

- Optional Hop-by-hopヘッダー解析後、Nameに従いObjectを処理
- 転送が必要な場合、Strategyに従い転送先を決定
- 転送するObjectは送信キュー（共有メモリ）にインキュー

- 転送先はPluginのStrategyがPITの検索結果から決定

FIB実装概要

- cefore-0.6.0/src/lib/cef_fib.cにFIBに関する処理を実装
- FIBはハッシュテーブル（CefT_Hash_Handle構造体)として実装
  - cefore-0.6.0/src/cefnetd/cef_netd.hのCefT_Netd_Handle構造体メンバfibでハッシュテーブルを保持
  - FIBに収容可能なFIBエントリ数は、cefnetd.confのFIB_SIZEを満たす素数となる
  - FIBエントリ（CefT_Fib_Entry構造体）は以下のデータを保持する
    - unsigned char*　　Name（検索キー）
    - unsigned int　　Name長（バイト）
    - CefT_Fib_Face　　転送先Faceリスト
  - Nameをハッシュ値に変換し、FIBエントリをハッシュテーブルで管理
    - ハッシュ操作APIはcef_hash.c/hに実装されている
      - ✓ ハッシュテーブル作成：cef_hash_tbl_create()
      - ✓ データの追加：cef_hash_tbl_item_set()
      - ✓ データの検索（アクセス）：cef_hash_tbl_item_get()
      - ✓ データの削除：cef_hash_tbl_item_remove()
- FIBに同じ上流ルータに対してTCPとUDP接続が両方存在する場合、Interestを受信したプロトコルと同じプロトコルを優先する
  - 同じプロトコルの転送先が1つも存在しない場合は、異なるプロトコルの全転送先に転送する
- Transport Pluginへは受信Interestと共にFIB検索結果(転送先Face一覧)を受け渡す
  - Transport Pluginはプロトコル仕様、Strategyに従い転送する

FIB実装（Advanced）

- FIBエントリの更新、アクセス
  - NameをキーにFIBエントリをLongest Matchで検索
    - cef_fib_entry_search()
  - FIBエントリを追加
    - cef_fib_entry_create()　　：FIBエントリの作成
    - cef_hash_tbl_item_set()　　：ハッシュテーブルに追加
  - FIBエントリに転送先Faceを追加
    - cef_fib_faceid_insert()
  - FIBエントリから転送先Faceを削除
    - cef_fib_faceid_remove()
  - FIBよりNameで指定したFIBエントリを削除
    - cef_fib_entry_destroy()
- FIBの拡張
  - FIBエントリ(CefT_Fib_Entry構造体)に状態変数を追加
  - プロトコル毎のFIBを作成
    - CefT_Netd_Handle構造体メンバにハッシュテーブルを追加
    - 上記FIB操作APIとハッシュ操作APIで難しい実装は不要

PIT実装概要

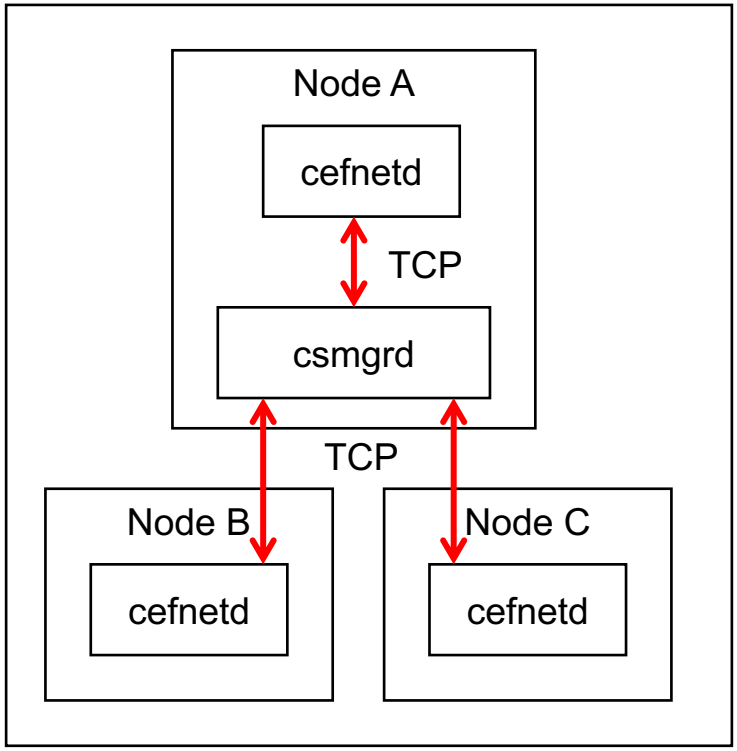
- cefore-0.6.0/src/lib/cef_pit.cにPITに関する処理を実装
- PITもFIB同様にハッシュテーブルとして実装
  - cefore-0.6.0/src/cefnetd/cef_netd.hのCefT_Netd_Handle構造体メンバpitでハッシュテーブルを保持
  - PITに収容可能なPITエントリ数は、cefnetd.confのPIT_SIZEを満たす素数となる
  - PITエントリ(CefT_Pit_Entry構造体)は以下のデータを保持する
    - unsigned char* Name (検索キー)
    - unsigned int Name長 (バイト)
    - CefT_Down_Faces Interestを受信したFaceリスト
Face毎の生存時間 (lifetimeなども管理)
    - CefT_Up_Faces Interestを転送したFaceリスト
  - PITエントリにFace毎の生存時間を設定しておけば、cefnetdが定期的にチェックしエントリを削除する
- Transport Pluginへは受信Content Objectと共にPIT検索結果(転送先Face一覧)を受け渡す
  - Transport Pluginはプロトコル仕様、Strategyに従い転送する

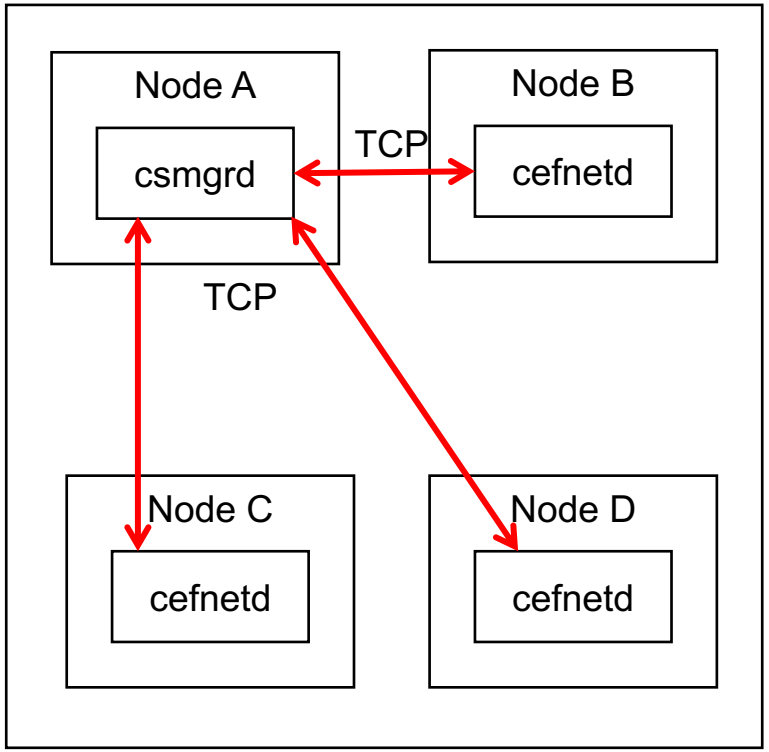
PIT実装（Advanced）

- PITエントリの更新、アクセス
  - NameをキーにPITエントリをExact Matchで検索
    - cef_pit_entry_lookup() / cef_pit_entry_search()
      - cef_pit_entry_lookup()は指定されたNameのPITエントリが未作成の場合は作成する
  - PITエントリにInterest受信Faceを追加
    - cef_pit_entry_down_face_update()
      - PITエントリの有効期限の再計算、受信Interestの転送の要否判定も実行する
      - PITの集約アルゴリズムの変更はこの関数を改変する
  - PITエントリにInterest転送先Faceを追加
    - cef_pit_entry_up_face_update()
  - PITエントリの削除
    - cef_pit_entry_free()

csmgrd

- cefnetdは外部機能デーモンであるCS管理デーモン（csmgrd）と連携
- cefnedとcmsgrd間はTCPにて接続
  - 1つのcsmgrdに対して複数のcefnetdの接続が可能
    - 設定ファイル（csmgrd.conf）にて接続可能なcefnetdを指定
  - リモートからのTCP接続を使用したcsmgrdの状態確認も可能





Cefore Plugin

Cefore Plugin

- 機能追加・拡張のためのPluginライブラリ
  - フォワーディング機能（cefnetd）に対する「拡張機能」の追加
  - CS機能（csmgrd）に対する「キャッシュ領域選択機能」及び「キャッシュアルゴリズム」の追加
- Pluginライブラリ連携用コールバック関数インターフェース
  - 所定のCallback関数を用いて、Pluginをcefnetd/csmgrdと組み合わせて使う
  - 必要なPluginを必要に応じて開発し、着脱も可能
    - 軽微なMakefileの変更とリコンパイルで機能追加可能
    - 追加した各機能はplugin.confにてON/OFF可能
  - 異なるPluginライブラリ間で機能拡張・追加の影響を与えない

Cefore Plugin – cont'd

- cefnetd用Plugin実装例
  - Sample transport
    - Plugin実装用のサンプルコード
  - NDN
    - NDNメッセージフォーマットのInterest/DataをNDN FIBに従い転送
- csmgrd用Plugin実装
  - Cache領域選択((標準で提供)filesystem, memory)
  - Cache algorithm
    - (標準で提供)LRU, LFU, FIFO
    - その他アルゴリズム拡張
- Plugin間の連携処理
  - Pluginの上にmodule/library(既存の外部実装など)を実装することも可能
  - Plugin間の共通(or 汎用)APIは規定していない(原則、cefnetd経由)

API概要

- Callback関数
  - 各PluginのCallback関数
    - Callback関数は、(1) 初期化処理、(2) Interest受信処理、(3) Object受信処理、(4) PIT変更、(5) 後処理、の5種類（次項）
    - 必要ない処理については実装不要
  - Callback関数のインタフェース(引数、戻り値)以外の制約は設けない
    - Plugin内でスレッド化
    - 独自のタイマ管理
    - メッセージの送出（InterestやObjectの再送信、レポートの送信）
    - 外部ライブラリ使用
    - 外部機能との通信
- その他
  - コンフィグファイル解析、パラメータ参照処理は共通API化
  - 送信バッファ操作、メモリプール操作、ログ操作なども共通API化

Callbackの種類

- Pluginで利用されるCallback
  - init
    - cefnetdの初期処理内で呼出すCallback。内部変数の初期化、メモリ確保およびスレッド生成などPlugin毎の初期化処理を実行する。
  - cob
    - cefnetdがObjectを受信した際に呼出すCallback。受信ObjectおよびPITの検索結果（転送先候補）を引数に呼出される。
  - interest
    - cefnetdがInterestを受信した際に呼出すCallback。受信InterestおよびFIBの検索結果（転送先候補）を引数に呼出される。
  - destroy
    - cefnetdの後処理内で呼出すCallback。プラグインの後処理（スレッドの回収、メモリ解放など）を実行する。
- 一部のPluginのみ利用されるCallback
  - pit
    - cefnetdがPITよりPITエントリを削除した際に呼出すCallback。cefnetdよりPITエントリの削除が発生した通知を受信したい場合は必要。

Callbackのインタフェース

- 各Callback関数のインタフェースを以下に示す。“CefT_Plugin_X“は、各プラグイン情報管理構造体である。
  - 構造体”CefT_Rx_Elem“は受信メッセージ、メッセージ解析情報、転送先候補Faceなど、構造体“CefT_Rx_Elem_Sig_DelPit“は削除されたPITエントリに関する情報が設定されている。

```
int (*init)(CefT_Plugin_X*);
int (*cob)(CefT_Plugin_X*, CefT_Rx_Elem*);
int (*interest)(CefT_Plugin_X*, CefT_Rx_Elem*);
void (*pit)(CefT_Plugin_X*, CefT_Rx_Elem_Sig_DelPit*);
void (*destroy)(CefT_Plugin_X*);
```

- Callback内で処理したメッセージを転送するには
  - Callback関数内で処理したメッセージを転送する場合、送信キュー（全Pluginからアクセス可能）にメッセージをインキューする。インキューしたメッセージはPluginが決定した転送先にメッセージを転送する。
  - 逆にメッセージを転送しない場合は送信キューにインキューしない。
- Callback内からメッセージの送出も可能
  - 例えば、Callback”cob”にてObjectのロストを検知した場合、Interestを生成し送信キューにインキューすることにより、ロスとしたObject取得を試みることができる。

Callbackの設定

- Developerが実装したCallbackはcefnetdがCallできるよう設定する必要がある
- cef_X_plugin.c（Transportなどプラグイン毎に共通処理をまとめたソースファイル）にある初期化関数にて、Callback関数を設定する

```
#ifdef CefC_Plugin_Samptp    ・・・  configureで切り分け
    lp = cef_plugin_parameter_value_get ( “TRANSPORT”,  “samptp” );  ・・・ コンフィグファイルのsamptp=yes/no（ON/OFF）

    if (lp) {      ・・・ パラメータが未設定の場合はNULL
        value_str = (char*) cef_plugin_list_access (lp, 0); ・・・ 設定値はリスト化された形で取得

        if (strcmp (value_str,  “yes” ) == 0) {    ・・・ OFFの場合、Callbackは初期値のNULLとなり、cefnetdよりCallされない

            work[CefC_T_OPT_TP_SAMPTP].variant    = CefC_T_OPT_TP_SAMPTP;          ・・・ Transport Variantを設定
            work[CefC_T_OPT_TP_SAMPTP].tx_que     = tx_que;                        ・・・ 送信キューの共有メモリ
            work[CefC_T_OPT_TP_SAMPTP].tx_que_mp  = tx_que_mp;                     ・・・ メモリープールハンドル
            work[CefC_T_OPT_TP_SAMPTP].init       = cef_plugin_samptp_init;        ・・・ 以下がCallbackの設定となる
            work[CefC_T_OPT_TP_SAMPTP].cob        = cef_plugin_samptp_cob;              関数名に制約はなく、引数、戻り値
            work[CefC_T_OPT_TP_SAMPTP].interest   = cef_plugin_samptp_interest;         のみ規定の型にすればよい
            work[CefC_T_OPT_TP_SAMPTP].pit        = cef_plugin_samptp_delpit;      ・・・ cefnetdからCallしない場合はNULL
            work[CefC_T_OPT_TP_SAMPTP].destroy    = cef_plugin_samptp_destroy;
        }
    }
#endif // CefC_Plugin_Samptp
```

Ceforeディレクトリと追加Plugin構成

```
cefore-x.x.x                                ・・・ Ceforeのルートディレクトリ
      |- configure.ac                        ・・・ 「1. オプションチェックの追加」を参照
      |- Makefile.am                         ・・・ 「1. configureオプションの追加」を参照
      |- src
      |   |- cefnetd
      |   |- include
      |   |   |- cefore
      |   |   |   |- cef_plugin.h
      |   |   |   |- cef_newplugin.h         ・・・ 新規に追加するPluginのヘッダーファイル
      |   |   |   +- Makefile.am             ・・・ 「2. ヘッダーファイルの追加」を参照
      |   |   +- Makefile.am
      |   |- lib
      |   |- Makefile.am
      |   +- plugin
      |       |- cef_plugin.c
      |       |- cef_X_plugin.c
      |       |- Makefile.am                 ・・・ 「2. ソースコードファイルの追加」を参照
      |       +- X                           ・・・ Transport、Cache algorithmなどのディレクトリ
      |           +- newplugin               ・・・ 新規に追加するPlugin用ディレクトリ
      |               +- cef_newplugin.c     ・・・ 新規に追加するPluginのソースコードファイル
      |- tools
      +- utils
```

Pluginの追加手順（1）

- オプションチェックの追加
  - cefore-x.x.x/configure.ac

```
# checks for arg
dnl
dnl  check newplugin
dnl
AC_ARG_ENABLE(
  newplugin,
  AS_HELP_STRING([--enable-newplugin], [new plugin (default off)]),
  [enable_newplugin=yes],
  [enable_newplugin=no]
)
AM_CONDITIONAL(NEWPLUGIN_ENABLE, test x"${enable_newplugin}" = xyes)
```

- configureオプションの追加
  - cefore-x.x.x/Makefile.am

```
# set distcheck configure flag
DISTCHECK_CONFIGURE_FLAGS=--enable-csmgr --enable-hoge … --enable-newplugin
```

Pluginの追加手順（2）

- ヘッダーファイルの追加
  - cefore-x.x.x/src/include/cefore/Makefile.am

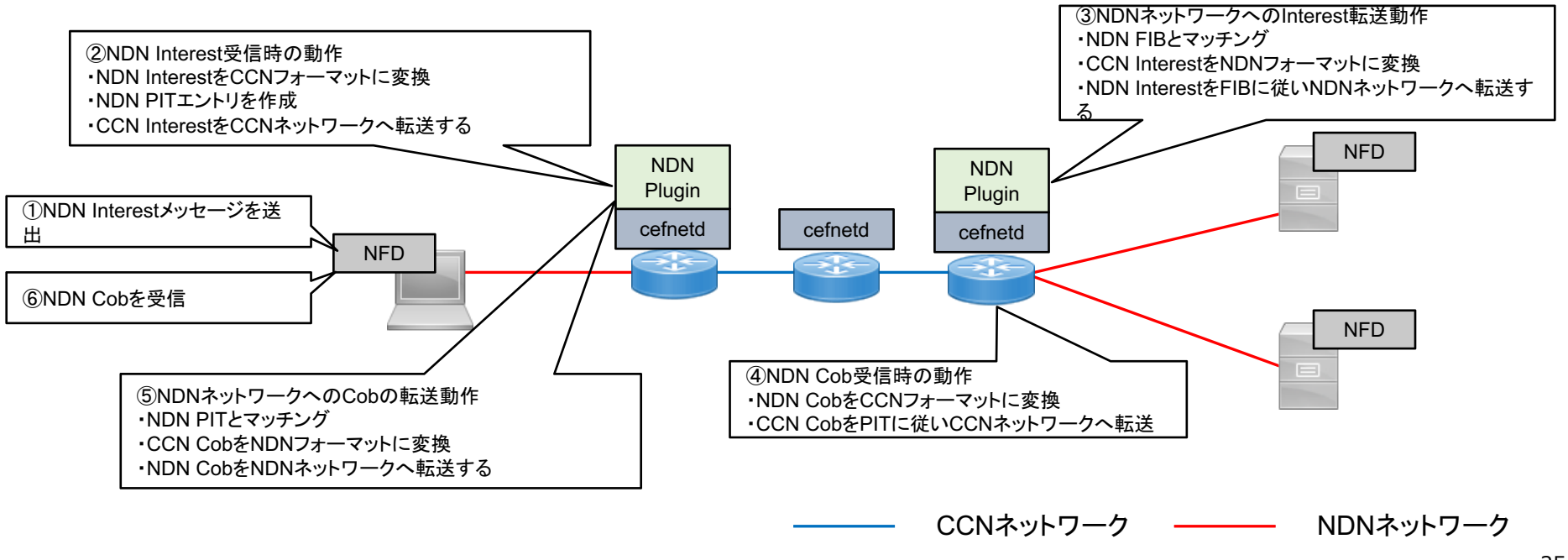
```
if NEWPLUGIN_ENABLE
CEF_HEADER+=cef_newplugin.h
endif # NEWPLUGIN_ENABLE
```

- ソースコードファイルの追加
  - 新規に追加するCallbackなどを記述したPluginのソースは、cefore-x.x.x/src/plugin/X/newpluginに追加
  - cefore-x.x.x/src/plugin/Makefile.amにビルド対象となるソースコードファイル、およびマクロを追加

```
if NEWPLUGIN_ENABLE
# AM_CFLAGSはソースコード、ヘッダーファイルのifdef/ifndefで使用
AM_CFLAGS+=-DCefC_Plugin_Newplugin
AM_CSOURCES+=X/newplugin/cef_newplugin.c
endif # NEWPLUGIN_ENABLE
```

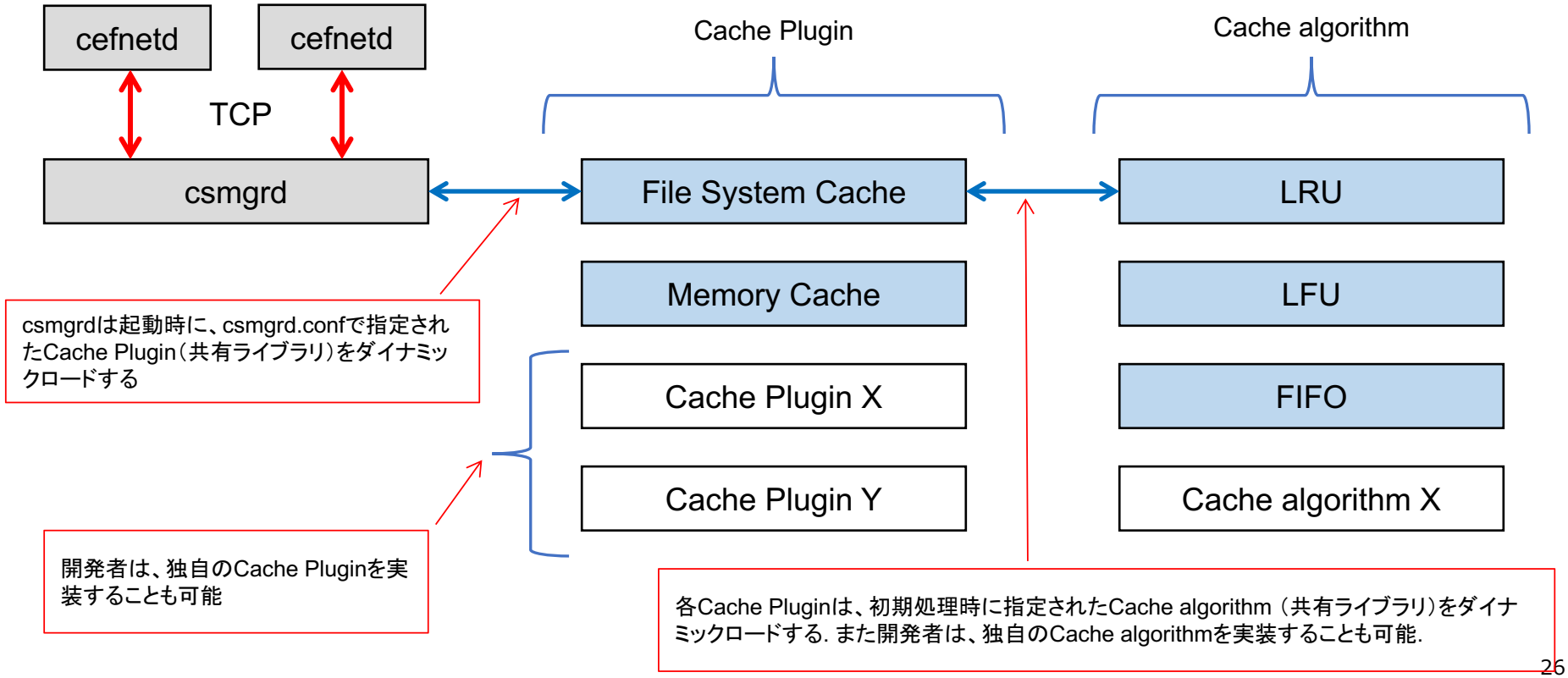
NDN Plugin

- NDNメッセージ⇔CeforeメッセージNDN変換するためのPlugin
- ゲートウェイ（CeforeとNDNの境界）ノードにてNDN Pluginを有効化
- CeforeネットワークよりNDNネットワークへメッセージを転送するためには、NDN FIBが必要
  - 現時点での実装ではCCN FIBおよびPITとのマッチングを優先し、CCN FIB/PITに転送先が存在しない場合のみNDN FIB/PITとマッチングする



Cache Plugin & Cache Algorithm

- csmgrdは起動時に使用するCache Pluginを指定
  - File System Cache（低速、大容量CS）、Memory Cache（高速、小容量CS）をpluginにて具備
- 各Cache Pluginは起動時に使用するCache algorithmを指定
  - LRU、LFU、FIFOを具備
  - ユーザー独自のCahce algorithmを実装可能



Tools / Command Utilities

Tools/Command Utilities

- コンテンツの配信、取得
  - Named Cobのアップロード・ダウンロード
    - cefputfile/cefgetfile
  - 特定のContent Objectのダウンロード
    - cefgetchunk
  - ストリーム配信・受信
    - cefputstream/cefgetstream
- ネットワーク管理ツール
  - コンテンツがキャッシュされているノードの特定
    - conping
  - コンテンツまでの経路、キャッシュされているコンテンツの詳細取得
    - contrace
- その他
  - Wireshark

Conping

- conping
  - 指定したコンテンツがキャッシュされているノードの特定ツール
    - Contrace（次項）と異なりコンテンツまでの経路やキャッシュされているコンテンツの詳細を取得することはできない
  - conping prefix [-r responder] [-w wait_time] [-h hop_limit]

    responder　　応答を要請するノード（IPアドレス）

    wait_time　　最大応答待ち時間（sec）

    hop_limit　　要求メッセージの最大ホップ数
  - 実行結果は以下の形式で出力される

    response from *Responder: Result* time=*RTT* ms

    Responder　　：応答者のIPアドレス

    Result　　：conpingの実行結果

    　cache　　：「prefix」に前方部分一致するコンテンツがResponderにキャッシュされている

    　no cache　　：「prefix」に前方部分一致するコンテンツがResponderにキャッシュされていない

    　no route　　：要求メッセージの転送先がResponder のFIBに登録されていないため転送できなかった

    RTT　　：coping実行から応答メッセージを受信するまでの経過時間

Contrace

- contrace
  - 指定したコンテンツがキャッシュされているノード、そのノードまでの経路、各ホップ間の遅延、キャッシュされているコンテンツの詳細情報を取得するツール
    - 各ホップ間の遅延を正確に計測するためには各装置が時刻同期している必要がある
  - contrace name_prefix [-p] [-n] [-o] [-r hop_count] [-s skip_hop] [-w wait_time]

name_prefix	トレースするコンテンツのプレフィックス
-p	「name_prefix」とコンテンツのName照合方法として前方部分一致を許可
-n	経路とRTTのみを取得する
-o	publisherに対する応答を要求する
hop_count	要求メッセージの最大ホップ数
skip_hop	contraceを実行したノードより「skip_hop」ホップ以下に位置するノードは、「name_prefix」とマッチングするコンテンツをキャッシュしていても応答しない
wait_time	最大応答待ち時間（sec）