Cefpyco：Cefore アプリケーション開発用 Python パッケージ

大岡 睦† 朝枝 仁†

†国立研究開発法人 情報通信研究機構 〒184-8795 東京都小金井市貫井北町 4-2-1

E-mail: †{a-ooka, asaeda}@nict.go.jp

あらまし コンテンツ指向ネットワーク技術（Content-Centric Networking: CCN）を用いた通信を可能とするソフトウェアプラットフォームとして，我々は Cefore を開発し公開している．しかし C 言語で開発されている Cefore を用いたアプリケーション開発が困難であるという課題があった．そこで，CCN アプリケーション開発を容易にするために，CCN 機能を Python から呼び出せるように Python を用いた Cefore のアプリケーション開発用ラッパーコード cefpyco を開発した．cefpyco は Cefore パッケージとして公開し，CCN アプリケーション開発ならびにそれを利用した研究遂行の円滑化を図っている．

キーワード ICN, CCN, Cefore, open source, programming

Cefpyco : Python Compact Package for Developing Cefore Applications

Atsushi OOKA† and Hitoshi ASAEDA†

†National Institute of Information and Communications Technology (NICT)

4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan

E-mail: †{a-ooka, asaeda}@nict.go.jp

Abstract We have developed and released an open software platform enabling Content-Centric Networking (CCN) named Cefore. Cefore provides some tools to test CCN communications; however, it is difficult to enhance the C-language-based Cefore functions or implement user applications. We have therefore newly developed the Cefore Python Compact Package named “cefpyco”, which is a python wrapper for the application library of Cefore. The python package cefpyco is a user-friendly implementation as developers can call CCN functions with a few lines of code. The package is released as well as Cefore in order to promote CCN application development and research activities in the research communities.

Keywords ICN, CCN, Cefore, open source, programming

1. はじめに

現在のネットワークの課題を解決する技術として，情報/コンテンツ指向ネットワーク技術（Information-/Content-Centric Networking; ICN/CCN）が注目されている．本技術の一実装として我々は CCN ソフトウェアプラットフォーム Cefore を開発し，オープンソースとして公開している[1,2]．Cefore は ICN の一実装である CCNx の仕様[3,4]に準拠した通信をサポートするソフトウェアプラットフォームである．軽量性と拡張性の両立を重視しており，最小構成では CCN 通信を実現するための最低限の CCN パケット転送機能のみを持つソフトウェアルータ cefnetd を用いることで Raspberry Pi 等の非力なデバイスでも動作可能でとなっている．一方，高機能・大容量のキャッシュ実装である csmgrd のように，ライブラリや外部ツールとの連携によって，研究者が独自に実装した提案方式の組み込みも可能になる．

Cefore は軽量性と性能を重視した結果，ソースコードやライブラリ群は全て C 言語で開発されている．しかし，C 言語を用いた開発では，文字列操作や動的なメモリ確保･解放に関するコード量が多くなりやすい．デフォルトオプション等も指定しづらいため，Cefore とアプリケーション間の通信を初期化する処理も冗長になりがちである．特に CCN アプリケーションでは Name 等の文字列データや可変長データが頻繁に扱われるため，概念実証等の目的で性能よりも迅速な実装・検証実験が求められる場面では，その開発コストが課題となっていた．

そこで，開発コスト削減および CCN 研究開発推進のため，我々は Cefore アプリケーション用ライブラリ群の Python ラッパーを開発し，Python パッケージ cefpyco として公開した[2]．cefpyco では基本的な CCN 通信で用いる Interest および Data パケットの送受信をサポートしており，例えば，単に Interest を 1 つ送出するために，C 言語では 33 行必要だが，Python であれば 4 行で同じ処理を実装できる．また，容易にコンテンツ

表 1　CefpycoHandle クラスのメソッド一覧

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．

表 2　CcnPacketInfo クラスのプロパティ値一覧

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

の受信者（以下，コンシューマ）や供給者（以下，パブリッシャ）用のアプリケーションを実装することができ，特に，Interest パケットを受信して Data パケットを返送するパブリッシャアプリケーションは，Cefore の標準ツールではサポートされていないが，cefpyco を用いることで容易に開発可能である．他にも，Python 独自の略記法等を用いて記述が簡略に済むように工夫を施している．このように，cefpyco は CCN アプリケーション開発を容易化するものであり，研究において独自のアプリケーションを実装して実験・検証が必要な場面に適しており，CCN 研究開発促進に寄与することが期待される．

2. パッケージの設計・構成

cefpyco は Cefore のアプリケーション開発用 C 言語ライブラリの Python ラッパーとして設計・開発されている．実行には Cefore が動作する PC と，Python 2 または Python 3 環境を必要とし，Python 用ライブラリと

ソースコード 1　コンシューマアプリケーション

```
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_succeeded and (info.name == "ccnx:/test") and (info.chunk_num == 0):
         print("Success")
         print(info)
            break
 sleep(1)
```

ソースコード 2　パブリッシャアプリケーション

```
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, cache_time=7200000)
```

してインストール可能なように構成されている．つまり，インストール後は“import cefpyco”を python のソースコードに記述するだけで，PC 内のどこからでも cefpyco の機能群を呼び出すことが可能である．Python による実装を選択した理由は，現在主流となりつつあるプログラミング言語の中で多くのプラットフォームにおいて動作し，かつ動的型付けのインタプリタ型オブジェクト指向言語であることから，所望の CCN 機能を手軽に実装して研究開発を効率化する効果が期待できるからである．また，インデントを使ってブロックを定義することから，同種の Ruby 等よりコードの可読性を確保しやすく，今後の CCN 関連研究の進展に伴ってコードの共有が行われるようになった場合にも適当であると考えられる．

cefpyco は主に CefpycoHandle と CcnPacketInfo の 2 つのクラスから構成される．CefpycoHandle は Cefore との通信を仲介する役割を果たし，Interest・Data パケットの送受信等の基本的な CCN 通信を実現する．CcnPacketInfo は受信したパケットの情報を格納するためのクラスであり，CefpycoHandle.receive()メソッドの返り値として，Cefore から受信したパケットの情報を Python コード内で確認することができる．

アプリケーション開発者は主に CefpycoHandle を用いて所望の CCN 通信を実装することとなる．CefpycoHandle がサポートするメソッドは表 1 の通りである．この中で，begin と end メソッドは with 構文で自動的に呼び出されるため，NICT 独自機能の Symbolic Interest を送出するメソッドを除けば，基本的なメソッドは 5 つであると考えてよい．C 言語の場合，少なくとも 11 個の関数を把握しておかねばならず，それぞれ専用の構造体を引数に取るため習得には時間を要するが，本クラスのメソッドは receive 関数の返り値を除きプリミティブ型のみをやり取りするため，Python 経験者であれば容易に利用することが可能である．各メソッドの用法については，実装例と共に 3 章で説明する．

アプリケーションでパケットを受信する必要がある場合には，CefpycoHandle.receive()メソッドを呼び出して CcnPacketInfo クラスのオブジェクトを受け取ることになる．このクラスのプロパティ値を表 2 に示す．CCN では Interest・Data パケットという 2 種類のパケットに，URL のような Name と呼ばれるアドレスを載せてコンテンツの送受信を行うが，そのパケット種別や Name アドレス，および Data パケットのペイロードの値等を確認することができるようになっている．例えばコンテンツ供給者の役割を持つアプリケーションを実装する場合は，CcnPacketInfo.name プロパティを確認して自身が提供するどのコンテンツが要求されているかを判断し，適切に Data パケットを返送するような処理を実装することとなるだろう．C 言語で同様の処理を実装する場合，先述の関数群に加えて，それぞれが 10 以上の変数を持つ構造体を少なくとも 4 種類理解しておく必要があり，それと比較すると理解するために必要な情報量は少なくて済む．

3. 実装例

ソースコード 1，2 に，cefpyco を用いた簡易コンシューマ，パブリッシャアプリケーションの実装例を示

す．これは，基本的な CCN 通信である Interest と Data パケットの交換を行うアプリケーションである．具体的には，先にパブリッシャアプリケーションが起動して Interest パケットを待機しているところに，コンシューマアプリケーションを起動して１つの Data パケットとして提供されている ccnx:/test/Chunk=0 という名前のコンテンツを要求し，受信するというシナリオを実現するためのアプリケーションである．両アプリケーションが cefnetd で疎通しているようなネットワーク環境であれば，CCN パケットの転送処理等は Cefore 側で行われる．

両アプリケーションに共通の処理として，どちらも最初に cefpyco.create_handle()メソッドを呼び出しているが，これは Python の with 構文を使用して簡単に CefpycoHandle オブジェクトを扱う際に用いるためのユーティリティである．この呼び出しだけで，実装する Python アプリケーションの cefnetd との接続の初期化・終了処理を記述できる．

コンシューマアプリケーションは，ccnx:/test/Chunk=0 という Name アドレスを持つコンテンツを要求するためには Interest パケットを送出する必要がある．そのために先ずは CefpycoHandle.send_interest()メソッドを呼び出し，次に要求したコンテンツの入った Data パケットがパブリッシャアプリケーションから返送されてくるまで CefpycoHandle.receive()で待機する．受信した場合は，それが所望のパケットであることを確認した後，そのパケットに含まれる情報を表示して終了する．

一方，パブリッシャアプリケーションは，ccnx:/test という名前のコンテンツを提供するために，CefpycoHandle.register()関数を用いて cefnetd に対して予め自身の提供するコンテンツのプレフィックス名を通知しておく．こうすることで cefnetd にパブリッシャアプリケーション宛の FIB が構築され，パブリッシャアプリケーションは cefnetd から Interest パケットを受信することができるようになる．コンシューマアプリケーションに先だって起動されたパブリッシャアプリケーションは，コンシューマアプリケーションからの Interest パケットを受信し次第，CefpycoHandle.send_data()メソッドを呼び出して Data パケットを返送する．

以上の実装によって，CCN で疎通している２ノード間でのコンテンツの交換を行うアプリケーションを実現できる．ただし，この例は極めて単純な実装例に過ぎない．この例では Data パケットを１つだけ返送しているが，一般にはコンテンツが１パケット（約 1500 バイト）よりも大きいサイズとなる．コンテンツ全体を送るためには，複数回の Interest と Data パケットの交換が必要となるだろう．また，輻輳制御処理は現在の Cefore 本体には組み込まれていないため，独自に実装する必要がある．簡単なパイプライン要求を行うアプリケーションの実装は cefpyco のソースコードと共に提供されている CefApp アプリケーションで行っており，その実装を参照されたい．

4. 結論

本稿では，CCN アプリケーション研究開発推進のために我々が開発した，Cefore アプリケーション開発用ライブラリの Python ラッパーパッケージ cefpyco の構成について，また，cefpyco を用いた簡単なアプリケーションの実装例について解説した．cefpyco を利用すれば C 言語よりも手軽に CCN アプリケーション開発が可能であり，極端な性能が求められない概念実証目的の利用であれば十分に役割を果たすことが期待される．

将来的にはパッケージの充実を図りたい．例えば，Cefore でサポートされているツールの実装例や，輻輳制御等についても簡単なアルゴリズムについてはライブラリでの提供が可能であることが望ましい．また，アプリケーション開発のみならず，cefnetd との連携機能として，cefnetd が有する CCN ルータ専用テーブルから情報を取得したり制御したりする機能の実装も望まれる．そして，実装した機能を開発者同士で共有する git のような仕組みを作り，更なる CCN 研究開発活動の促進を図りたい．

文　献

[1] H. Asaeda, A. Ooka, K. Matsuzono, R. Li, “Cefore: Software Platform Enabling Content-Centric Networking and Beyond,” IEICE Transactions on Communications, E102.B. no.9, pp. 1792-1803, Sept. 2019.

[2] “Cefore,” https://cefore.net (Accessed Sept. 2019)

[3] M. Mosko, I. Solis, and C. Wood, “Content-Centric Networking (CCNx) Semantics,” IETF, RFC8569, https://tools.ietf.org/html/rfc8569, July 2019.

[4] M. Mosko, I. Solis, and C. Wood, “Content-Centric Networking (CCNx) Message in TLV Format,” IETF, RFC8609, https://tools.ietf.org/html/rfc8609, July 2019.