社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

証明支援系 Coq を使った CCN のモデル化と検証について

森嶋　崇†　後藤　瑞貴†　高橋　和子†

† 関西学院大学理工学部情報科学科
〒 669–1337 兵庫県三田市学園 2–1

あらまし　Content-Centric Networking(CCN) は 2009 年に提案された通信方式であり，アドレスではなくコンテンツ名に注目して通信をおこなうものである．現在はシミュレーションをベースとして動作や性能のチェックがおこなわれているが，確立した技術ではないため，実用化にむけて，動作の正当性の検証が望まれる．本研究では，証明支援系 Coq を用いて CCN のプロトコルを帰納的にモデル化し，各ノードでおこなわれているマッチング処理を実装した．そのモデル上で，あるコンテンツがネットワーク上に存在するとき，ユーザがそれを要求すれば，必ず正しいものを受信できることを検証した．

キーワード　Coq, CCN, 正当性検証, ネットワークプロトコル, 定理証明, 証明支援系

Modeling and verification for CCN using a proof assistant Coq

Takashi MORISHIMA†, Mizuki GOTO†, and Kazuko TAKAHASHI†

† School of Science and Technology, Kwansei Gakuin University
2-1,Gakuen, Sanda, 669–1337, Japan

Abstract　Content-Centric Networking (CCN) is a communication architecture which was developed on 2009. Communication on CCN is based on the names of objects, rather than on addresses. Although its behavior and performance are explored mainly by simulation, they have not been well-established, and behavioral correctness is required to be verified for its practicel use. In this paper, we create an inductive model for a CCN protocol and implement the maching process undertaken in each node. Then we prove that a node can retrieve the content if and only if the user sends a request when it exists in the network, as one of behavioral correctness.

Key words　Coq, CCN, behavioral correctness, network protocol, theorem prover, proof assistant

1. はじめに

インターネットでパケット通信の構造ができて以来，約 50 年で既に 500 エクサバイトものコンテンツがネットワーク上に存在している．しかし，ユーザはコンテンツがどこから得られるかということには関心がなく，コンテンツを得ることができれば十分である．これは現在の，“どこ”という“アドレス”を利用したインターネットの構造と矛盾していることになる．本研究の対象となる Content Centric Networking(以下 CCN) [1] は“なに”という“コンテンツ名”に注目した新しい通信方式である．従来の通信では，アドレスを利用して通信をしていたが，CCN はオブジェクトの名前で通信をおこなう．CCN の利点として，コンテンツを任意の中継ノードでキャッシュできるので，ネットワークの利用効率の向上や，応答時間の短縮が挙げられる．ただし，CCN は現在も性能評価がおこなわれており，確立された技術ではなく，動作の正当性の研究はされていない．また，ネットワークプロトコルの正当性を検証した事例 [6] [7] はあるが，CCN を対象とした研究はされていない．本研究では，CCN のプロトコルを証明支援系 Coq で帰納的にモデル化し，動作の正当性の検証をした．具体的には，正当性の 1 つとして，ユーザがあるコンテンツを要求すれば，必ず正しいものを受信できることを検証した．

本論文の構成は，以下の通りである．第 2 章では証明支援系 Coq の概要を述べる．第 3 章では CCN の概要を述べる．第 4 章では CCN のモデル化と検証について述べる．第 5 章ではまとめと今後の課題について述べる．

2. 定理証明とモデル検査

一般に対象をプログラムで扱えるようにモデル化し，要求される仕様を満たしているか自動的に計算し検証する方法を形式的手法といい，形式的に問題を捉えることでさらに厳密な安全性の検証や複雑な環境での検証が可能である．ただし，どのように形式化するか，対象をどこまで深く検証するか，といった点でアプローチの手法が変わってくる．代表的な手法としてモデ

ル検査と定理証明の 2 つがある．モデル検査は起こりうる全ての状態変化を網羅的に検証する手法で，扱うことのできるモデルは有限である [3]．一方，定理証明は，ある公理系とその推論規則を使用し，検証する性質を満たしているかどうか検証する方法である．定理証明の代表的なツールに，Isabelle/HOL [2], Coq [4], PVS [8], ACL2 [9], Agda [10] などがある．これらは証明支援系とも呼ばれる．定理証明は無限状態を扱えるため，モデル検査よりも検証範囲が広い．また，検証するために実行にかかる一動作ずつの処理も高速である．定理証明に比べモデル検査では検証できる性質が限定される点では劣っているが，検査対象が正しく動くかどうかというバグの早期発見を目的とするならば，反例が生まれるかどうかのみを判定するモデル検査の方が優れている．また，定理証明では証明が解けない場合の解析は非常に困難である．本研究では，通信方式の検証をおこなう．コンテンツ名は無限であるので，無限個の値を扱える定理証明の方が向いている．また，CCN のソースコードが Java で書かれているため，後の応用を考え，他言語との互換性の多い Coq を使うことにした．

Coq は，INRIA(フランス国立情報学自動制御研究所) で開発されている証明支援系である．Coq は型理論に基づいており，一般的なプログラミング言語である C 言語等の手続き型言語とは違い，関数型言語 OCaml で書かれている．Prop(命題), Set(項)，Type(2 つともを含む) の 3 つの型を中心に，bool 型や数値型が表現されており，それにより型チェックによる制約が強い．また，Coq は対話型言語の 1 つでもある．命令を 1 ステップずつ読み込み処理するといった流れで動き，もし誤りがあった場合はそこで処理が停止するので，エラーの早期発見・修正がしやすい．

3. Content Centric Networking

CCN [1] は Van Jacobsen が提案した，コンテンツの名前に基づいて通信をおこなうネットワークアーキテクチャである．物理的なネットワークの構造に変更を加えることなく，ソフトウェアとしての機能を書き換えるだけで，既存のインフラストラクチャ上に構築することができる．ノードとはユーザ，ルータ，サーバの 3 種類のことである．従来の通信では，パケットは 2 つのアドレスを利用し，2 つのノード間で対話が成り立っていた．CCN では中継ノードでコンテンツをキャッシュすることができ，効率的にコンテンツを取得できる．TCP/IP と CCN の構造の違いは以下の 3 点である．まず，通信で使うパケットの種類は，TCP/IP では単独なのに対して，CCN では要求と応答の 2 種類に分けている．次に，TCP/IP ではパケットに行き先のアドレスを持たせていたのに対して，CCN ではコンテンツ名を持たせている．また，TCP/IP ではノードはバッファの機能しか持たないが，CCN では新たに 3 種類の構成要素を持つ．

3.1 CCN のパケット

CCN は従来のようにパケットを利用した通信ではあるが，Interest パケットと Data パケットの 2 種類で構成される．Interest パケットは，Content Name と Selector，Nonce を持つ

図 1　各パケットの持つ情報

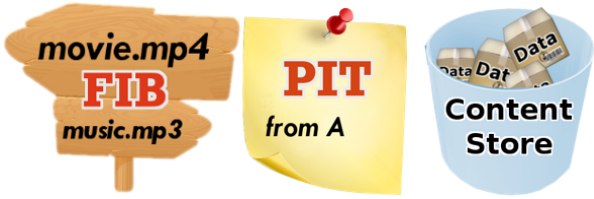

図 2　ノードの持つ 3 つの構成要素

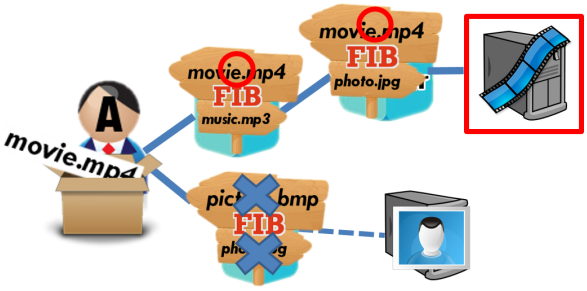

図 3　Forwarding Information Base

が，今回は Content Name のみに着目し，コンテンツ名のみを持つとした．このパケットは，ユーザがサーバに向けてコンテンツを要求する際に用いる．Data パケットは，Content Name, Signature，Signed Info，Data の 4 つを持つが，Interest パケットと同様に，今回は Content Name と Data の 2 つに着目し，コンテンツ名とそのコンテンツデータを持つとした．このパケットは，サーバがデータを返送する際に用いる (図 1)．

3.2 CCN の各ノードが持つ構成要素

各ノードの持つ構成要素は，従来のようにパケットをキャッシュする機能だけではなく，Forwarding Information Base(以下 FIB)，Pending Interest Table(以下 PIT)，Content Store(以下 CS) の 3 種類である (図 2)．

FIB は，Interest パケットが目的のコンテンツを持つサーバにたどり着くように転送するのに用いられる (図 3)．FIB にもコンテンツ名が含まれ，Interest パケットのコンテンツ名と比較し，一致すれば隣接するノードに送信することができる．

PIT は，ユーザからサーバに向けて Interest パケットが通過した軌跡を記録するリストで，各ノードがリストを持っている．FIB と同様に PIT にもコンテンツ名が含まれ，コンテンツ名ごとに違うリストで管理している．これは，サーバからユーザに Data パケットを送信する際に，転送先を決定するのに用いられる (図 4)．

CS はバッファで，これは従来の機能と似ているが，パケット送信後もすぐに破棄せずにできる限り Data パケットを保持し，保持しきれなくなると，利用されずに古くなったものから

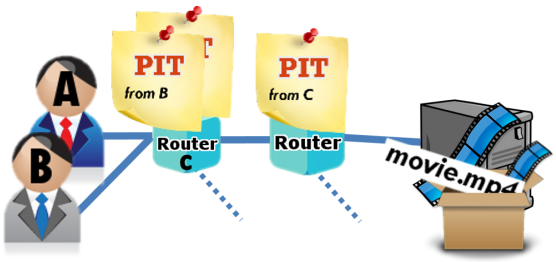

図 4　Pending Interest Table

破棄する．この CS のおかげで，同一のコンテンツ名の要求に対して，再利用することができる．

3.3　CCN の動作

CCN では，パケットが持つコンテンツ名と構成要素が持つコンテンツ名でマッチングをおこない，どの構成要素でマッチングしたか次第でパケットの送信先が異なる．マッチングでは同時にそのノードの構成要素の内容を更新する．

3.3.1　Interest パケットに対する動作

ユーザがコンテンツ名を含む Interest パケットを，隣接する全てのノードに送信する．各ノードは Interest パケットが到着すると CS，PIT，FIB の優先度順で，パケット内のコンテンツ名と各ノード内のコンテンツ名でマッチングをおこなう (図 5)．

- CS でマッチングする場合

到着したノードに目的の Data パケットがあるということなので，Data パケットを要求のあった方に返送し，Interest パケットは破棄する．

- PIT でマッチングする場合

同じコンテンツ名を含んだ Interest パケットが既にこのノードを通ったことがあるということなので，どこから来たかという情報 (PIT) をこのノードのリストに追加し，Interest パケットは破棄する．

- FIB でマッチングする場合

コンテンツ名が一致する Interest パケットが，初めてこのノードに到着したということなので，どこから来たかという情報 (PIT) をこのノードのリストに追加し，目的のコンテンツを保持するサーバに近い側のノードに Interest パケットを送信する．

- どれもマッチングしない場合

Interest パケットを破棄する．

3.3.2　Data パケットに対する動作

Interest パケットの場合と同様に，各ノードは Data パケットが到着すると CS，PIT，FIB の優先度順で，パケット内のコンテンツ名と各ノード内のコンテンツ名でマッチングをおこなう．Data パケットは，PIT とマッチングした場合以外はパケットが破棄されるので，PIT のマッチング以外は例外処理といえる (図 6)．

- CS でマッチングする場合

既に同じコンテンツを持つ Data パケットが到着したノードに存在しているので，Data パケットを破棄する．

- PIT でマッチングする場合

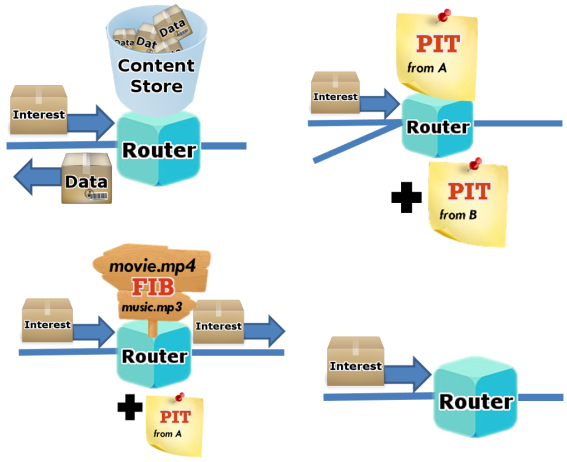

図 5　Interest パケットのマッチング

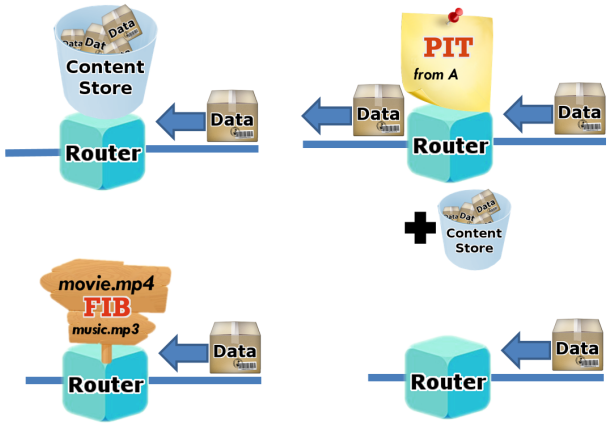

図 6　Data パケットのマッチング

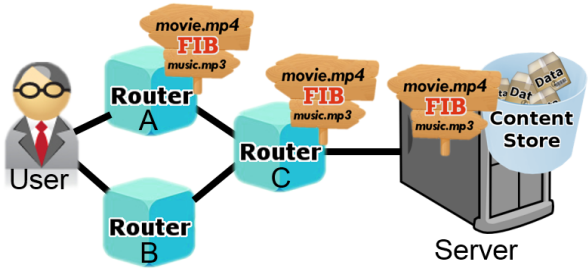

図 7　初 期 状 態

到着したノードには，どこから来たかという情報がリストで残っているので，データをキャッシュしつつ (CS)，リストの情報をもとに，Data パケットを送信する．

- FIB でマッチングする場合

このノードを通った Interest パケットがなかったということになるので，Data パケットを破棄する．

- どれもマッチングしない場合

Data パケットを破棄する．

動作例として図 7 のようなモデルを考える．

まず Interest パケットの流れを説明する．ユーザは，隣接するルータ A とルータ B に Interest パケットを送信する．ルータ A が Interest パケットを受信すると，ルータ A には FIB が存

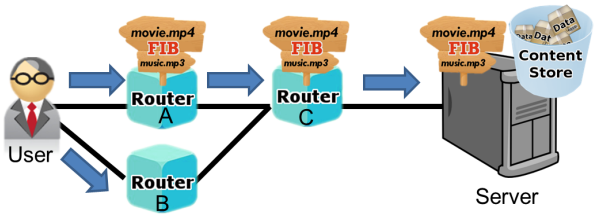

図 8　例：Interest パケットの流れ

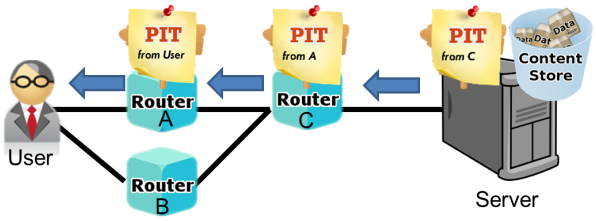

図 9　例：Data パケットの流れ

在するため，FIB でマッチングをおこなう．この場合，パケットがユーザから送信されたことを記録するためにルータ A の持つ PIT にこの情報が追加され，隣接するルータ C に Interest パケットが送信される．ルータ B が Interest パケットを受信すると，ルータ B には 1 つも構成要素が存在しないため，Interest パケットは破棄される．ルータ C がルータ A から Interest パケットを受信すると，ルータ C には FIB が存在するため，FIB でマッチングをおこなう．この場合，パケットがルータ A から送信されたことを記録するためにルータ C の持つ PIT にこの情報が追加され，隣接するサーバに Interest パケットが送信される．サーバがルータ C から Interest パケットを受信すると，サーバには CS と FIB が存在し，CS とのマッチングが優先されるので，サーバからルータ C に Data パケットが送信され，Interest パケットは破棄される (図 8)．

次に Data パケットの流れを説明する．ルータ C がサーバから Data パケットを受信すると，ルータ C には PIT と FIB が存在するので，PIT で優先的にマッチングをおこなう．ルータ C からユーザ側に向かってはルータ A とルータ B の 2 つの隣接ノードがあるが，PIT には Interest パケットを送信したのはルータ A のみであるという記録があるため，ルータ A のみに Data パケットが送信される．ルータ A がルータ C から Data パケットを受信すると，ルータ A には PIT と FIB が存在するので，PIT で優先的にマッチングをおこなう．同様に PIT の情報を利用してルータ A からユーザに Data パケットが送信される．この結果，Data パケットがユーザに到達し，ユーザのデータ取得が成功したことになる (図 9)．

4.　CCN のモデル化と検証

4.1　CCN のモデル化

CCN のモデル化を，帰納法に基づいておこなう．帰納法を用いる利点は，無限状態や変数のとる値が無限の場合が扱えることである．現在は各ノードの接続，ネットワークの構造は固定し，FIB を持つノードを初期状態で与えたモデルを対象にしている．また，オリジナルのコンテンツを，サーバに CS として持たせることで実装している．今回は図 7 に示す単純な事例を扱う．

Agent とはノードのことであり，user，routerA，routerB，routerC，server の 5 つで帰納的に定義する．この定義により，Agent 型はこの 5 つのみになり，これら全てについて，ある命題 P が成り立つならば，任意の Agent について P が成り立つといった定理が自動的に加わる．

Struct とは構成要素のことであり，FIB，PIT，CS の 3 つをコンストラクタとして帰納的に定義する．

Packet とはパケットのことであり，Interest と Data の 2 つをコンストラクタとして帰納的に定義する．

Cname とはコンテンツ名のことであり，無限を扱えるように自然数として定義する．

CCN は，マッチングが起こると各ノードが持つ構成要素のうち，PIT と CS が変化する場合がある．そこで，あるノードからあるノードへ送信するというイベント (Say)，Interest パケットの PIT，FIB マッチングの際に起こる，PIT を追加するというイベント (AddPIT)，Data パケットの PIT マッチングの際に起こる，CS を追加するというイベント (AddCS) を Event として以下のようにそれぞれ定義する．

```
Inductive Event : Type :=
 |Say : Agent → list Agent → Packet → Event
 |AddPIT : Struct → Agent → Event
 |AddCS : Struct → Agent → Event.
```

これらのイベントを時系列リストで管理し，その時々ごとのノードが持つ構成要素は何があるか，このリストを全て確認することで対応した．また Coq では，一般的な else if 文のような，条件分岐するパターンマッチングの機能はあるが，プログラムの上から順に優先順位が付く訳ではないため，マッチングが成功するか否かで真偽値を返す関数を追加して対応した．

以下のプログラムは，Interest パケットがマッチングするときの定義である．

```
Definition Mat (A1 A2 :Agent)(C1 : Cname)
(LE1 : list Event) : list Event :=
 match (CSmat (Entry A2 LE1 C1)),
(PITmat (Entry A2 LE1 C1)),
(FIBmat (Entry A2 LE1 C1)) with
 |true, _, _ ⇒
(Say A2 (A1::nil) (Data C1))
::(AddPIT(PIT C1 A1) A2) ::LE1
 |false, true, _ ⇒
(AddPIT(PIT C1 A1) A2)::LE1
 |false, false, true ⇒
 (Say A2 (connect A2) (Interest C1))
::(AddPIT (PIT C1 A1) A2)::LE1
 |false, false, false ⇒ LE1
 end.
```

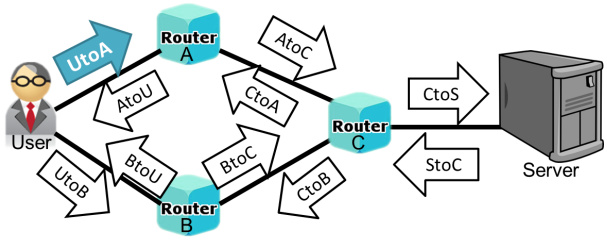

図 10　例の Send のコンストラクタの数

“::”はリスト構成子を示す．A::B は A がリストの先頭要素，B が残りの要素のリストである．Mat は，2 つの Agent 型，Cname 型，Event 型リストから Event 型リストへの関数である．この関数の本体では，真偽値を返す CSmat，PITmat，FIBmat においてパターンマッチングさせることで場合分けしている．例えば，Csmat が false，PITmat が false，FIBmat が true であった場合，これは FIB でマッチングしたということを表す．この場合，それまでのイベントの時系列リストの先頭に，Say という Interest パケットの送信イベントと，AddPIT という PIT を加えるイベントを追加する．

Send は，CCN のプロトコルであり，リストの Event 型を取り，命題の型を返す．リストイベント LE1 に対して，”Send LE1” は LE1 が CCN のプロトコルにしたがって動作したものであることを表す．全てのイベントはどの時刻でも起こりうるものであり，イベントによってノードの構成要素が変化する．マッチングは構成要素に依存して結果が変わるため，イベントの起こる順序によってパケットに対する動作が変わる．したがって，イベントリストは複数存在する．Send は各ノード間の接続の数だけコンストラクタをもつ (図 10)．その 1 つ，Sa について説明する．

```
Sa : forall(C1 : Cname)(LE1 : list Event),
Send LE1
→ In (Say user (connect user) (Interest C1)) LE1
→ Send (( Mat user routerA C1 LE1 ))
```

リストイベント LE1 はこのプロトコルにしたがって動作したものであり，ユーザから，隣接するノードへの送信イベントが LE1 の要素に含まれているならば，ルータ A でマッチングをおこなう．これは，上流に向かって 1 つ前のノードからパケットが到着した場合に限り，マッチングをおこなうということである．マッチングの結果となるイベントを LE1 の先頭に追加する．その結果得られるリストイベントはまた CCN のプロトコルにしたがって動作したものである．

Send では，例えばユーザからルータ A に Interest パケットが送信されたイベントの直後にすぐにルータ A からルータ C に Interest パケットが送信される場合もあればユーザからルータ A に再度 Interest パケットが送信される場合もある．ただし，ルータ C からサーバに送信が起こることはない．

4.2　Coq による検証

通信方式の安全性，正当性は，様々な保証を定式化したものの複合体として示される．この保証の選び方には決まりが無く，検証する目的に応じて選択する必要がある．本研究では，定式化した保証の命題がプロトコルの性質を常に満たすかを証明することになる．今回の事例では，ユーザに隣接するノードに必ず 1 つは FIB が存在するので，マッチングが一切起こらず通信が途絶えることはないという前提がある．

forward として (i) リストイベントが CCN のプロトコルにしたがって動作しており，ユーザに隣接する全ルータに Interest パケットの送信をするならば，ユーザは将来必ず Data パケットを受信する．という検証をおこなった．なお，今回の事例ではユーザはルータ A 経由でしか Data パケットを受信しないような初期状態に設定している．また，その逆の backward として，(ii) リストイベントが CCN のプロトコルにしたがって動作しており，ユーザが Data パケットを受信するならば，以前にユーザが Interest パケットの送信をしていたはずである．という検証をおこなった．これらを実際のプログラムでは以下のように表現した．

```
Lemma FORWARD :
forall(LE1 : list Event)(C1 : Cname),
 Send LE1
→ In (Say user (connect user) (Interest C1)) LE1
→ exists eLE : list Event,
Send (Say routerA (user :: nil) (Data C1)
 :: eLE ++ LE1).
```

```
Lemma BACKWARD :
forall(LE1 : list Event)(C1 : Cname),
 Send LE1
→ In (Say routerA (user :: nil) (Data C1)) LE1
→ In (Say user (connect user) (Interest C1)) LE1.
```

“++”はリスト連接子を示す．”A++B” は，リスト A とリスト B をつなげた 1 つのリストを表す．“In”は要素がリストに含まれることを示す関数で，“In A B”は，リスト B は A を要素に含むことを表す．

(i) の場合，先に述べたように現在のイベント列である LE1 はその後起こるイベントによって複数の列に分かれる．したがって，ユーザが隣接するルータへ Interest パケットを送信してから 0 個以上のイベントをはさんでルータ A がユーザにその Data パケットを送信するというイベントが起こるという記述をしている．

(ii) の場合，LE1 が CCN のプロトコルにしたがって動作したものであり，ルータ A からユーザに Data パケットを送信するイベントが LE1 に含まれていれば，ユーザからルータ A に Interest パケットを送信するイベントが LE1 に含まれているという記述をしている．先に述べたように，Interest パケットでは上流に向けて，Data パケットでは下流に向けてそれぞれ 1 つ前のノードからパケットが到着した場合に限り，次の送信イベントが起こり，リストイベント LE1 の先頭にそのイベントが追加される．したがって，ルータ A がユーザに Data パケットを送信するためにはルータ A にどこかからパケットが

到着していなければならない．このようにさかのぼって考えると，”Send LE1” を満たす LE1 においてはルータ A がユーザに Data パケットを送信するイベントよりもユーザがルータ A に Interest パケットを送信するイベントの方が早く起こっているはずである．したがって，backward の性質を証明するには上記のような補題を証明すれば十分である．

同じノードで Interest パケットの PIT マッチングが複数回起こったときに，同じ所から来たという情報を何度も追加する場合がある．つまり，ユーザが何度も同じ要求をする DOS 攻撃に近い形である．このとき，PIT マッチングで同一情報を重複して追加していくと，この情報をみて Data パケットの返送先を決定する際に無限の分岐が生じる可能性がある．そのためイベントの型が決まらない．また同一情報の数によって場合分けして証明する必要があるため証明が有限で終わらない．これを回避するために，本研究では，PIT マッチングについては，1 度だけ AddPIT というイベントを認め，2 度目以降はイベントとして認めないような規則にすることで対応した．なお，この DOS 攻撃を低減させる別の手法も提案されている [5].

(i) の証明は 7 個の補題を使って， (ii) の証明は 5 個の補題を使ってそれぞれ証明することができた．証明は，全て合わせて約 1600 行ほどであるが，簡約化すればもっと短くできると思われる．証明は主に帰納法と場合分けで構成される．

5. まとめと今後の課題

本報告では，ネットワークアーキテクチャである CCN のプロトコルを，証明支援系 Coq を用いて帰納的にモデル化し，各ノードでおこなわれているマッチング処理を実装した．そのモデル上で，動作の正当性の 1 つとして，ユーザがあるコンテンツを要求すれば，必ず正しいものを受信できることを検証した．

今回対象とした事例は非常に単純であるが，1 対複数の送信を考慮し，Interest パケットの 3 種のマッチング，Data パケットの 3 種のマッチングが全て含まれている．したがって，ノードの数，ノード同士の接続を任意にして応用が可能であると考える．また，コンテンツ名は無限に扱うことができる．

証明は，各送信ごとに起こりうる全ての送信イベントについて場合分けしている．モデルを大きくすると，送信を 1 回おこなうだけで，ノード間の接続の数だけの場合分けが増え，その送信に対しマッチングが 3 パターンある．したがって，応用はできても実際に証明しようとすると計算量が膨大になりすぎる問題がある．この問題は，現在のモデルが特定のネットワーク構造をそのまま記述していることに起因する．今後はモデルを一般化して記述することで解決できると考える．また，実際の CCN の仕様では PIT を複数回認めるという DOS 攻撃のような要求を許可している可能性があるので，この点についても考慮したい．

文　献

[1] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. and Braynard, R. L., “Networking Named Content ”, ACM CoNEXT, 2009.

[2] Nipkow, T., Paulson, L. C., and Wenzel, M., “Isabelle/HOL a proof assistant for higher-order logic ”, Springer Verlag, 2002.

[3] Clarke, E. M., Grumberg, O. and Peled, D. A., “ Model checking ”, MIT Press, 1999.

[4] Bertot, Y. and Castŕan, P., “ Interactive theorem proving and program development - Coq’Art: The calculus of inductive constructions ”, Springer Verlag, 1998.

[5] Dai, H., Wang, Y., Fan, J., Liu, B., “Mitigate DDoS Attacks in NDN by Interest Traceback ”, NOMEN 2013.

[6] Bharadwaj, R., Felty, A., Stomp, F., “ Formalizing Inductive Proofs of Network Algorithms ”, Proceedings of the 1995 Asian Computing Science Conference.

[7] Stewart, G., “ Computational Verification of Network Programs in Coq ”, CPP 2013.

[8] Owre, S., Rushby, J. M. and Shankar, N., “ PVS: A prototype verification system ”, CADE92, pp.748-752, 1992.

[9] Kaufmann, M., Monolios, P. and Moore, J. S., “ Computer-aided reasoning an approach”, Kluwer Academic Publishers, 2000.

[10] Norell, U., “ Dependently typed programming in Agda ”, Advanced Functional Programming 2008, pp.230-266.