Vol.2016-ITS-64 No.2
2016/3/7

LT ネットワーク符号化通信の情報指向ネットワークへの適用

大塚祐輔[†1]　北村優汰[†1]　磯部光平[†1]　毛利公美[†2]　白石善明[†3]

概要：情報指向ネットワーク（Information-Centric Networking，ICN）は，従来の IP アドレスではなく情報名を基に情報を要求する，そして，ネットワーク内の中間ノードはキャッシュ機能を持つインネットワークキャッシュという特徴がある．車両アドホックネットワーク（Vehicular Ad-hoc Network，VANET）のための ICN が考えられている．また，キャッシュをより高度に利用するために，ランダム線形ネットワーク符号化（Random Linear Network Coding，RLNC）を適用した ICN が提案されている．しかし，RLNC を適用すると復号の計算量が大きい．本稿では，ICN のアーキテクチャで復号の計算量が少ないネットワーク符号化である LT Network Codes（LTNC）が動作する Information-Centric Networking with Built-in LT Network Codes（ICN-LTNC）を提案する．トラフィックシミュレータと連携したネットワークシミュレータを用いたシミュレーション実験により ICN-LTNC が VANET で動作可能であることを確認した．ICN-LTNC と ICN-NC の配信性能，復号に要する計算回数について評価を行い，ICN-LTNC は ICN-NC より復号に要する計算回数が少ないことを確認した．

キーワード：情報指向ネットワーク，データ配信，ネットワークシミュレーション

LT Network Codes Enabled Information-Centric Networking

YUSUKE OTSUKA[†1]　YUTA KITAMURA[†1]　KOHEI ISOBE[†1]
MASAMI MOHRI[†2]　YOSHIAKI SHIRAISHI[†3]

Abstract: Information-Centric Networking (ICN) has following characteristics: A receiver requests information with the name of information instead of IP address and an intermediate node has cache function called in-network cache. Vehicle Ad-hoc Network (VANET) for ICN is considered. To use cache effectively, Information-Centric Networking that Random Linear Network Coding (RLNC) is applied has been proposed. However, RLNC is high decoding complexity. This paper proposes Information-Centric Networking with Built-in LT Network Codes (ICL-LTNC). LT Network Codes is a low decode complexity network codes. Simulation experiments with a network simulator associated with a traffic simulator show that ICN-LTNC is applicable to VANET. Evaluation results in terms of the distribution performance and the number of calculation to decode show that the number of calculation to decode of ICN-LTNC is less than that of ICN-NC.

Keywords: Information-Centric Networking, Data Distribution, Network Simulation

1. はじめに

インターネットはホスト間でのデータ送受信に焦点が当てられていたが，近年では動画や音楽など情報の配信に焦点が当てられている[1]．そのため，従来のホスト中心の TCP/IP アーキテクチャに代わる次世代ネットワークアーキテクチャとして，情報を中心とした情報指向ネットワーク（Information-Centric Networking，ICN）が注目されている[2]. また, 車両アドホックネットワーク（Vehicular Ad-hoc Network，VANET）のための ICN が考えられている[3]. ICN は, 従来の IP アドレスではなく情報名を基に情報を要求する，ネットワーク内の中間ノードはキャッシュ機能を持つインネットワークキャッシュという特徴がある．インネットワークキャッシュを導入することで情報の配信を効率化することができる[4].

キャッシュをより高度に利用するために ICN のアーキテクチャにランダム線形ネットワーク符号化（Random Linear Network Coding，RLNC）[5]を適用した Information-Centric Networking with Built-in Network Coding（ICN-NC）[6]が提案されている．ICN に送信ノード，中間ノードで代数的に符号化し受信ノードの復号性能を向上させる RLNC を適用することで，キャッシュをより高度に利用し，配信性能が向上することが示されている．

RLNC の復号は連立方程式を解く処理となり計算量が大きい. VANET におけるノードは移動あるいは停止している車両であり，車両に搭載されるネットワーク機器はリソース制約が強い[7]．VANET 上で動作可能かつ計算量の少ない符号化方式が動作する ICN のアーキテクチャが求められる．

本稿では，ICN のアーキテクチャで復号の計算量が少ないネットワーク符号化である LT Network Codes（LTNC）[8]が動作する Information-Centric Networking with Built-in LT Network Codes（ICN-LTNC）を提案する．LTNC は符号化，復号を排他的論理和のみで行う LT 符号に基づくネットワーク符号化であり，計算量が少なく，ICN 上でのネットワーク符号化通信の計算量低減が期待できる．

†1 名古屋工業大学
Nagoya Institute of Technology
†2 岐阜大学
Gifu University
†3 神戸大学
Kobe University

トラフィックシミュレータと連携したネットワークシミュレータを用いたシミュレーション実験により ICN-LTNC が VANET で動作可能であることを確認する．ICN-LTNC と ICN-NC の配信性能，復号に要する計算回数について評価を行い，ICN-LTNC は ICN-NC より復号に要する計算回数が少ないことを確認する．

2. 既存方式：Information-Centric Networking with Built-in Network Coding（ICN-NC）

ICN は，パケットロスが発生するとキャッシュを高度に利用できないことがある．そこで，パケットロス耐性，伝達速度の向上が期待できる RLNC を ICN に適用することが考えられている．ICN-NC は，ICN で RLNC が動作するアーキテクチャである．文献[6]では，ICN-NC は ICN よりインネットワークキャッシュを高度に利用できるようになることが示されている．

2.1 パケットフォーマット

ICN-NC は，Interest パケット，Data パケットの 2 種類のパケットがある．Interest パケットはコンテンツの要求に使用する．Data パケットは要求されたコンテンツを返すのに使用する．

Interest パケットのパケットフォーマットを図 1 に示す．受信ノードは所望するコンテンツの名前を Interest パケットの Content Name に設定し，送信することで所望したコンテンツの取得を試みる．

文献[6]では Data パケットのフォーマットを図 2 のように定めている．RLNC では，データサイズが大きい場合，一度に全てのデータを符号化せずに，いくつかの generation にわけて符号化する．Gen_ID は，どの generation であるかを示すための ID であり，長さは 2Byte である．また，ネットワーク符号化では，符号化データからデータを復元するのに符号化ベクトル（Coding_Vector）を用いる．そのため，Coding_Vector を Data パケットに含める．Coding_Vector の長さはデータの分割数 k としたとき kByte である．ICN は，Data パケットの Signature，SignedInfo によって，コンテンツベースのセキュリティを実現する．Signature は Name と Content から作成される．受信ノードは，SigInfo を用いて Signature を得ることで，データの完全性と認証が検証される[9]．ICN では，Content が変わらない前提で Signature を作成するが，ネットワーク符号化では，送信ノード，中間ノードの符号化によって，Content は変わってしまう．そのため，ICN-NC では Signature_ID，Message_ID を利用する．Signature_ID は，Signed されたメッセージの ID であり，1Byte である．Message_ID は，generation 内の何番目のメッセージであるかを示し，1Byte である．Message_ID の値は，Signature_ID の値と同じにする．符号化により値が変わるのは Content と Message_ID だけである．受信ノードの復号後，Message_ID は符号化前の Message_ID と同じになる．

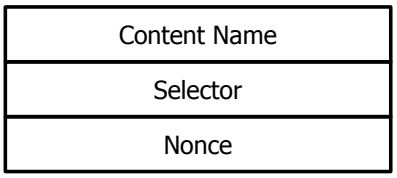

図 1 ICN-NC Interest パケットフォーマット

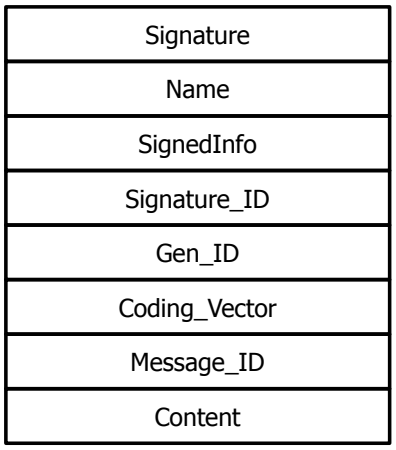

図 2 ICN-NC Data パケットフォーマット

Message_ID と Signature_ID が同じ場合，Content が正しいことがわかる．

2.2 符号化・復号

ICN-NC では，送信ノードが Encode，中間ノードが Re-encode，受信ノードが Decode の処理をする．それぞれの処理を以下に示す．

【Encode】

配信データ$\boldsymbol{x}$をk個のシンボル$\boldsymbol{x} = (x_1, x_2, \cdots, x_k)$に分割する．送信ノードは以下の手順を$k$回繰り返す．

Step 1. 符号化ベクトル$\boldsymbol{c}$をランダムに生成する．

$$\boldsymbol{c} = (c_1, c_2, \ldots, c_k)$$

Step 2. シンボル$x_1, x_2, \cdots, x_k$と符号化ベクトル$\boldsymbol{c}$を用いて符号化データyを作成する．

$$y = \boldsymbol{c}\boldsymbol{x} = c_1x_1 + c_2x_2 + \cdots + c_kx_k$$

Step 3. 符号化データyと符号化ベクトル$\boldsymbol{c}$から符号化パケットを作成する．

【Re-encode】

中間ノードは，キャッシュ内のm個の符号化パケットを用いて以下の処理を行う．

Step 1. 符号化ベクトル$\boldsymbol{c'}$をランダムに生成する．

$$\boldsymbol{c'} = (c_1, c_2, \ldots, c_m)$$

Step 2. キャッシュ内のm個の符号化データ$\boldsymbol{y}$と符号化ベクトルを用いて再符号化データzを作成する

$$z = \boldsymbol{c'y} = c_1y_1 + c_2y_2 + \cdots + c_my_m$$

Step 3. シンボル$x_1, x_2, \cdots, x_N$を復号できるように符号化ベクトル$\boldsymbol{c'}$とキャッシュ内の符号化データ$\boldsymbol{y}$の符

号化ベクトル行列$\boldsymbol{C}$を用いて符号化ベクトル $\boldsymbol{c}''$を作成する．

$$\boldsymbol{c}'' = \boldsymbol{c}'\boldsymbol{C}$$

Step 4. 符号化データzと符号化ベクトル$\boldsymbol{c}$から符号化パケットを作成し，キャッシュに保存する．

【Decode】

受信ノードはk個以上の符号化パケットを受信したとき，k個のパケットを用いて以下の処理を行う．

Step 1. 受信した符号化データと符号化ベクトル行列を用いて連立方程式を解くことによりシンボル$x_1, x_2, \cdots, x_k$を復号する．

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,k} \\ c_{2,1} & c_{2,2} & & c_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{k,1} & c_{k,2} & \cdots & c_{k,k} \end{pmatrix}^{-1} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix}$$

Step 2. 復号したシンボル$x_1, x_2, \cdots, x_N$から配信データ$\boldsymbol{x}$を復元する．

2.3 ICN-NC の通信モデル

ICN-NC の通信モデルは送信処理を行う送信ノード，中継処理を行う中間ノード，受信処理を行う受信ノードから構成される．ICN-NC でのコンテンツ取得の流れを図 3 のシーケンス図で示す．

2.3.1 受信ノード

コンテンツはk個のシンボルに分割し符号化されており，受信ノードはコンテンツを取得するために k 個の符号化パケットを受信したい．受信ノードは Interest に取得したいコンテンツの name に加えてr，vを含める．rは復号に必要な符号化パケット数である．これにより，所望の個数の符号化パケットを集める．符号化パケットには，Interest を送信したノードから Interest に応答するノードまでに Interest が中継された回数 L が ACK として含まれている．vはLに 1 加算した値である．vは Interest が中継される度に 1 減算される．vが 1 のとき，Interest を受信したノードは Interest に応答する．そうすることで，同じノードが Interest に応答することを避け，重複した符号化パケットの受信を減らせる．受信ノードのコンテンツ取得までの処理を示す．受信ノードは k 個の符号化パケットが欲しいのでrの初期値はkとする．vの初期値は 1 とする．

【コンテンツを取得するまでの処理】

Step 1. Interest にr，vを含めて送信する．

Step 2. p 個の符号化パケットを受信する．v，rを更新する．

$$v = L + 1$$
$$r = r - p$$

Step 3. 復号に必要な符号化パケットの数を受信していない場合，Step1 へ．

Step 4. 受信したk個の符号化パケットを用いて，Decode を行う．

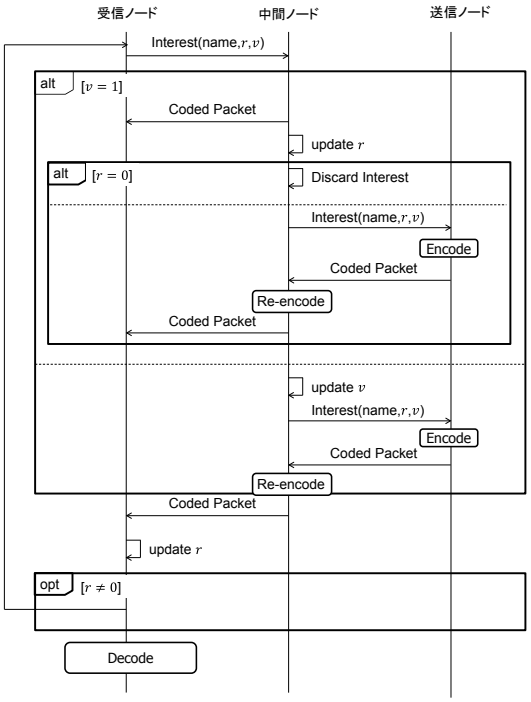

図 3 ICN-NC シーケンス図

2.3.2 中間ノード

中間ノードはキャッシュ機能を持つ．Re-encode した符号化パケットをキャッシュに保存する．

中間ノードは Interest に含まれるvが 1 でない場合，自身は Interest に応答するノードではないため，Interest を中継する．Interest に含まれるvが 1 である場合，Interest に応答する．中間ノードの Interest への応答処理を示す．中間ノードは Interest に対応するコンテンツの符号化パケットをp個キャッシュ内に保持している．ノードが中継した Interest のrを保存しておくr_mを持つとする．r_mの初期値は 0 である．r_mは中間ノードが要求された符号化パケット数の最大値である．

【Interest への応答処理】

1. Interest のr以上の符号化パケットを保持する場合

Step 1. Interest の name に対応する p 個の符号化パケットを送信する．

Step 2. Interest を捨てる．

2. Interest のr未満の符号化パケットを保持する場合

Step 1. Interest の name に対応する p 個の符号化パケットを送信する．rを更新する．

$$r = r - p$$

Step 2. 既に要求している符号化パケットの数r_mより，rが大きければ，Interest を中継する．r_mを更新する．

$$r_{\mathrm{m}} = r$$

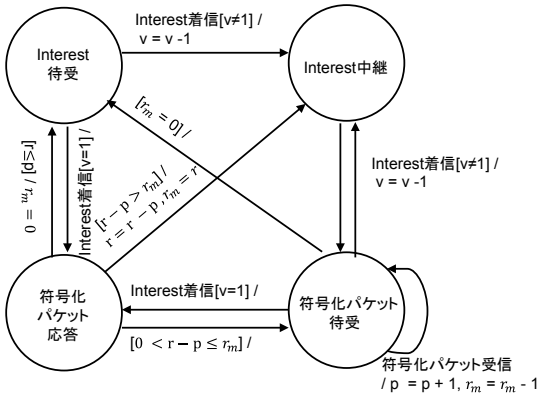

図 4 ICN-NC　中間ノードの状態遷移図

中間ノードの状態遷移図を図 4 に示す．始め中間ノードは Interest 待受状態である．受信した Interest に含まれるvが 1 である場合，符号化パケット応答状態になる．中間ノードは Interest の name に対応するp個の符号化パケットで Interest に応答する．受信ノードが所望している r 個の符号化パケットに応答できれば，Interest を破棄し，Interest 待受状態になる．受信ノードが所望している r 個の符号化パケットに応答できない場合，応答できなかった$r-p$個の符号化パケットと自身が持つr_mを比較する．$r-p$がr_mより大きい場合，r，r_mを応答できなかった符号化パケットの個数$r-p$に更新する．その後，Interest 中継状態になり，応答できなかった数rを Interest に含めて中継する．Interest の中継が終わると符号化パケット待受状態になる．$r-p$がr_mより小さい場合，既に応答できなかった符号化パケット数以上の符号化パケットを他のノードに要求しているので，符号化パケット待受状態になる．符号化パケット待ち受け状態で，符号化パケットを受信すると自身が持つ符号化パケットの数pを 1 増やし，自身が所望している符号化パケット数r_mを 1 減らす．r_mが 0 になれば，Interest の応答を終え，Interest 待ち受け状態となる．符号化パケット待ち受け状態で Interest を受信し，vが 1の場合， p 個の符号化パケットで Interest に応答する．vが 1でない場合，vの値を減らし Interest を中継する．

2.3.3 送信ノード

送信ノードの Interest への応答処理を示す．

【Interest への応答処理】

Step 1.　Encode を行い，Interest の name に対応する r 個の符号化パケットを送信する．

3. 提案方式：Information-Centric Networking with Built-in LT Network Codes（ICN-LTNC）

ICN-NC は ICN アーキテクチャで RLNC を動作させることでキャッシュを高度に利用できる．しかし，RLNC が復号で行う連立方程式を解く処理は計算量$O(m \cdot k^2)$が大きい．

Content Name
Selector
Nonce
Interest Type
Vector

図 5 ICN-LTNC　Interest パケットフォーマット

Signature
Name
SignedInfo
Signature_ID
Gen_ID
Coding_Vector
Degree
Message_ID
Content

図 6 ICN-LTNC　Data パケットフォーマット

そこで本稿では，ICN アーキテクチャで復号の計算量が少ない LTNC が動作するネットワークアーキテクチャを提案する．LTNC を用いることで，受信ノードの復号に要する計算量は RLNC の$O(m \cdot k^2)$から$O(m \cdot k\log k)$に削減できる[8]．なお，LTNC は復号に$n(n > k)$個のパケットを用いるため，RLNC より多くのパケットを受信する必要がある．文献[7]では，$n = k + O(\log^2(k/\delta)\sqrt{k})$ $(\delta \in [0,1])$個のパケットを受信し，復号を失敗する確率は最大でδといわれている．

ICN-LTNC は送信ノード，キャッシュ機能を持った中間ノード，受信ノードから構成される．

3.1 パケットフォーマット

ICN-LTNC は，Interest パケット，Data パケットの 2 種類のパケットがある．Interest パケットはコンテンツの要求に使用する．Data パケットは要求されたコンテンツを返すのに使用する．ICN－LTNC のパケットフォーマットは ICN-NC のパケットフォーマットを基としている．

Interest パケットのパケットフォーマットを図 5 に示す．ICN-LTNC では，Interest パケットは Content Name の他に unreceive symbols もしくは connected component を用いることでコンテンツの取得を行う．両方ともデータの分割数 k の場合， kByte あれば表記できるので，Interest パケットフォーマットに kByte の Vector を追加する．また，どちらを利用しているかを区別するため 1Byte の Interest Type を追加する．

Data パケットのパケットフォーマットを図 6 に示す．

ICN-NC 同様，符号化により Content が変化し ICN 同様の Signature は作成できないため，Signature_ID，Message_ID を利用する. LT 符号を使用するため Gen_ID, Coding_Vector の他に符号化データに使用されているシンボルの数を示す Degree を 1byte 設けている．

3.2 符号化・復号

ICN-LTNC では，送信ノードは Encode，受信ノードでは Decode を行う．それぞれの処理を示す.

【Encode】

コンテンツのデータ$\boldsymbol{x}$をk個のシンボル$\boldsymbol{x} = (x_1, x_2, \cdots, x_k)$に分割する．送信ノードは以下の符号化処理を繰り返す.

Step 1. ある確率分布に従い，符号化に使用するシンボルの数である次数 d を選ぶ.

Step 2. k個のシンボルからd個のシンボルを一様ランダムに選ぶ.

Step 3. 選んだ d 個のシンボルの排他的論理和をとり，符号化データ y を作成する.

$$\mathrm{y} = \bigoplus_{i=1}^{k} \delta_i x_i, \delta_i \in \{0,1\}$$

Step 4. 符号化データ y と次数 d と符号化ベクトル$\{0,1\}^k$から符号化パケットを作成する.

【Decode】

受信ノードは符号化パケットを受信したとき，以下の処理を行う.

1. 受信した符号化データの次数が 1 の場合

Step 1. 受信したシンボルを復号する.

Step 2. 受信バッファの次数 2 以上の符号化データに排他的論理和をとり，次数を 1 下げる.

Step 3. 受信バッファの次数が下がった符号化データからシンボルが復号できた場合，step 2 へ.

Step 4. 全てのシンボル$(x_1, x_2, \cdots, x_k)$を復号し，xを復元する.

2. 受信した符号化データの次数が 2 以上の場合

Step 1. 符号化データと符号化データに使用されている復号したシンボル$x_1, x_2, \cdots, x_i$の排他的論理和をとり，次数を下げる.

Step 2. 符号化データの下がった次数が 1 ならば, 1.を行い，2 以上ならば，受信バッファに入れる.

3.3 ICN-LTNC の通信モデル

ICN-LTNC の通信モデルは送信処理を行う送信ノード，中継処理を行う中間ノード，受信処理を行う受信ノードから構成される．ICN-LTNC のノードは Encoded packet by degrees と Connected Symbols のデータ構造を持つ．Connected Symbols とは，次数 2 以下の符号化パケットの組み合わせによって$x \oplus x'$が作成できるシンボルx, x'のことであり，$x \sim x'$と表記する．例として，符号化データ $\mathrm{y}_1 = x_1 \oplus \mathrm{x}_2, y_2 = x_2 \oplus x_3$を保持している場合を考える．$\mathrm{y}_1$より$x_1, x_2$，$\mathrm{y}_2$より$x_2, x_3$は Connected Symbols である.

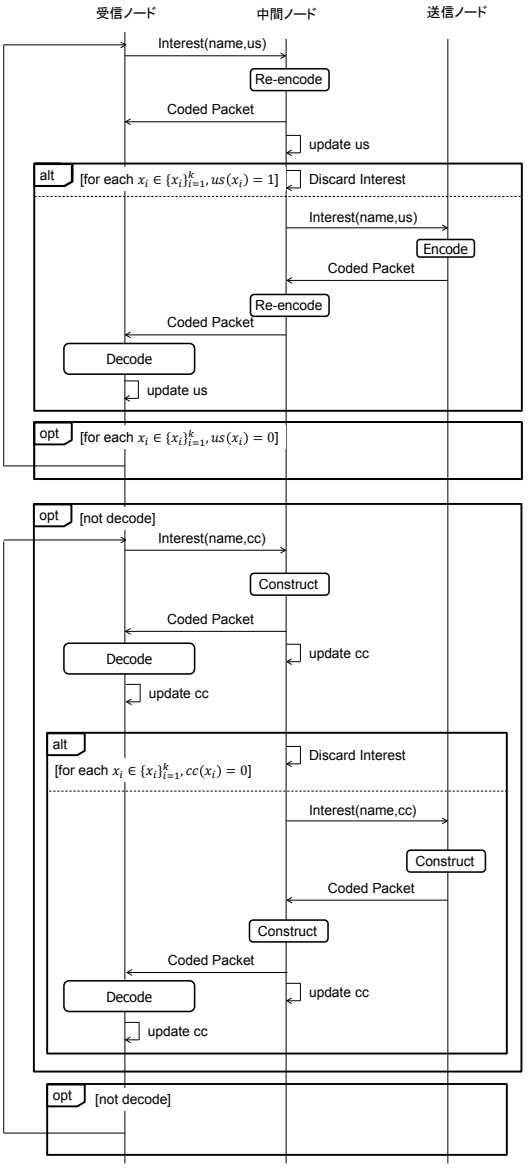

図 7 ICN-LTNC　シーケンス図

$\mathrm{y}_1 \oplus \mathrm{y}_2 = x_1 \oplus x_3$より，$x_1, x_3$は Connected Symbols なので，$x_1, x_2, x_3$は Connected Symbols である．また，Connected Symbols を表すために connected components of symbols(cc)を定義する．$\mathrm{cc}(x_i)$の初期値はiとし，復号したシンボルx_jの$\mathrm{cc}(x_\mathrm{j})$の値は 0 とする．$x \sim x'$の場合，$\mathrm{cc}(x) = \mathrm{cc}(x')$となる.

ICN-LTNC のコンテンツ取得の流れを図 7 のシーケンス図で示す.

3.3.1 受信ノード

受信ノードは，取得したいコンテンツの name を含めた Interest を送信し，それに応じたパケットを受信することでコンテンツを取得する．ICN-LTNC では，受信したシンボルに応じて 2 種類の Interest を用いる．ICN-LTNC では，分割された k 個のシンボルを少なくともそれぞれ一度以上受信すればデータを復元できる可能性がある．そこで，受信ノードは未受信のシンボルを 1, 受信しているシンボルを 0

とする$\{0,1\}_{i=1}^{k}$（unreceived symbols，us）を含めた Interest を送信することで復号に必要な符号化パケットを受信することを目指す．us を含めた Interest を用いたコンテンツ取得の流れを示す．

【us を利用したコンテンツを取得するまでの処理】

Step 1. コンテンツの name と us を Interest に含めて送信する．

Step 2. 符号化パケットを受信し，Decode を行う．全てのシンボルを受信していなければ Step1 へ．

受信ノードは，全てのシンボルを一度以上受信してもデータを復号できないことがある．例としては，データがシンボル(x_1, x_2, x_3)に分割されている場合，符号化データ$y_1 = x_1, y_2 = x_2 \oplus x_3$を受信すれば，全てのシンボルを一度は受信しているが，シンボルx_1しか復号できない．そのため，us を含めた Interest だけでは，コンテンツを取得できない可能性がある．

LTNC では，全てのシンボルが繋がっている，すなわち全てのシンボルの cc の値が同じである場合，次数 1 の符号化パケットが 1 個あれば，データを復号できる．そのため，受信ノードは cc の値が全て同じになるように cc の値を更新することを目指す．受信ノードの cc の値を更新する次数 2 以下のパケットを innovative パケットと定義する．cc を含めた Interest を用いたコンテンツ取得の流れを示す．受信ノードの cc を$\mathrm{cc_r}$とする．

【cc を利用したコンテンツを取得するまでの処理】

Step 1. コンテンツの name と$\mathrm{cc_r}$を Interest に含めて送信する．

Step 2. 符号化パケットを受信し，Decode を行う．全シンボルの$\mathrm{cc_r}$の値が 0 でない場合，Step1 へ．

受信ノードは全てのシンボルの cc の値が 0 になることでデータを復号し，コンテンツを取得することができる．

3.3.2 中間ノード

中間ノードはキャッシュ機能を持つ．受信した符号化パケットを Decode しキャッシュに保存する．ノードはキャッシュにある符号化パケットを使用し Interest に応答する．

受信ノードから us を含めた Interest を受信した場合の動作を示す．中間ノードは us を含めた Interest を受信した場合，キャッシュにある符号化パケットを利用し Re-encode を行い応答する．Re-encode には図 8 に示す Build[8]，図 9 に示す Refine のアルゴリズムを利用する．Build では次数dとデータ構造 Encoded packet by degrees である$\mathcal{S}$を入力することで次数dの符号化パケットzを作成する．

図 10 に Build のアルゴリズムの実行例を示す．$d(y_i)$で符号化パケットに含まれるシンボルの数である次数を返す．この例では Build には入力として集合$S = \{y_1, y_2, y_3, y_4, y_5, x_5\}$と次数$d = 4$が与えられている．始めに次数 4 の符号化パケットがあるかをチェックする（図 10 中①）．この例では次数 4 の符号化パケットはSに含まれないので，次数を 1 下げて，次数 3 の符号化パケットの中からランダムに 1 つ選択する．この例では$\mathrm{y_2}$が選ばれたものとする．次に，y_2と排他的論理和を計算した結果，次数が 4 になる符号化パケットを探索する．次数 3 の符号化パケットy_3を選択し，$d(y_2 + y_3)$を計算する（図 10 中②）．$d(y_2 + y_3) \neq 4$なので，探索は継続する．次数 3 の符号化パケットはすべて探索し終えたので，字数を 1 下げて次数 2 の符号化パケットを探索する．次数が 2 である符号化パケットy_4，y_5とそれぞれ$d(y_2 + y_4) = 1$，$d(y_2 + y_5) = 3$を計算する(図 10 中③と④)．$d(y_2 + y_4) \neq 4$，$d(y_2 + y_5) \neq 4$な

Input: d,$\mathcal{S}$
Output: z
1:$z \leftarrow 0$
2:$i \leftarrow d$
3:$\mathcal{S}' \leftarrow \mathcal{S}[i]$
4:**while** $d(z) < d$ and $i > 0$ **do**
5: **if** $\mathcal{S}' = \emptyset$ **then**
6: $i \leftarrow i - 1$
7: $\mathcal{S}' \leftarrow \mathcal{S}'[i]$
8: **else**
9: $y \leftarrow$ uniform randomly select from $\mathcal{S}'$
10: $\mathcal{S}' \leftarrow \mathcal{S} \setminus \{y\}$
11: **if** $d(z) < d(x \oplus y) \leq d$ **then**
12: $z \leftarrow z \oplus y$
13: **end if**
14: **end if**
15:**end while**

図 8 アルゴリズム Building an encoded packet of a given degree

Input: z,us
Output: z'
1: $z' \leftarrow z$
2: **for each** $x \in z$ **do**
3: $\mathcal{A} \leftarrow \{x''$ s.t. $x \sim x'$ and $x'' \notin z'$ and $us(x'') = 1\}$
4: **if** $\mathcal{A} \neq \emptyset$ **then**
5: $x' \leftarrow$uniform randomly select from $\mathcal{A}$
6: $z' \leftarrow z' \oplus (x \oplus x')$
7: $us(x') = 0$
8: **end if**
9: **end for**

図 9 アルゴリズム Refining an encoded packet

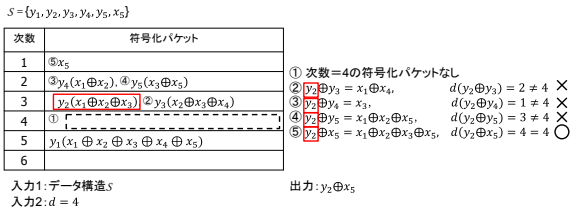

図 10 Build の実行例

ので，さらに次数を下げ次数 1 のパケットを探索する．$d(y_2 + x_5) = 4$である（図 10 中⑤）ため，最終的に Build は$y_2 + x_5$を出力する．

Refine では Build で作成された符号化パケット z, Interest に含まれる us を入力することで, 未受信のシンボルを符号化パケットに含めるためのアルゴリズムである．

【us を含めた Interest への応答処理】

Step 1. ある確率分布に従い次数dを選ぶ.

Step 2. Build を行い符号化パケットzを作成する．

Step 3. Refine を行い符号化パケットz'を作成する．

Step 4. 次数dと符号化ベクトル$\{0,1\}^k$，符号化データをパケットで送信する．符号化データに含まれるシンボルの us を 0 にする．

Step 5. 全てのシンボルの us が 0 の場合, Interest を捨てる．全てのシンボルの us が 0 でない場合，us を含めた Interest を中継する．

受信ノードから cc を含めた Interest を受信した場合の動作を示す． 中間ノードは，Interest に含まれる受信ノードの$\mathrm{cc_r}$を更新するために，$\mathrm{cc_r}(x) \neq \mathrm{cc_r}(x')$となる innovative データ$x \oplus x'$を送信することを目指す. Interest に応答するノードのccを$\mathrm{cc_s}$とする．図 11 は次数d，$\mathrm{cc_r}$，$\mathrm{cc_s}$を入力することで innovative データを作成する Construct のアルゴリズムである[8].

【cc を含めた Interest への応答処理】

Step 1. Construct を行い，符号化データを作成する．

Step 2. 次数d，符号ベクトル$\{0,1\}^k$，符号化データをパケットで送信する．$\mathrm{cc_r}$を更新する．

中間ノードの状態遷移図を図 12 に示す．中間ノードは Interest を受信するまで Interest 待受状態である．us を含めた Interest を受信すると us 符号化パケット応答状態となる．符号化パケットで応答し全てのシンボルの us の値が 0 の場合，Interest への応答を終了し，Interest 待機状態になる．全てのシンボルの us の値が 0 にならない場合, Interest（us）状態となり，us を含めた Interest を中継する．中継後，us 符号化パケット待受状態となる．us を含めた Interest を受信した場合，us 符号化パケット応答状態になる．符号化パケットを受信した場合，自身の$\mathrm{cc_s}$を更新する．Interest 待機状態，us 符号化パケット応答状態，us 符号化パケット待受状態のとき，$\mathrm{cc_r}$を含めた Interest を受信すると cc 符号化パケット応答状態となる．符号化パケットで応答し，全てのシンボルの$\mathrm{cc_r}$の値が 0 の場合，Interest への応答を終了し，Interest 待受状態となる．全てのシンボルの$\mathrm{cc_r}$の値が 0 でない場合，Interest（$\mathrm{cc_r}$）中継状態となり，$\mathrm{cc_r}$を含めた Interest を中継する．中継後，$\mathrm{cc_r}$符号化パケット待受状態となる．$\mathrm{cc_r}$を含めた Interest を受信すると cc 符号化パケット応答状態となる．符号化パケットを受信した場合，自身の$\mathrm{cc_s}$を更新する．

Input: d, cc_r, cc_S
Output: $\mathcal{Y}$
1:$\sigma\{0, \dots, k\} \mapsto (\perp, \perp)$
2 **if** $d = 1$ **then**
3: **for each** $x \in \{x_i\}_{i=1}^{k}$ **do**
4: **if** $cc_s = 0$ and $cc_r \neq 0$ **then**
5: $\mathcal{Y} \cup \{x_i\}$
6: **end if**
7: **else if** $d = 2$ **then**
8: **for each** $x \in \{x_i\}_{i=1}^{k}$ **do**
9: $(j, x) \leftarrow \sigma[cc_s(i)]$
10: **if** $j = \perp$ **then**
11: $\sigma[cc_s(i)] \leftarrow (cc_r(i), x_i)$
12: **else if** $j \neq cc_s(i)$ **then**
13: $\mathcal{Y} \cup \{x \oplus x_i\}$
14: **end if**
15: **end for**
16: **end if**

図 11 アルゴリズム Construct an innovative data

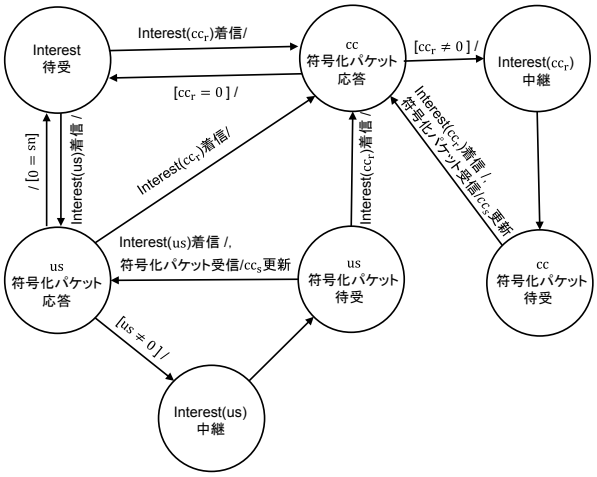

図 12 ICN-LTNC 中間ノードの状態遷移図

3.3.3 送信ノード

送信ノードの Interest への応答の処理を示す.

【us を含めた Interest への応答処理】

Step 1. Encode を行い，Interest の name に対応する符号化パケットを送信する．

【cc を含めた Interest への応答処理】

Step 1. Construct を行い，符号化データを作成する．

Step 2. 次数d，符号ベクトル$\{0,1\}^k$，符号化データをパケットで送信する．$\mathrm{cc_r}$を更新する．

4. シミュレーション評価

4.1 実装

ICN-LTNC と ICN-NC を比較するシミュレーションはネットワークシミュレータ OMNeT++4.4.1[10], 移動体ネットワークシミュレーション用のフレームワーク Veins 3.0[11], トラフィックシミュレータ SUMO 0.21.0[12]を用いる．シミュレーションフィールドは神戸・三宮とする．ノードは移動体である車両とし，ノード数は 50 とする．roadside

unit(RSU)はコンテンツを 1 個所持している．コンテンツの分割数は 50 である．ノードは RSU にコンテンツの要求を行う．通信規格は IEEE802.11p，データの送信周期は 100ms とする．

両方式の性能を確認するため経過時間に対するコンテンツを取得したノード数，受信ノードの復号に要した計算回数を評価する．

4.2 結果

図 13 は経過時間に対するコンテンツを取得したノード数を示している．図 13 より，全てのノードがコンテンツを取得するまでの時間は．ICN-NC は，ICN-LTNC より短いことがわかる．ICN-NC が ICN-LTNC より短い時間で全てのノードがコンテンツを取得できた理由は，復号に要するパケット数が少ないからだと考えられる．ICN-NC で用いられている RLNC は，データの分割数であるk個のパケットを受信すれば復号できる．一方，ICN-LTNC で用いられている LTNC は，復号するのに k より大きい$n(= k + O(\log^2(\mathrm{k}/\delta)\sqrt{k})$ $(\delta \in [0,1])$個のパケットを要する．よって，配信性能は ICN-NC は ICN-LTNC に比べて高いといえる．

図 14 は，復号に要した計算回数をログスケールで示している．図 14 より，ICN-LTNC は，ICN-NC に比べて復号に要した計算回数が小さいことがわかる．計測終了時には，ICN-LTNC は，ICN-NC の約 1 万分の 1 の計算回数となる．これは，復号の処理が違うことが原因である．ICN-LTNC は，LT 符号を使用するので符号化ベクトルが 1 と 0 で構成されており，排他的論理和のみを使用し復号できる．一方，ICN-NC は，k 行 k 列の符号化行列に減乗除算を使用し，連立方程式を解くことで復号するからである．

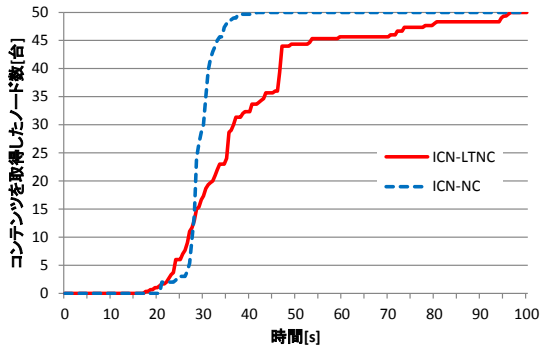

図 13 経過時間に対するコンテンツを取得したノード

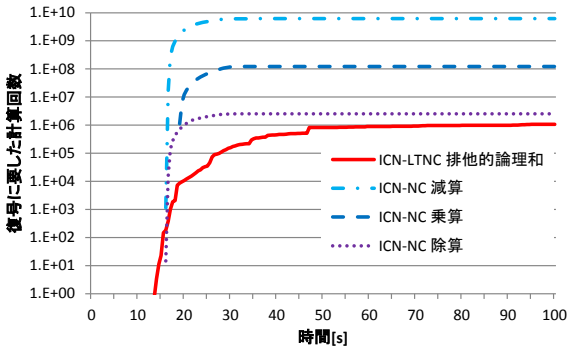

図 14 経過時間に対する復号に要した計算回数

5. おわりに

本稿では，ICN のアーキテクチャで復号の計算量が少ないネットワーク符号化である LTNC が動作する ICN-LTNC を設計した．トラフィックシミュレータと連携したネットワークシミュレーションにより，ICN-LTNC が VANET でも動作可能であることを確認した．ICN-LTNC は，復号の計算量が多い RLNC を適用した ICN-NC より，配信性能は劣るが復号に要した計算回数が少ないことを確認した．

参考文献

[1] 伊藤章, 福田健一. ICN が切り開く次世代ネットワークアーキテクチャー，Fujitsu, Vol.66, No.5, pp. 53-61 (2015).
[2] Xylomenos, G., Ververidis, C.N., Siris, V.A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K.V., Polyzos, G.C.: A survey of information-centric networking research, IEEE Communications Surveys & Tutorials, vol.16, no.2, pp.1024-1049, (2014).
[3] Bruno, F., Cesana, M., Gerla, M., Mauri, G., Verticale, G.: Optimal Content Placement in ICN Vehicular Networks, 2014 International Conference and Workshop on the Network of the Future, pp.1-5,(2014).
[4] Tian, H., Mohri, M., Otsuka, Y., Shiraishi, Y., Morii, M.: LCE In-Network Caching on Vehicular Networks for Content Distribution in Urban Environments, 2015 Seventh International Conference on Ubiquitous and Future Networks, pp.551-556,(2015).
[5] Ho, T., Médard, M., Koetter, R., Karger, R., Effros, M., Shi, J., and Leong, B.: A Random Linear Network Coding Approach to Multicast, IEEE Transactions on Information Theory, vol.52, no.10, pp.4413-4430, (2006).
[6] Liu, W. X., Yu, S. Z., Tan, G., and Cai, J.: Information-centric networking with built-in network coding to achieve multisource transmission at network-layer, Computer Networks, (2015), http://dx.doi.org/10.1016/j.comnet.2015.05.009
[7] Li, C., Jose, J., and Wu, X.: Distributed-Fountain Network Code (DFNC) for Content Delivery in Vehicular Networks, the tenth ACM international workshop on Vehicular inter-networking, systems, and applications, pp.31-40, (2013)..
[8] Champel, M.L., Huguenin,K, Kermarrec, M.A., Le Scouarnec, N.: LT network codes, IEEE 30th International Conference on Distributed Computing Systems (ICDCS 2010), pp.536–546, (2010).
[9] 朴容震:情報指向ネットワーキングの研究動向， GITS/GITI Research Bulletin, pp.8-13(2013).
[10] OMNeT++, available form <http://www.omnetpp.org/>, (accessed 2016-02-10).
[11] Veins, available form <http://veins.car2x.org/>, (accessed 2016-02-10).
[12] Simulation of Urban MObility, available form <http://sumo.sourceforge.net/>, (accessed 2016-02-10).