ID/Locator分離とLocator分割に基づくネットワークスライシングにおけるDiffservによる資源隔離

永山 裕人[1,a)] 渡邊 大記[1,b)] 寺岡 文男[2,c)]

概要：第 5 世代移動通信システム (5G) では, ネットワークスライシング技術により単一の物理ネットワーク上に複数の論理ネットワークを構築することにより複数の異なるネットワーク要件を満足させる. ネットワークスライス内のネットワーク資源はそれぞれで隔離された状態となる. ネットワークスライス構築の既存手法にはホップバイホップ方式とエッジオーバレイ方式が存在する. 筆者らは ID/Locator 分離と Locator 分離に基づく方法を提案している. 資源隔離においては, 以前は Multi-Protocol Label Switching (MPLS) を利用していたが実装の複雑さゆえ, 本稿は Diffserv を利用する. さらにスライス構築管理システムも実装した. 資源隔離の Diffserv 実装とはパケット内のヘッダの一部にマーキングした値を用いることでネットワーク内の規定保証帯域の割当をパケットが受けることである. 具体的には, IPv6 パケット内のトラフィッククラスフィールドの上位 6 ビットを Differentiated Service Code Point (DSCP) 値というマーキング値として用い, ネットワークスライス構築ルータは DSCP 値に応じた資源を各パケットに切り分ける. 提案方式を Linux に実装し, 資源隔離が実現できていることを確認した.

Resource Isolation by Diffserv in Network Slicing based on ID/Locator Separation and Locator Division

Yuto Nagayama[1,a)] Watanabe Hiroki[1,b)] Teraoka Fumio[2,c)]

1. はじめに

第 4 世代移動通信システム (LTE-Advanced) では単一ネットワークで高速, 低遅延等様々なネットワーク要件を満たそうとするが, それは困難である. 第 5 世代移動通信システム (5G) では, ネットワークスライシング技術により単一の物理ネットワーク上に複数の論理ネットワークを構築することでネットワーク要件を満足させる. ネットワークスライス内のネットワーク資源はそれぞれで隔離された状態となる.

ネットワークスライス構築の既存手法にはホップバイホップ方式とエッジオーバレイ方式が存在する. ホップバイホップ方式では Software Defined Network (SDN) コントローラと SDN スイッチによる中央制御でネットワークスライスを構築する. ネットワークスライス構築の柔軟性が利点であるが, SDN コントローラが単一障害点になりえる点, 既存物理ネットワークの流用が困難な点が問題である. エッジオーバレイ方式では仮想ネットワークコントローラ, 仮想スイッチによるトンネリング処理でネットワークスライスを構築する. 既存ネットワークを流用できる点, 単一障害点が無い点が利点であるが, ネットワークスライス構築に柔軟性が無い点, Ethernet フレームのヘッダのオーバヘッドが大きくなる点が問題である. これらに対して筆者らは Locator/ID 分離モデルの IPv6 アドレス内に所属ネットワークスライス情報を埋め込み, MPLS でスライス間の資源を分離することでネットワークスライスを構築する. 単一障害点が無い, 既存ネットワークを流用できる, ヘッダのオーバヘッドがないといった利点がある. 文献 [1] では資源隔離のために MPLS を利用していたが,

[1] 慶應義塾大学大学院理工学研究科
Graduate School of Sciense and Technology, Keio University
[2] 慶應義塾大学理工学部
Faculty of Sciense and Technology, Keio University
[a)] nem@inl.ics.keio.ac.jp
[b)] nelio@inl.ics.keio.ac.jp
[c)] tera@keio.jp

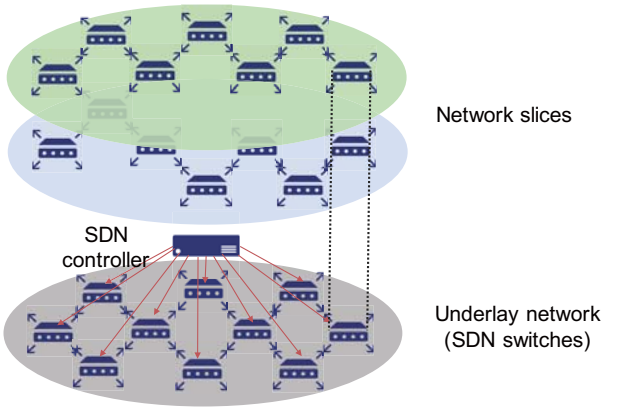

図 1　ホップバイホップ方式.(文献 [1] から引用)

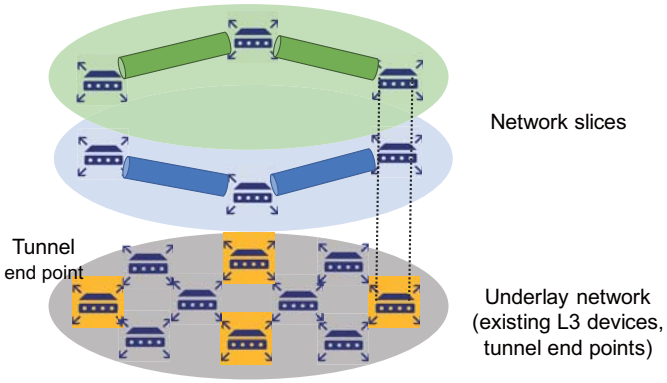

図 2　エッジオーバレイ方式.(文献 [1] から引用)

実装が複雑であった.

本稿では提案手法の利点を受け継ぎながら, 資源隔離を Diffserv 実装することで簡易なスライスの実装を提案する.

2. 既存のスライシング手法

2.1 ホップバイホップ 方式

ホップバイホップ方式では図 1 のように SDN コントローラと SDN スイッチによってネットワークスライスを構築する. より具体的には, SDN コントローラが 全 SDN スイッチに対してパケットの受け入れ条件と転送ルールを的確に設定することで構築される. 用いられる SDN 技術の代表例には OpenFlow [2] があり, SDN コントローラは OpenFlow Controller (OFC) , SDN スイッチは OpenFlow Switch (OFS) となる.

この手法における問題点を以下に示す.

- アンダーレイネットワーク (ネットワークスライスの構築基盤となる物理ネットワーク) がスイッチと SDN コントローラで構成されるため, 既存のネットワークシステムからの移行が困難である.
- SDN コントローラが SDN スイッチを集中制御するため, SDN コントローラが単一障害点となる.
- 各 SDN スイッチがそれぞれ異なるフローテーブルに従って動作するため, ネットワーク障害発生時の障害原因の分析, 切り分けが困難である.

2.2 エッジオーバレイ方式

エッジオーバレイ方式では図 2 のように L3 層 (ネットワーク層) の上にトンネリングによって仮想的な L2 層 (データリンク層) のリンクを設ける形でネットワークスライスを構築する. 用いられるトンネリング技術の代表例には Virtual eXtensible Local Area Network (VXLAN) [3] や Network Virtualization using Generic Encapsulation (NVGRE) [4] が存在する.

この手法における問題点を以下に示す.

- Ethernet フレームのヘッダにおけるオーバヘッドが大きい. 例として, VXLAN によるトンネリングと MPLS による資源隔離を用いた際を挙げると, 74bytes のオーバヘッドが発生する.
- VXLAN では, パケットのカプセル化の際に UDP ヘッダを用いるため, TCP Segmentation Offload (TSO) や Large Receive Offload (LRO) のような TCP オフロード機能を使用できない. TCP オフロード機能が有効な環境においては, トンネリング処理なしの場合と比較してスループットが低下する [5].

2.3 ID/Locator 分離と Locator 分割を用いたネットワークスライス構築

ホップバイホップ方式, エッジオーバレイ方式の問題点を解決するべく, 筆者らは ID/Locator 分離と Locator 分割によるネットワークスライス構築手法を提案している (先行研究 [1]). 図 3 に筆者らが提案する IPv6 アドレス形式を示す. このアドレスにおいては, 上位 64bits がノードの接続場所を意味する Locator, 下位 64bits がノードの識別子を意味する ID として扱われる. Locator の 5byte 目は Network slice ID を意味し, この IP アドレスを割り振られたノードがどのスライスに所属しているのかを示す. Locator の 6,7 byte 目はネットワークスライス内でこの IP アドレスを割り振られたノードがどのサブネットに所属しているのかを示す. なお，ここに示す Network slice ID のビット長やサブネットのビット長は一例であり，ネットワーク管理者が自由に決めることができる.

ネットワークスライス間の資源隔離は MPLS による資源隔離を利用する. 図 4 のように IPv6 パケットヘッダの前に 4bytes の MPLS ヘッダを挿入し, それをルータが読むことで MPLS による資源隔離をスライスに適用する.

図 5 に先行研究の Linux への実装例を示す. この例ではスライス番号 0x01, サブネット番号 0x0016 に対して MPLS のパスを設定している. 図に示すように 1 つのパスに対して Linux カーネル内に複数のデバイスを設定する必要があり, 実装が複雑であった. また, スライスを構築するための管理機構も未実装であった.

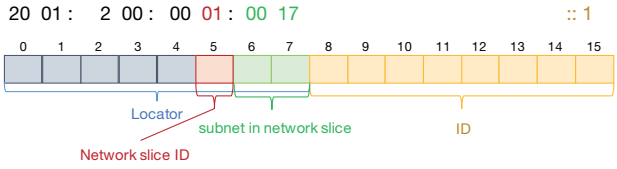

図 3　IPv6 アドレス形式.

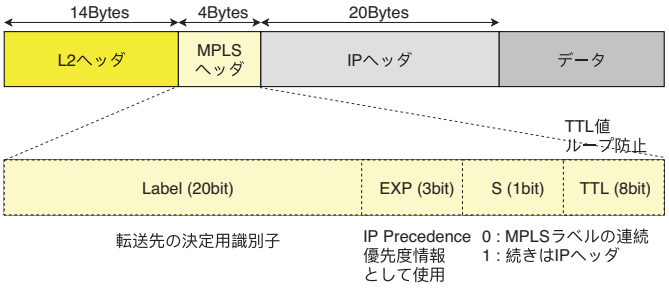

図 4　MPLS ヘッダ.

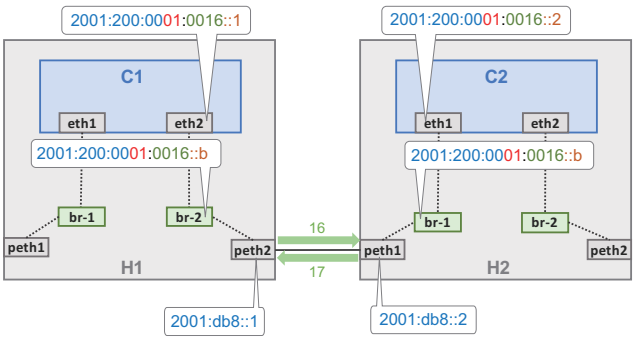

図 5　MPLS による資源隔離でのスライス構築.(文献 [1] から引用)

2.4　本稿の目的

本稿では, 先行研究によるスライス構築法を踏襲し, スライス間の資源隔離には MPLS に代わって Diffserv を利用することにより, 実装の複雑さを軽減することを目的とする. また, スライス構築管理システムのプロトタイプも実装する.

3.　設計

3.1　Diffserv による資源隔離の設計

Diffserv とは, 分類, マーキング, キューイング, スケジューリングの 4 過程でパケットを操作し, CoS (Class of Service) を実現するものである.

IPv6 における Diffserv とは, 図 6 に示すように IPv6 ヘッダのトラフィッククラスフィールドの前 6bits に優先度情報である DiffServ Code Point (DSCP) として使用し, そのDSCP 値に応じてパケットにあらかじめ設定されたサービスクラスを適用する技術である [6]. 本稿ではネットワークスライスを利用するパケットにそのネットワークスライスの持つ優先度に適した DSCP 値を埋め込み, DSCP 値に設定された帯域保証および帯域の上限を受ける. 特定のネットワークスライスにおける DSCP 値は一意であるが, 異なるスライスが同一の DSCP 値を持つこともある. すなわち, ネットワークスライスと DSCP 値の関係は多対一である.

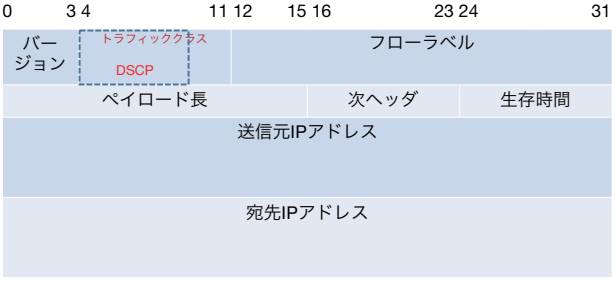

図 6　IPv6 のヘッダフォーマット.

3.2　提案システムの構成

本稿で提案するシステムは, エンドノード, Ingress/Egress ルータ, スライス構築マネージャに分けることができる.

エンドノード上で動作するアプリケーションには, アプリケーションの要件に応じたスライス ID が与えられているものと仮定する.

Ingress/Egress ルータはネットワークスライス領域の出入り口となるルータである. ネットワークスライス領域の外側から内側に転送されるパケットに対して, このルータは Ingress ルータとして振る舞い, 0 である DSCP フィールドをネットワークスライス ID に対応する DSCP 値に変更する. ネットワークスライスの内側から外側に転送されるパケットに対して, このサーバは Egress ルータとして振る舞い, 変更されている DSCP 値を 0 に戻す. ネットワークスライス ID と DSCP 値の対応関係を変更したい場合, この Ingress/Egress ルータの設定に該当するスライス構築マネージャの設定ファイルを変更するだけでよい. 以上の流れを図 7 に示す.

スライス構築マネージャは設定ファイルを読み込み, 各ルータにスライス構成や対応するサービスクラスを設定する.

3.3　ネットワークスライスを構築する経路制御

ネットワークスライス構築時, ルータはネットワークスライスへ参加するもの, 参加しないものに分かれる. このとき, 参加しないルータはネットワークスライス内では経路上に存在しないかのように扱われる必要がある. ネットワークスライス参加ルータ間では L3 ルーティング, 参加ルータと非参加ルータ間や非参加ルータ間では L2 フォワーディングを行う.

4.　スライス構築管理システムの設計

ネットワークスライスの構築をより簡易にすべく, ネットワーク管理者が JSON ファイルで定義したスライス構築

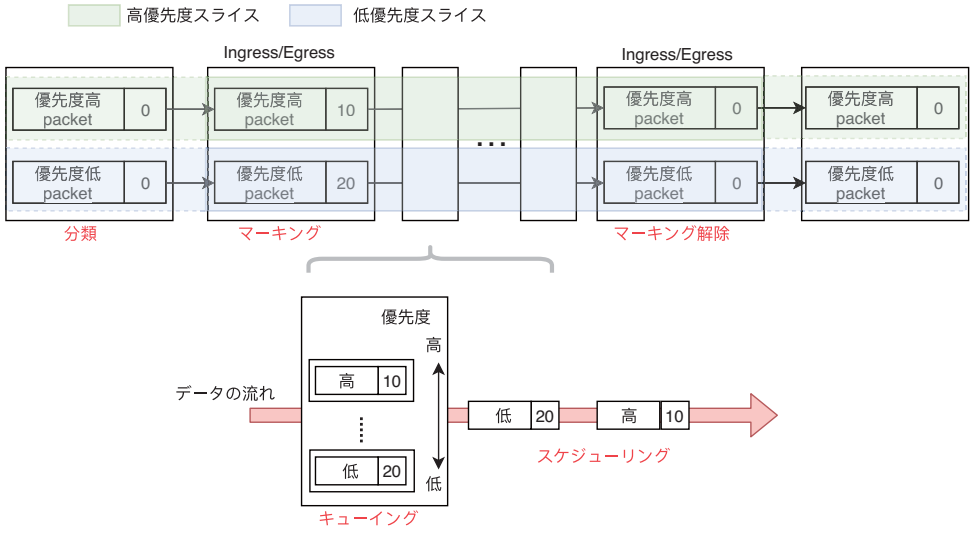

図 7　Diffserv による資源隔離の流れ.

```
"common_prefix": IPv6 アドレスの Locator 共通部分
```

図 8　common_prefix の記述方法.

が反映される機構を設計する.

全体の流れとしては, スライス構築マネージャが JSON で記述された設定ファイルを読み込み, ルータごとの設定ファイルを生成し, 各ルータに設定ファイルを送信する. 各ルータにはスライス構築クライアントが動作しており, スライス構築マネージャから受信した設定ファイルに基づき, スライスの設定を行う. なお, 本稿ではスライス構築クライアントとしてシェルを利用した. 管理者が記述する設定ファイルのフォーマットを以下の節で示す.

4.1　common_prefix

筆者らの提案する IPv6 アドレスは common_prefix, slice_id , subnet_num, および node_id の 4 つの値を連結することで構成される. 図 3 を例に取ると, common_prefx = 2001:200:0000::/40, slice_id = 01, subnet_num = 0017, および node_id = 1 である. slice_id と subnet_num は後述する slices の要素であり, node_id は後述する p_nodes の構成要素である. 4 つの値を連結して構成した IPv6 アドレスは各ルータに紐付けられ, スライスの構築時に経路制御によるスライスの表現に用いられる.

4.2　p_nodes

p_nodes にはネットワーク内に物理的に存在する全ルータの名前, 識別子 (id), インターフェイス名を図 9 のように記載する. 物理的構成が変更となる頻度は低いと考えられるため, ネットワーク管理者は一度ここの記述を書いた後は変更することは少ない部分である.

```
"p_nodes": [{ 物理情報
    "node_name": { ルータの名前
      "node_id" : Loc/ID 分離 IPv6 アドレスの ID,
        "ifs": [ インタフェース名一覧
          "interface_name",
          ...
        ]
    },
    ...
}]
```

図 9　p_nodes の記述方法.

4.3　slices

slices にはスライスの構成情報を図 10 のように記載する. 情報としてはスライス名, スライス識別子, 対応する DSCP 値, スライス内のサブネットに関する情報が含まれる. サブネットの情報はサブネット名, サブネット番号, サブネット間の帯域上限, サブネット参加ルータで構成される. DSCP 値は qos の key 値と対応し, 各スライスの保証帯域等を表現している. slice_00 は物理ネットワークを表現しており, 資源隔離はベストエフォートを意味する DSCP 値 0 が割り当てられている.

4.4　qos

qos には各 DSCP 値ごとに保証帯域 (rate), 上限帯域 (ceil), 優先度 (pri) を図 11 のように記載する. 同じ DSCP 値が記載される slices 内のスライスはこの DSCP 値と同じ資源隔離を受けることとなる. DSCP 値が 0 の資源隔離はベストエフォートであり, pri が 0 と最低である. この際の保証帯域は 0 であり, 上限帯域は各サブネットの max_bw を参照した値である.

4.5　実行例

物理ネットワーク上に図 15 のようにスライスを構築したいとすると, 図 12-14 のようになる.

```
"slices": [{
    "slice_name" : { スライスの名前
        "slice_id" : スライス識別番号
        "dscp_val": スライス内 QoS の対応 DSCP 値,
        "subnets": [{ スライス内のサブネット名一覧
            "subnet_name" : { サブネット名
                "subnet_num" : サブネット番号,
                "max_bw" : サブネット間の帯域上限,
                "p_nodes": [サブネット参加ルータ名
                    "node_name"
                    ...
                ]
            }
            ...
        }]
    },
    ...
}]
```

図 10　slices の記述方法.

```
"qos": [{ QoS 情報
    "dscp_val": { 以下の QoS を表現する DSCP 値
        "rate": 保証帯域,
        "ceil": 上限帯域,
        "prio":  優先度
    },
    ...
}]
```

図 11　qos の記述方法.

5. 実装

5.1 Diffserv による資源隔離の実装

Ingress/Egress ルータ

Ingress ルータでは, IP6tables コマンドでネットワークスライスの中側のデバイスにおける mangle テーブルを変更し, IPv6 アドレスのスライス ID 部分に応じた DSCP 値を IPv6 ヘッダのトラフィッククラスフィールドに設定する.

Egress ルータでは, IP6tables コマンドで IPv6 ヘッダのトラフィッククラスフィールドの DSCP 値を 0 に戻す

ルータ

パケットの DSCP 値に基づきスライスへの資源割当ては Hierarchical Token Bucket (HTB) によって実装される.

HTB とは Linux に存在する Traffic Control (tc) と呼ばれる IP パケットの送信制御機構のキューイング決定を司る Queueing discipline (qdisc) の 1 つである.

tc は qdisc を明示したクラスにパケットを振り分けることでパケットの送信を制御する. HTB におけるクラスは 図 16 のように階層構造をなしている. それぞれのクラスにはそのクラスに振り分けられたパケット郡の保証帯域, 上限帯域, 優先度, qdisc などが設定できる. 本稿での

```
{
    "common_prefix" : "2001:0200:0000::/40"
    "p_nodes" : {
        "Ingress/Egress 1" : {
            "node_id" : "1",
            "ifs" : [
                "eth1"
            ]
        },
        "Ingress/Egress 2" : {
            "node_id" : "2",
            "ifs" : [
                "eth2"
            ]
        },
        "Ingress/Egress 3" : {
            "node_id" : "3",
            "ifs" : [
                "eth3"
            ]
        },
        "R1" : {
            "node_id" : "4",
            "ifs" : [
                "eth4-1", "eth4-2",
                "eth4-3", "eth4-4"
            ]
        },
        "R2" : {
            "node_id" : "5",
            "ifs" : [
                "eth5-1", "eth5-2"
            ]
        },
        "R3" : {
            "node_id" : "6",
            "ifs" : [
                "eth6-1", "eth6-2", "eth6-3"
            ]
        }
    },
    "slices" : {
        "slice_00" : {
            "slice_id" : "00"
            "dscp" : "0",
            "subnets" : [
                "subnet_0001" : {
                    "subnet_num" : "0001",
                    "max_bw" : "500",
                    "p_nodes" : [
                        "Ingress/Egress 1", "R1"
                    ]
                },
                "subnet_0002" : {
                    "subnet_num" : "0002",
                    "max_bw" : "500",
                    "p_nodes" : [
                        "R1", "Ingress/Egress 2"
                    ]
                },
                "subnet_0003" : {
                    "subnet_num" : "0003",
                    "max_bw" : "500",
                    "p_nodes" : [
                        "R1", "R3"
                    ]
                },
```

図 12　スライスを表現する JSON ファイル (1/3).

```
            "subnet_0004" : {
                "subnet_num" : "0004",
                "max_bw" : "500",
                "p_nodes" : [
                    "R1", "R2"
                ]
            },
            "subnet_0005" : {
                "subnet_num" : "0005",
                "max_bw" : "500",
                "p_nodes" : [
                    "R2", "R3"
                ]
            },
            "subnet_0006" : {
                "subnet_num" : "0006",
                "max_bw" : "500",
                "p_nodes" : [
                    "R3", "Ingress/Egress 3"
                ]
            }
        ]
    },
     "slice_01" : {
        "slice_id" : "01"
        "dscp" : "10",
        "subnets" : [
            "subnet_0011" : {
                "subnet_num" : "0011",
                "max_bw" : "500",
                "p_nodes" : [
                    "Ingress/Egress 1", "R1"
                ]
            },
            "subnet_0012" : {
                "subnet_num" : "0012",
                "max_bw" : "500",
                "p_nodes" : [
                    "R1", "Ingress/Egress 2"
                ]
            },
            "subnet_0013" : {
                "subnet_num" : "0013",
                "max_bw" : "500",
                "p_nodes" : [
                    "R1", "Ingress/Egress 3"
                ]
            },
            "subnet_0014" : {
                "subnet_num" : "0014",
                "max_bw" : "500",
                "p_nodes" : [
                    "R1", "R2"
                ]
            },
            "subnet_0015" : {
                "subnet_num" : "0015",
                "max_bw" : "500",
                "p_nodes" : [
                    "R2", "Ingress/Egress 3"
                ]
            }
        ]
     },
```

図 13 スライスを表現する JSON ファイル (2/3).

```
        "slice_02" : {
            "slice_id" : "02"
            "dscp" : 20,
            "subnets" : [
                "subnet_0021" : {
                    "subnet_num" : "0021",
                    "max_bw" : "500",
                    "p_nodes" : [
                        "Ingress/Egress 1", "R1"
                    ]
                },
                "subnet_0022" : {
                    "subnet_num" : "0022",
                    "max_bw" : "500",
                    "p_nodes" : [
                        "R1", "Ingress/Egress 2"
                    ]
                },
                "subnet_0023" : {
                    "subnet_num" : "0023",
                    "max_bw" : "500",
                    "p_nodes" : [
                        "R1", "R3"
                    ]
                },
                "subnet_0024" : {
                    "subnet_num" : "0024",
                    "max_bw" : "500",
                    "p_nodes" : [
                        "R3", "Ingress/Egress 3"
                    ]
                }
            ]
        }
    },
    "qos" : {
        "0" : {
            "rate":"0", "ceil":"MAX_BW", "pri":"0"
        }
        "10" : {
            "rate":"250", "ceil":"500", "pri":"3"
        },
        "20" : {
            "rate":"150", "ceil":"500", "pri":"2"
        }
    }
}
```

図 14 スライスを表現する JSON ファイル (3/3).

クラスは図 16 と同じ階層構造を持つように実装されている. また, 各クラス内での qdisc は Stochastic Fair Queuing (SFQ) という規則が指定されている. SFQ はセッションごとにパケット送信機会を公平に与える qdisc であり, これのおかげで同じクラス内での公平性が保証される. パケットはこれらの子クラスのうちのどれかに振り分けられるが, どう振り分けるかを決定するのが tc filter である.

今回の子クラスは, 5G システムにおいて必要とされている通信形態である高速大容量通信 (eMBB), 超多数端末接続 (mMTC), 超高信頼・超低遅延通信 (URLLC) を模した 3 つのネットワークスライスに, ベストエフォートを加え

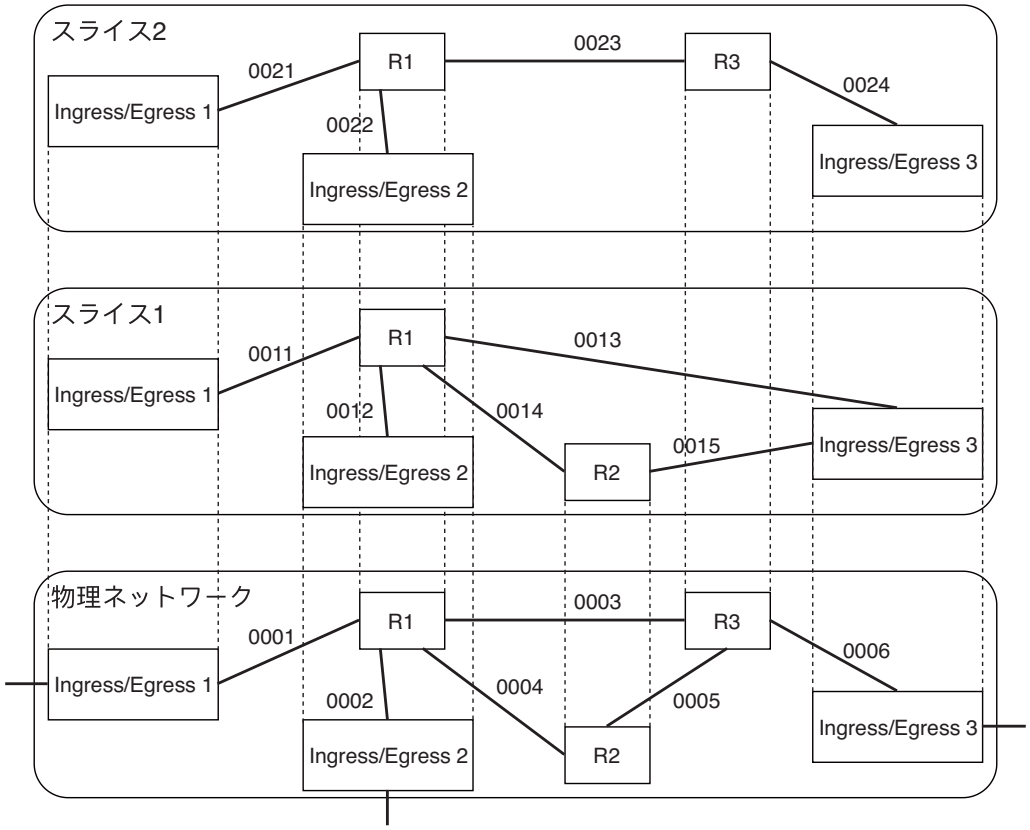

図 **15**　ネットワークスライスの構築例.

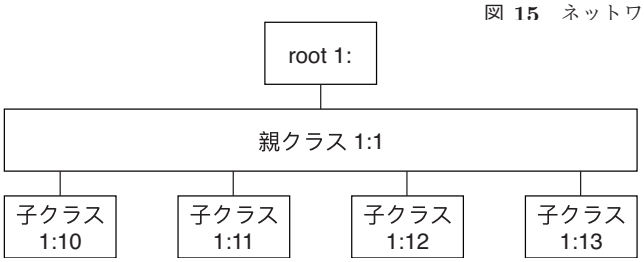

図 **16**　HTB の持つクラス階層構造.

た 4 つを用意し, それぞれの指定 DSCP 値, 帯域保証幅, 帯域上限幅, 優先度を表 1 に示す.

ここでの保証帯域はトラフィックの優先度が帯域の競合相手間で低く, 優先されないトラフィックであるとしてもその値までは保証されている帯域である. 上限帯域はトラフィックの優先度が帯域の競合相手間で高く, 優先されるトラフィックであるとしてもその値までしか帯域を占有できない帯域である. また, DSCP 値の割り振りには現在 Cisco ルータで採用されている DSCP の優先度基準を一部用いている [7]. DSCP 値が 0 であるパケットはベストエフォートとして扱われる.

帯域, 優先度の割当をこの数字とした理由を以下に示す.

- 超高信頼・超低遅延通信
  触感通信, 車車間通信による事故回避など, 最も優先されるべきトラフィックであるため, 優先度は最高の 4 である. しかし, それほど帯域を使用するとは予想されないトラフィックであり, 上限帯域は最大の 500Mbps ではなく 300Mbps となっている.
- 高速大容量通信
  動画再生など次点で優先されるトラフィックであるため, 優先度は 3 である. この通信は大容量であることが予想されるため, 保証帯域 250Mbps, 上限帯域 500Mbps と広めに確保されている.
- 超多数端末接続
  IoT デバイスによるネットワーク構成通信など, それほど優先されないトラフィックであるため, 優先度は 2 である. この通信は高速大容量通信ほどではないが大容量であることが予想されるため, 上限帯域が 500Mbps と高く設定されている.
- ベストエフォート
  ベストエフォートであるため, 優先度は 1 で保証帯域は 0 である. 帯域が空いている場合は使って良いが, 他に優先すべき通信がある際は全ての帯域を譲る設定になっている.

tc filter により, 特定の DSCP 値を持つパケットを対応するクラスに振り分けている.

5.2　ネットワークスライスの実装

ネットワークスライスに参加しているルータ間では L3 ルーティングを, 参加していないルータ間では L2 フォワーディングを行う. このことを本稿では物理デバイスを L2

表 1　5G で想定されるスライスのパラメータ.

スライスの種類	DSCP 値	帯域保証幅 [Mbps]	帯域上限幅 [Mbps]	優先度	クラス ID
超高信頼・超低遅延通信	46	100	300	4	1:10
高速大容量通信	10	250	500	3	1:11
超多数端末接続	20	150	500	2	1:12
ベストエフォート	10, 20, 46 以外	0	500	1	1:13

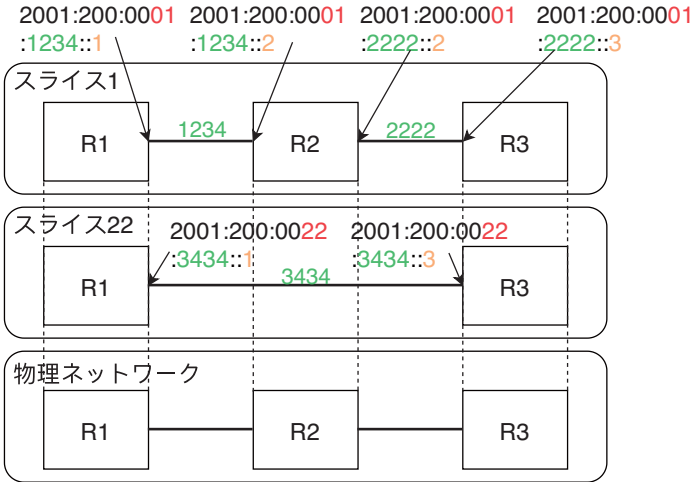

図 17　ネットワークスライスの構築例.

宛先	GW	I/F
2001:200:0001:1234::/64	-	br-host
2001:200:0002:2222::/64	2001:200:0001:1234::2	br-host
2001:200:0022:3434::/64	-	br-host

図 18　R1 の経路表.

宛先	GW	I/F
2001:200:0001:1234::/64	-	br-host
2001:200:0001:2222::/64	-	br-host

図 19　R2 の経路表.

スイッチとして扱うことで実装した. L3 スイッチ用として各物理ノード内にブリッジを作成しておく. このブリッジにスライス ID を埋め込んだ IPv6 アドレスを割り振ることにする. IPv6 アドレスの割り振る場所は, 図 17 のように各ネットワークスライス内のサブネットの端点に 1 つずつとなっている. スライス 22 におけるルータ 2 のデバイスには, L2 フォワーディングで通過するため IPv6 アドレスは必要ない.

また, このときの経路表は図 18-20 のようにする. こうすることで, ネットワークスライス 01 においてはルータ 2 で L3 ルーティングをするが, ネットワークスライス 22 においてはルータ 2 で L2 フォワーディングをし, ネットワークスライス 22 上ではルータ 1 の L3 における隣接ルータはルータ 3 のように見える.

宛先	GW	I/F
2001:200:0001:1234::/64	2001:200:0001:2222::2	br-host
2001:200:0001:2222::/64	-	br-host
2001:200:0022:3434::/64	-	br-host

図 20　R3 の経路表.

```
"slice_native_routing" :[
  {
  "slice_name_X" : [
    "node_name_A", "node_name_B","node_name_C",
    "node_name_D", "node_name_E",],
  "slice_name_Y" : [
    "node_name_A", "node_name_B","node_name_C",
    "node_name_D", "node_name_F",],
  ...
  }
]
```

図 21　ネットワークスライス構築ファイルの実装時追加.

5.3　ネットワークスライス構築ファイルの実装

基本は設計で述べた通りだが, 各サーバおよびルータに tc を仕込む際に各スライス内でパケットの通過する経路上の全マシンの情報が必要となる. それらの情報はネットワーク情報を読み解いていくと判明するが, 今回の実装では図 21 のように定義ファイルの情報から各スライスの経路上のルータの一覧を情報として追加して実装した.

6.　評価

本稿では, スライスの構築にかかる時間を測定し, 保証帯域の遵守を確認する.

6.1　評価環境

表 2 に示す仕様の PC3 台 (R1, R2, R3) を直線状に最大 500Mbps で接続した物理ネットワーク上に, 図 17 に示すようにスライス 01 とスライス 02 を作成した. スライス 01 には R1, R2, R3 が参加しており, スライス 22 には R1 と R3 が参加している. R1 と R2 間のスライス 01 のサブネット番号を 1234, R2 と R3 間のスライス 01 のサブネット番号を 2222, R1 と R3 間のスライス 22 のサブネット番号を 3434 と指定した. 評価に用いるスライスの種類は 5G を想定した 4 種類を用い, そのパラメータを表 1 にまとめた.

表 2　評価用物理マシンの性能.

項目	内容
OS	Ubuntu 16.04.5 LTS
CPU	Intel(R) Core(TM) i5-4570 CPU @ 3.20GHzv × 3
RAM	2GB

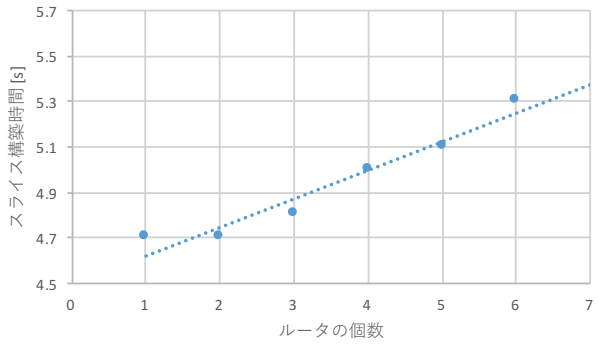

図 22　スライス構築にかかる時間.

6.2　ネットワークスライス構築にかかるオーバーヘッド

ネットワークスライス構築にかかるオーバーヘッドとして, 1 個から 6 個のルータにコントローラからスライス構築命令を送った際にかかる時間を計測した. 計測はそれぞれ 10 回ずつ行い, 図 22 には計測結果の平均値を示している. スライス構築命令を送るルータの数とネットワークスライス構築にかかる時間の間には正の相関関係があることがわかる. 本来は各ルータの設定は並列に実行できるので, スライスの設定時間はルータの台数には大きく関わらないはずであるが, 今回は実装の簡略化のため, 逐次実行するようにしたため, 結果に影響していると考えられる.

ネットワークスライスは設定変更が必要となった際に再構築すればよいので, 頻繁に再構築しない以上数秒であれば問題ないと言える.

6.3　2 スライス間における同一保証帯域の利用

スライス 1, スライス 22 に DSCP 値が同じであり, 結果同じ保証帯域を使うことになるトラフィックを流した. 結果を図 23 に示す.

結果より, スライスが異なっていても同一優先度の中では均等に帯域資源が振り分けられ, またその帯域の合計が上限帯域を超えていないことが実証されている. 同一優先度, つまり同じ HTB 内の子クラスに割り当てられたパケットにおける帯域の分配は sfq の担当であり, 異なるスライス間でも sfq が正常に機能していることの証明と言える.

6.4　2 スライス間における異なる保証帯域の利用

スライス 1, スライス 22 に互いの DSCP 値が異なり, 結果異なる保証帯域を使うことになるトラフィックを流した. 4 つの保証帯域が用意されている関係上, 2 × 4 × 3 = 24 通りの組み合わせが考えられるが, ここではスライス 1 : 超多数端末接続, スライス 22 : 超高信頼・超低遅延通信における評価を取り, その結果を図 24 に示す.

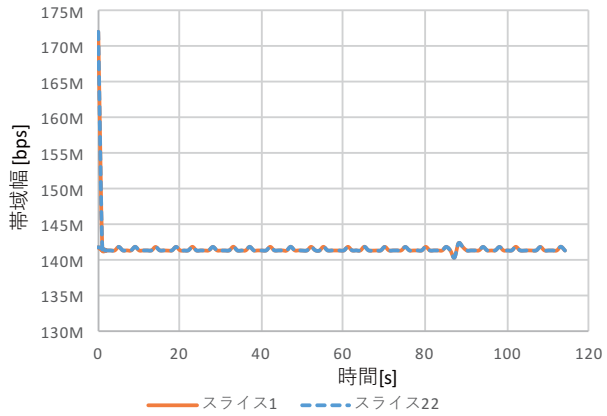

図 23　同一の DSCP 値を持つ 2 つのトラフィックの使用帯域.

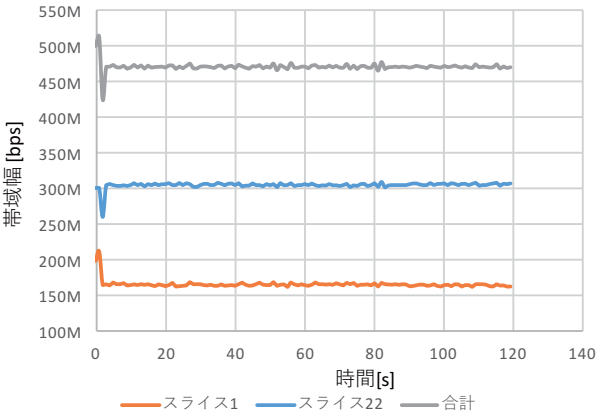

図 24　同一の DSCP 値を持つ 2 つのトラフィックの使用帯域.

スライス 1, スライス 22 に設定されている保証帯域はそれぞれ, 100Mbps, 150Mbps であり, これは合わせても 500Mbps には届かないため, まずその値だけは帯域が保証される. スライス 1 よりもスライス 22 の方が優先度が高いため, そちらに帯域資源を上限帯域になるまで割り振ろうとする. そうした結果 300Mbps, 150Mbps 程度で合わせて 450Mbps であり, 超多数端末接続の方に資源をあまり渡さないまま全体の帯域幅上限幅であるほぼ 500Mbps になったのだと考えられる.

7.　おわりに

現在の LTE-advanced とは異なり 5G システムではネットワークスライスを用いることで多種多様なトラフィックに対応する考えを採用している.

本稿ではネットワークスライスを IPv6 アドレッシング, 経路制御で構築しつつ, スライス間の資源隔離を Diffserv で実装した.

評価の結果, Diffserv により問題なく資源隔離がなされることを示した. 異なるスライス間で同一優先度トラフィッ

クが流れる際でもトラフィックは帯域幅を等分し, その優先度に約束された上限帯域を継続的に超えない. 異なるスライス間で異なる優先度トラフィックが流れる際, 全体帯域と保証帯域の総和が近く無い限り, それらのトラフィックが保証帯域, 上限帯域を割ることはない.

参考文献

[1] K. Matsueda, T. Ochiai, H. Takano, R. Kimura, R. Sawai, and F. Teraoka. Constructing network slices with locator/id split and locator division for 5g core network. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, May 2018.

[2] N. McKeown, T. Anderson, H. Balakrishnan, F. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, March 2008.

[3] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M.Bursell, and C. Wright. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348, August 2014.

[4] P. Garg and Y. Wang. NVGRE: Network Virtualization Using Generic Routing Encapsulation. RFC 7637, September 2015.

[5] S. Ogawa, K. Yamazaki, R. Kawashima, and H. Matsuo. T3: TCP-based High-Performance and Congestion-aware Tunneling Protocol for Cloud Networking. In *Proceedings of 2016 IEEE Conference on Standards for Communications and Networking*, pages 1–7, October 2016.

[6] K. Chan. DiffServ QoS Policy Information Base. RFC 3317, March 2003.

[7] Cisco Nexus 1000V Quality of Service コンフィギュレーション ガイド リリース 4.2(1)SV1(5.1). `https://www.cisco.com/c/ja_jp/td/docs/sw/dcswt/nex1000vswt/cg/018/n1000v-qos/n1000v-qos-6dscpval.html`.