WFQ制御を実現するSDN Southbound Interfaceの提案と評価

東京大学大学院　学際情報学府　中尾研究室
○徳永竣亮　安藤翔伍　中尾彰宏

2014 11/14

概要

1. 背景
2. 研究目的と提案手法
3. 実装・評価
4. まとめ
5. 課題

アプリケーション毎のQoS制御の重要性

インターネットはOne-Size-Fits-Allの収容モデルであり,
全てのアプリケーションは画一的に扱われている

しかし、異なるQoS要件を持つアプリケーションが氾濫しており、
将来ネットワークにおいて、ソフトウェア集中制御によるEnd-to-Endでの
アプリケーション毎に多様なQoS制御が必要となることが予想される

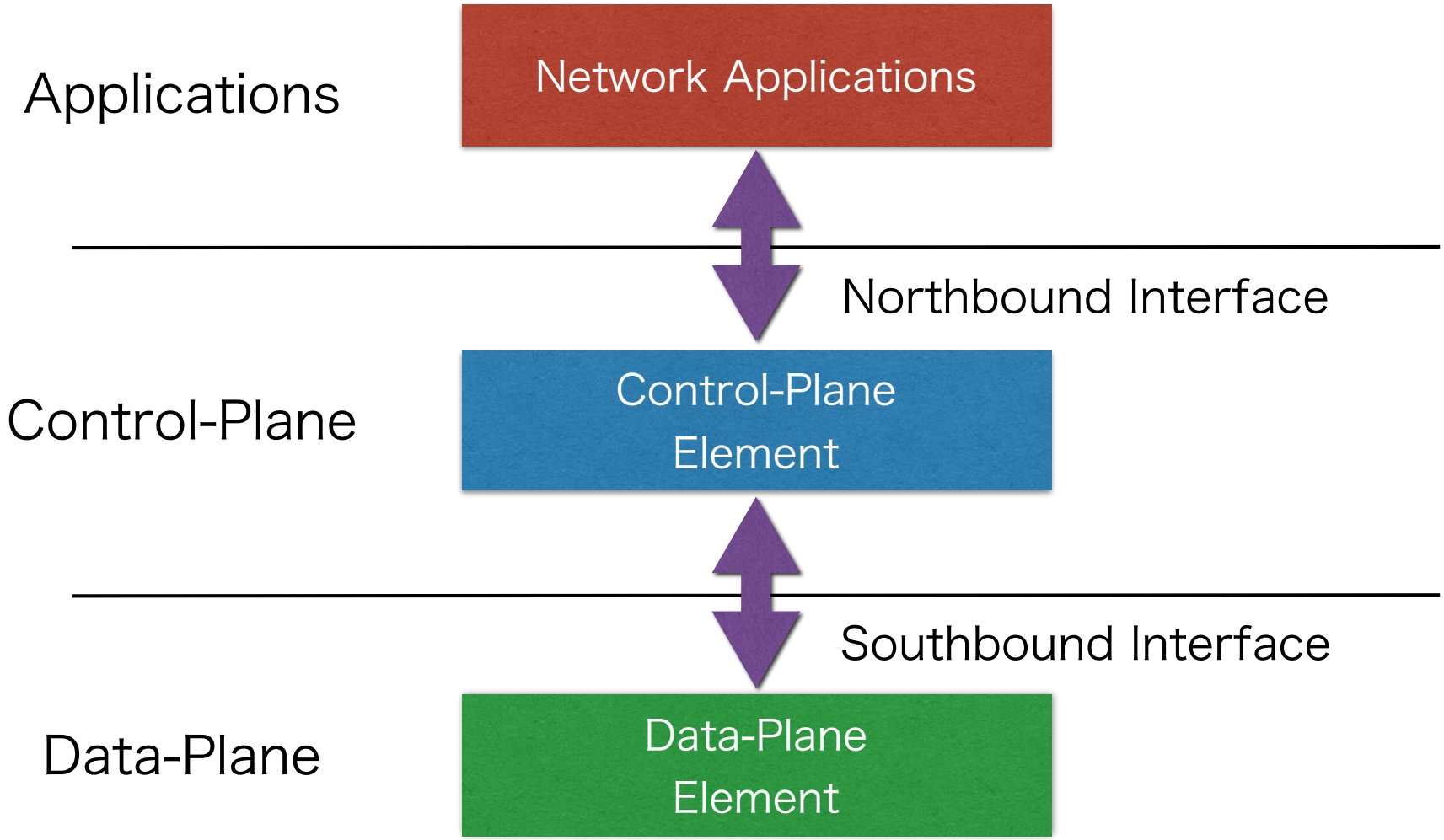
例

・VoIP, カンファレンスコールなどのリアルタイムアプリケーション

・4K, 8Kなどの大容量コンテンツ転送アプリケーション

・M2Mなど常時帯域を占有する多フローのセンサーデータ収集アプリケーション

SDN Architecture



既存のSDNデータプレーン要素における問題

- ハードウェアで構成されるデータプレーン要素(Data Plane Element)、
  および、Southbound Interface (SBI)の拡張が容易ではない

- 特にQoS優先制御を行うデータプレーン要素、
  およびSBIは十分な機能が備わっていない

- アプリケーションを同定するデータプレーン要素、
  およびSBIは存在しない

＊データプレーン要素：コントローラによって制御されるデータプレーン機器
＊Southbound Interface：コントローラがデータプレーンを制御する際に使用するInterface

OpenFlowにおけるQoS制御

SDNの代表的な標準規格であるOpenFlow[1]は，フローベースのQoS制御を持つ

-OpenFlowのマッチングルールはフローベースであり，
フロー毎に最大帯域・最小帯域を設定して帯域制御を行う

OpenFlowにおけるQoS制御

SDNの代表的な標準規格であるOpenFlow[1]は，フローベースのQoS制御を持つ

-OpenFlowのマッチングルールはフローベースであり，
フロー毎に最大帯域・最小帯域を設定して帯域制御を行う

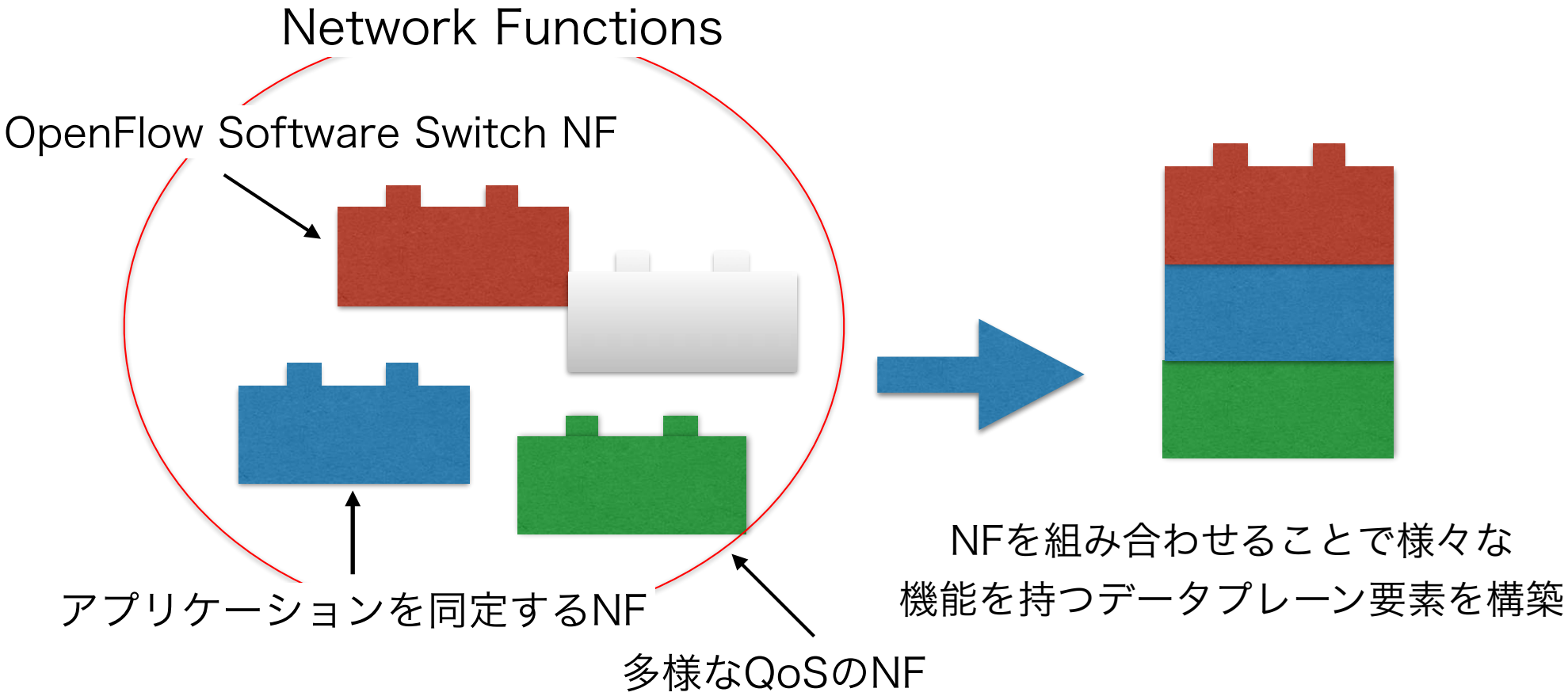
OpenFlow の仕様にない、多様なQoS制御を拡張するのは難しい

例：フローではなくアプリケーションプロセスを同定して、
WFQなどを組み合わせる

データプレーンの拡張

既存研究

・データプレーン内で再利用性の高いNetwork Function(NF)を
ソフトウェア実装し，NFを組み合わせてデータプレーン要素を構成[2][3]



データプレーンの拡張

既存研究

・データプレーン内で再利用性の高いNetwork Function(NF)を
ソフトウェア実装し，NFを組み合わせてデータプレーン要素を構成[2][3]

・アプリケーション特化型トラヒック制御[2,3,4]
トラフィックからアプリケーションを判別する制御方式

データプレーンの拡張

既存研究

・データプレーン内で再利用性の高いNetwork Function(NF)を
ソフトウェア実装し，NFを組み合わせてデータプレーン要素を構成[2][3]

・アプリケーション特化型トラヒック制御[2,3,4]
トラフィックからアプリケーションを判別する制御方式

本研究の提案

QoS制御を行うNFを実装し，他NFと組み合わせたデータプレーン要素を
構成することで従来のSDNでは困難な多様なQoS制御が可能

研究目的と提案手法

研究目的

再利用可能なNetwork Function(NF)でのデータプレーン要素の拡張によるアプリケーション毎のQoS制御の実現

提案手法

WFQ[5]による優先制御のNFとSouthbound Interface(SBI)を提案

評価

1. WFQによる優先制御を行う再利用可能なNFのClick Element によるソフトウェア実装と性能評価
2. 上記NFを制御するSBIの実装と動作確認

WFQを制御するSBIの設計

・コントローラからWFQ Network Functionを操作・利用するSouthbound APIを定義

・Southbound APIはメソッドを持ち，コントローラはこれらを呼び出してWFQ NFを管理・操作する

メソッド名	引数	説明
add	優先度 p, 分類情報 i	iを登録して， 優先度 pを割り当てる
delete	分類情報 i	登録されている 分類情報 iを削除
list		登録されている分類情報と その優先度を表示

Southbound APIが持つメソッド

実装・評価

1. WFQの概要
2. Click Modular Router[6]の概要
3. ClickによるWFQ Elementの実装
4. フローベースのWFQ Network Functionの実装
5. SBI Managerの実装
6. 実装の評価実験
    1）Network Simulator 3 (ns-3)を用いたWFQ関数の評価
    2）実機に実装したWFQ Network Functionの評価

WFQの概要

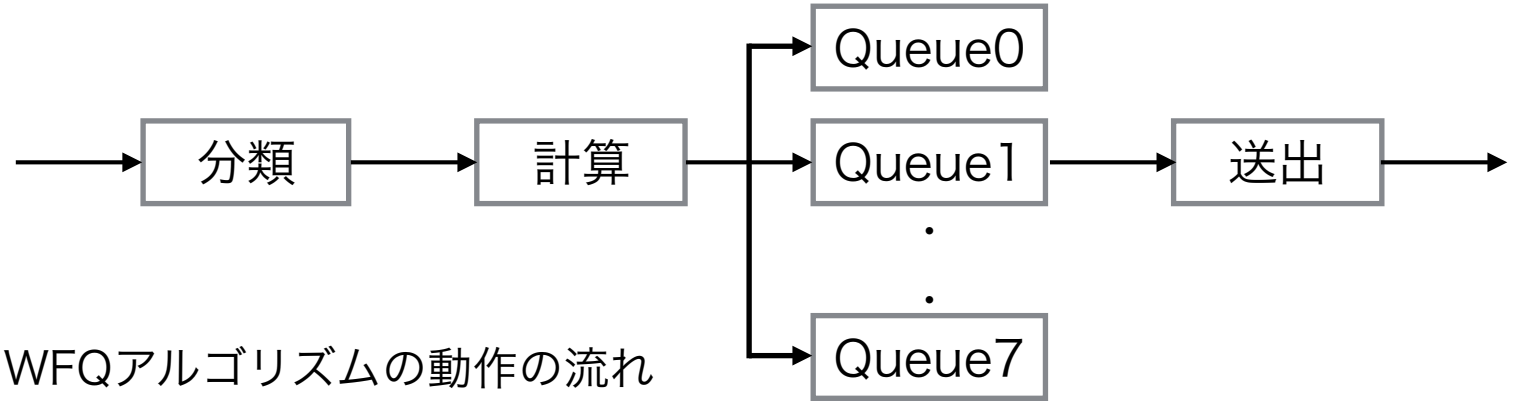
- WFQは，各データフローに対して，優先順位の異なるスケジューリングを行うアルゴリズムである
- 到着パケットを優先度毎に分類

  -ToSに応じた0～７の８段階の優先度をサポート

- シークエンスナンバーを計算し，Queueに格納

  -シークエンスナンバーは，送出スケジューリング時に使用する値で，パケットのサイズと優先度に依存する

- シークエンスナンバーの小さい順に送出



WFQアルゴリズムの動作の流れ

Click Modular Routerの概要

- 柔軟に設定変更が可能なRouterを構築するための
  Software Framework

- Elementと呼ばれる，Routerの各機能を実装した部品を
  組み合わせることで，Software Routerを構成する

  -ex. Classifier：パケットを条件に従って分類する
  　　Queue：パケットを格納する

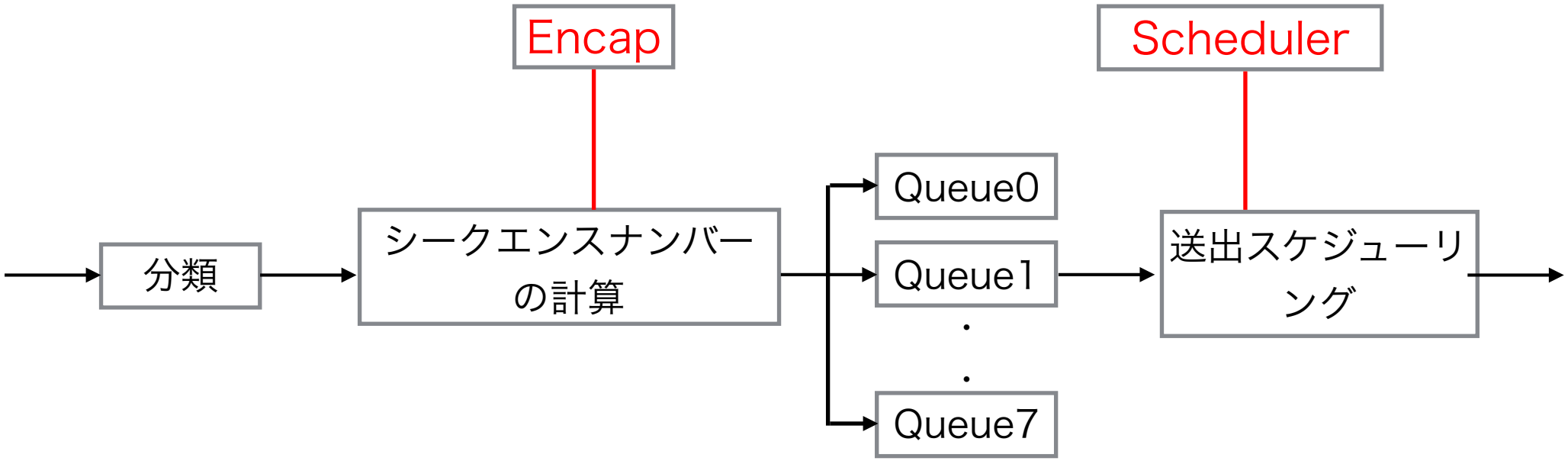
- Elementの中にはハンドラが実装されているものが有り，
  呼び出すことでElementの情報の取得・設定の変更を行う

  -ex. Queue　read capacity: 最大容量を返す
  　　write capacity: 最大容量を変更する

＊ハンドラ：ユーザーが実行中のRouter内Elementとやりとりを行うアクセスポイント

ClickによるWFQ Elementの実装

- WFQ Network Functionの構成要素として,
  WFQのアルゴリズムを実行するためのClick Elementを２つ実装



WFQアルゴリズムの動作の流れ

Encap

- 到着した各パケットに対し，送出スケジューリングに必要なシークエンスナンバーを計算するElement

$$seq_i(p) = \begin{cases} \frac{C}{w_i+1} * l(p) + round & (Queue\ i\ is\ empty) \\ \frac{C}{w_i+1} * l(p) + pre(i) & (else) \end{cases}$$

C : 定数

w_i : Queue iの優先度

$length(p)$: パケット p の長さ

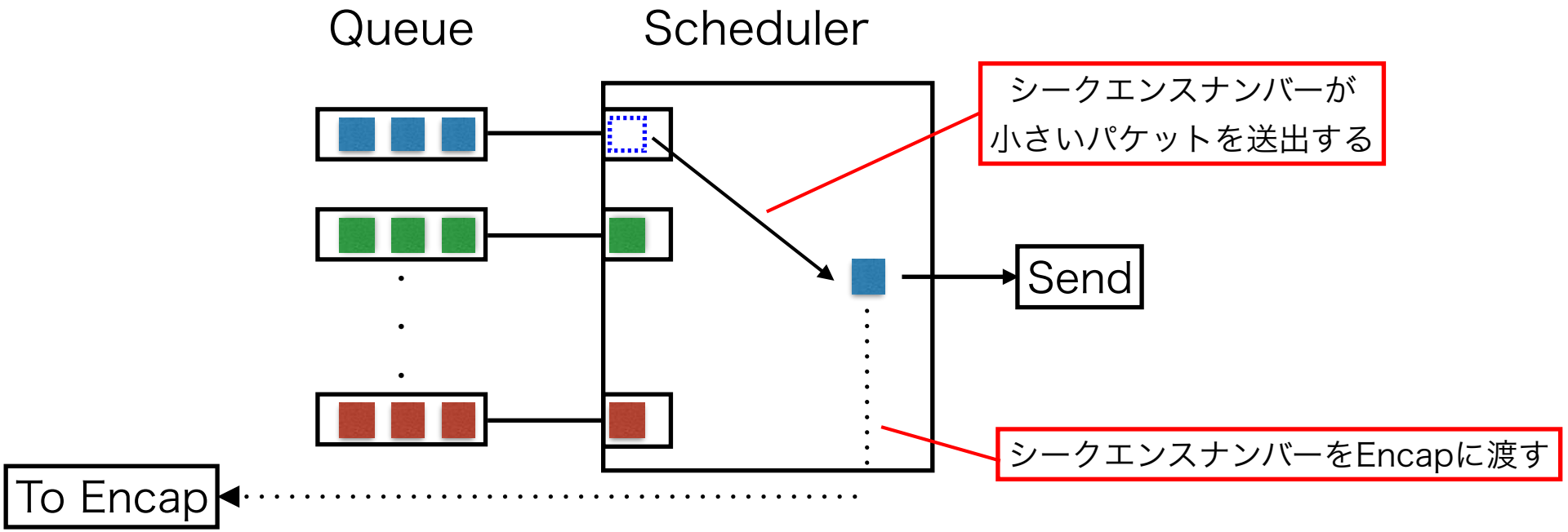
$round$: 各Queue共通の変数
直近に送出された
パケットのシークエンスナンバーをとる

$pre(i)$: Queue iの最後尾パケットの
シークエンスナンバー

Scheduler

- 送出スケジューリングを行うElement

- 各Queueの先頭パケットのシークエンスナンバーを比較し，
  小さいものから送出する



フローベースのWFQ Network Functionの実装

・今回は，WFQ機能の動作確認のため，フローベースのWFQ NFを実装

・WFQの根幹部分として，４つのElementにより実装している

1）Classifier：到着パケットを，登録されたフロー情報に従って分類
→既存Elementにハンドラを追加

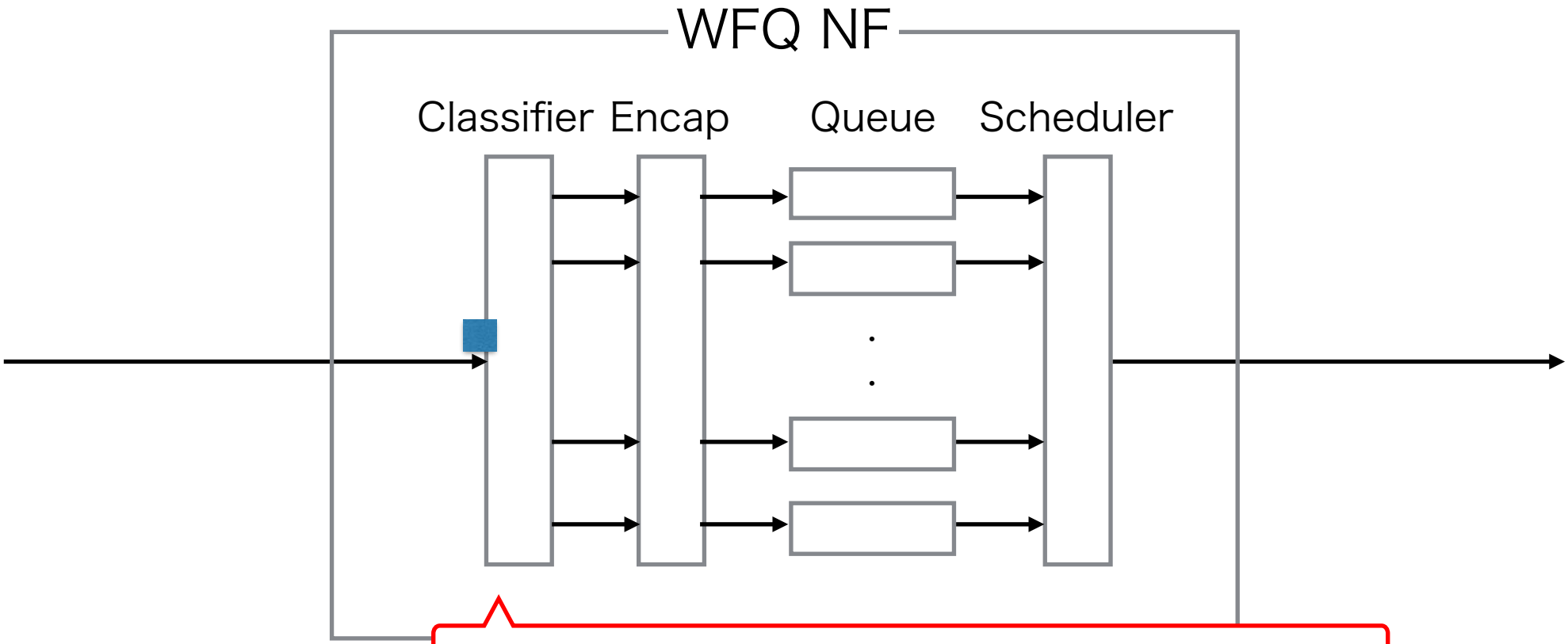
-追加ハンドラ　read list: 登録されているフローのリストを返す
write add: フロー情報を書き込む

2）Encap：シークエンスナンバーを計算し，パケットに装着
→Elementを作成

3）Queue：到着パケットを格納する
→既存Elementを使用

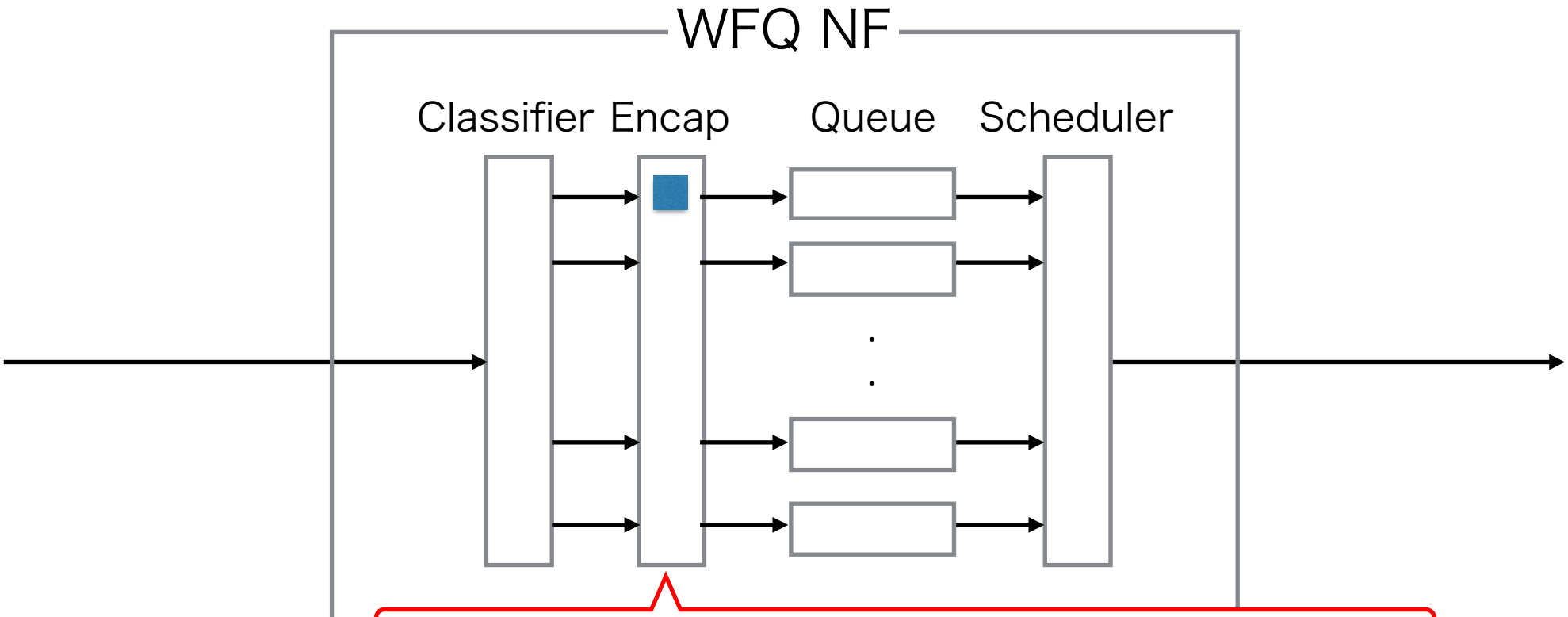
4）Scheduler：シークエンスナンバーの小さい順にパケットを送出する
→Elementを作成

WFQ Network Functionの制御の流れ



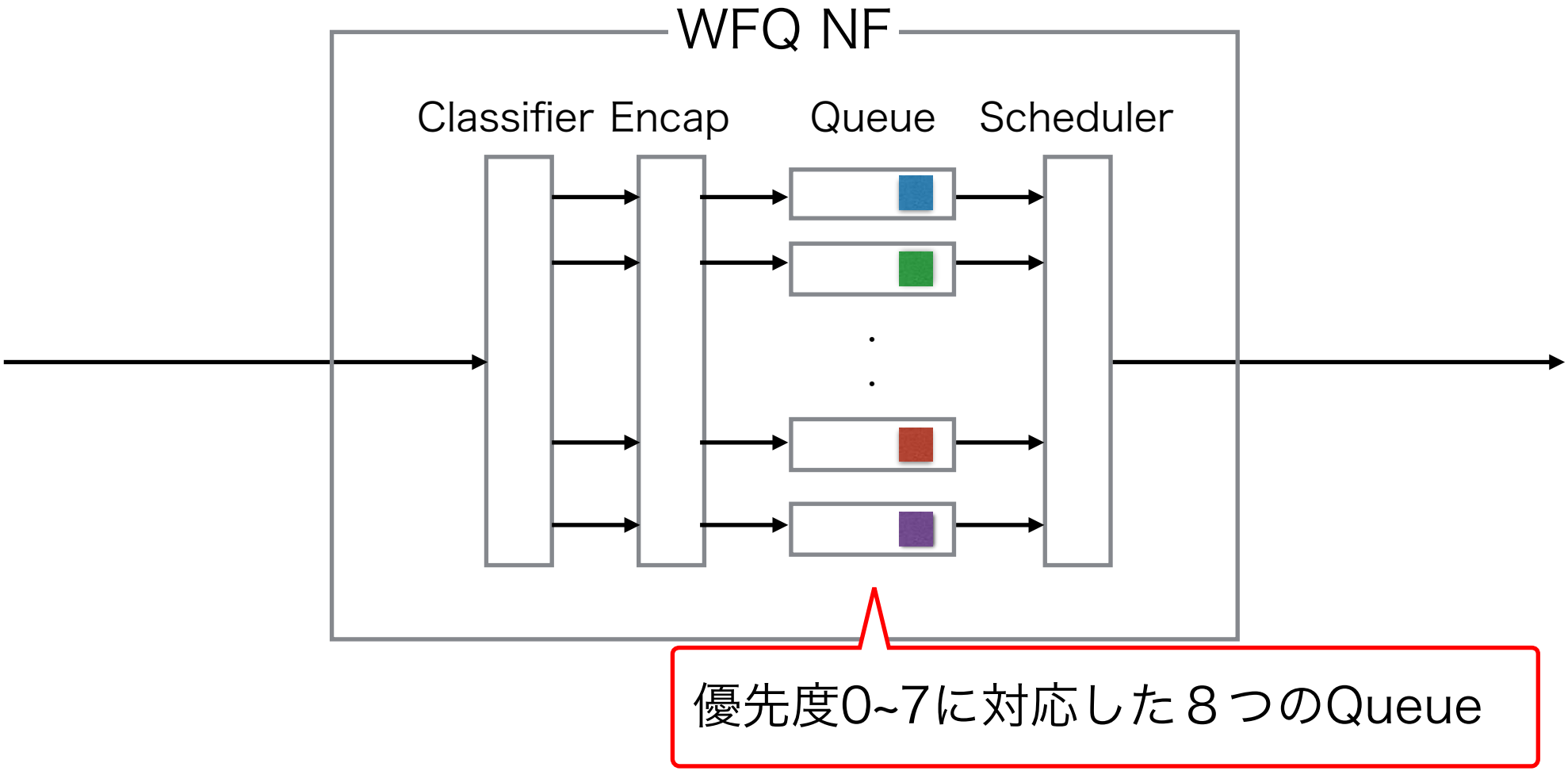
Classifier：到着パケットを，登録されたフロー情報に基づき優先度ごとに分類する

WFQ Network Functionの制御の流れ

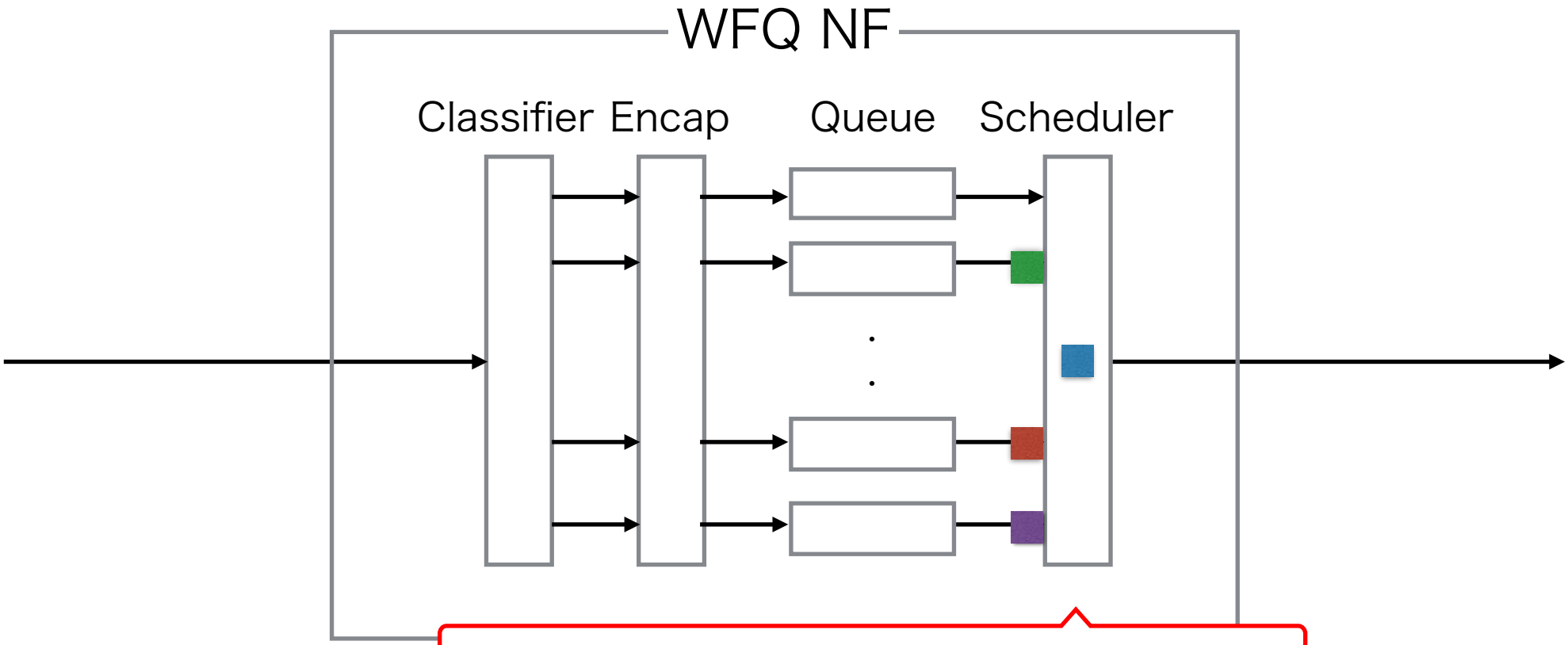


Encap：シークエンスナンバーの計算を行い,
パケットの先頭に装着
WFQはシークエンスナンバーの値で
パケット送出の順番を決定する

WFQ Network Functionの制御の流れ

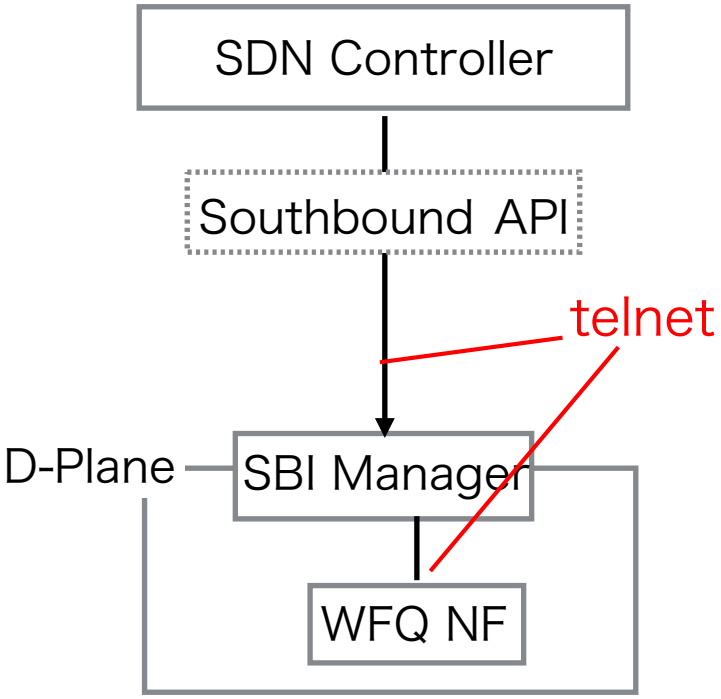


WFQ Network Functionの制御の流れ



Scheduler:シークエンスナンバーの
小さい順にパケットを送出する

SBI Managerの実装



- SBI Managerとは，APIを通してControllerとやり取りをし，WFQ NFの管理・設定を行うソフトウェアである
- Managerに実装されたメソッドは，WFQ NF内のElementのハンドラを呼び出すことでステータスの管理・設定を行う
- ManagerとController，ManagerとWFQ NFはtelnetで通信を行う

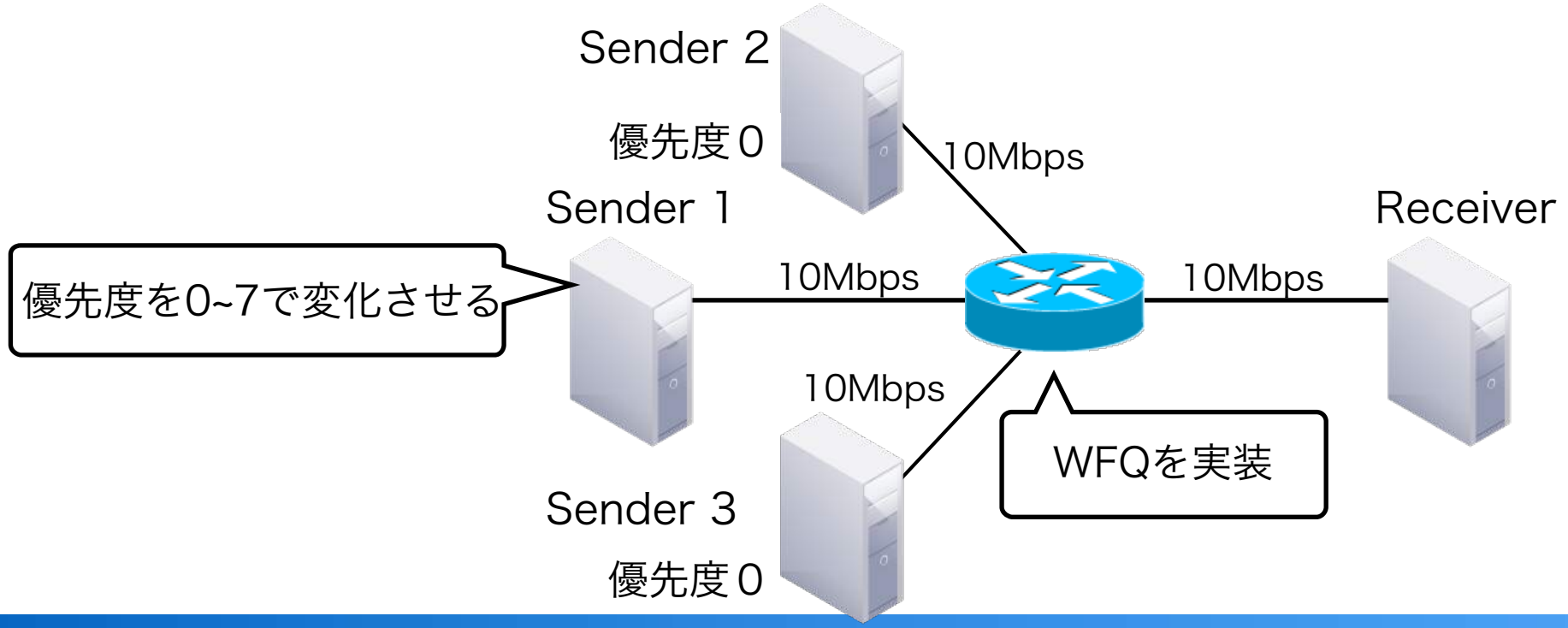
SBI Managerの実装

メソッド名	引数	仕様	実装
add	優先度 p, フロー f	優先度 p用の配列に, フロー fを加える	Pythonで配列を値に持つ辞書を作り，pをキーとして配列を使用する
delete	優先度 p, フロー f	優先度 p用の配列から, フロー fを削除	Pythonで配列を値に持つ辞書を作り，pをキーとして配列を使用する
write	優先度 p	優先度 p用の配列内のフローをClassifierに登録	telnetでClassifierのwrite addハンドラを呼び出す
list		登録されているフローとその優先度を表示する	telnetでClassifierのread listハンドラを呼び出す

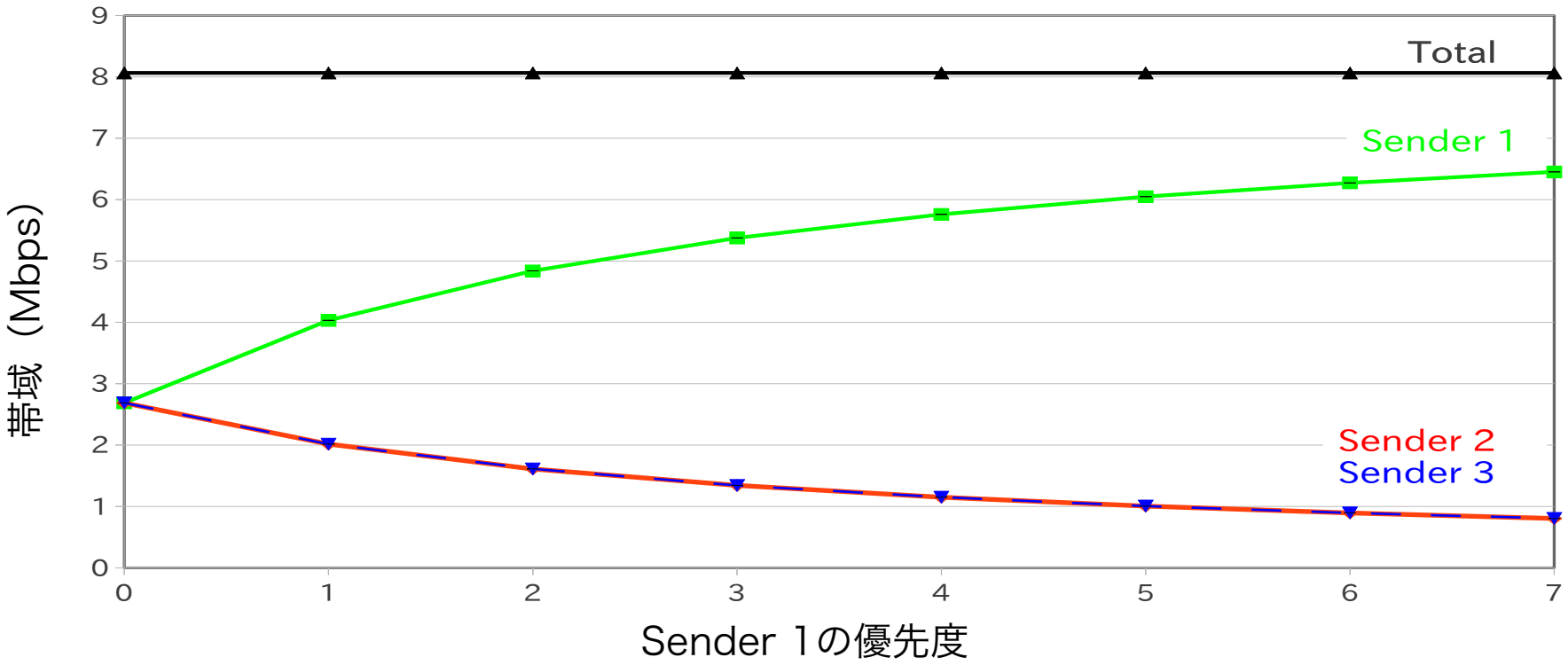
SBI Managerに実装したメソッド

ns-3によるWFQの評価実験

- ns-3の関数としてWFQを実装し，WFQアルゴリズムが正しく動作するか評価を行った
- ３つのSenderから0.1ms間隔でUDPパケットを送信し，各Senderが使用する帯域と，帯域の割合を測定
- 10秒間の測定を5回行い，平均を算出



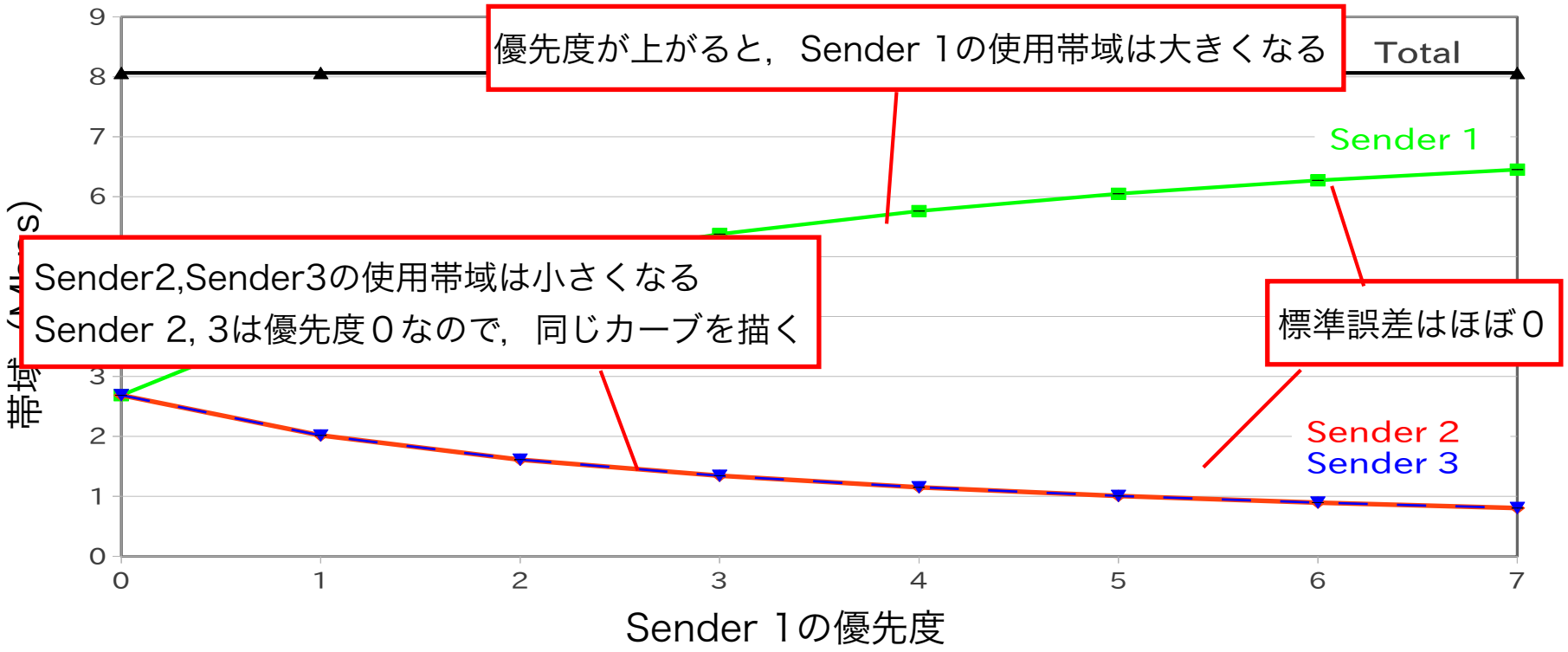
各Senderの使用帯域の変化



各Senderの使用帯域の変化

- Sender 1の優先度が上昇すると，Sender 1の使用帯域は増大する
- 各Senderの使用帯域の比は，優先度に１を足した値の比と一致する
- 例えば，Sender 1の優先度が5の時は，使用帯域の比は6:1:1になっている

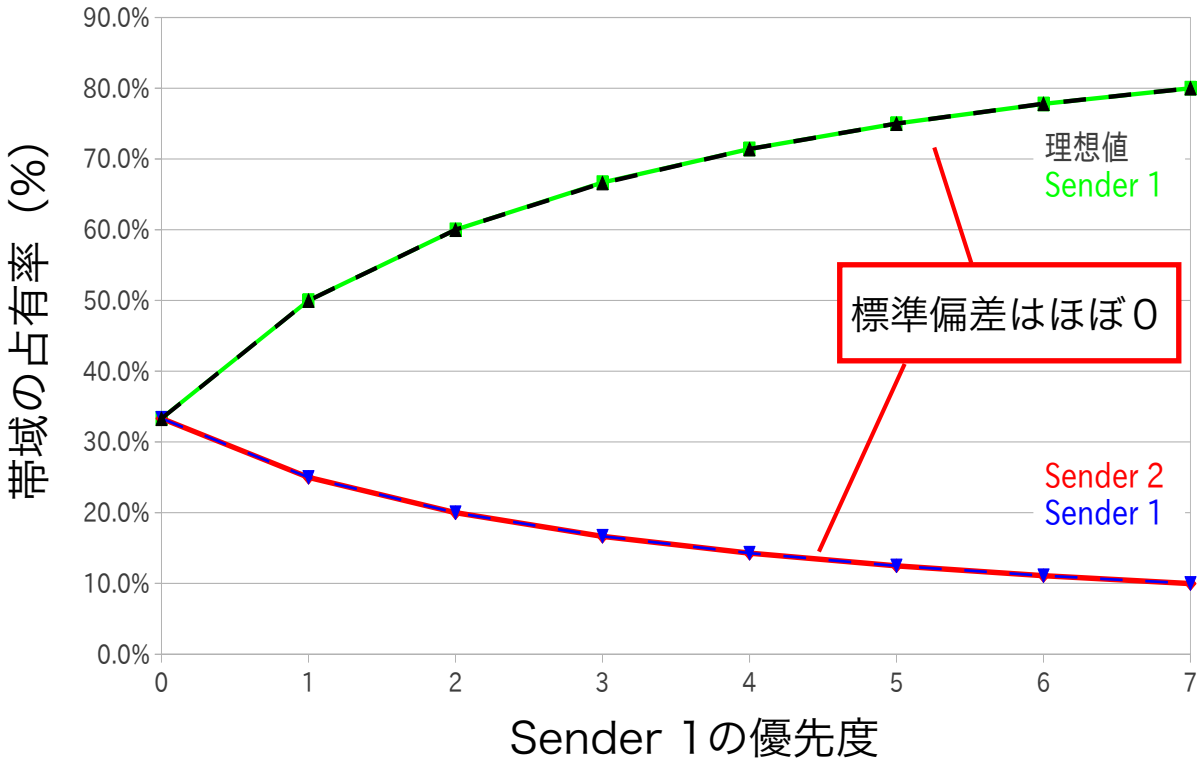
各Senderの使用帯域の変化



各Senderの使用帯域の変化

- Sender 1の優先度が上昇すると，Sender 1の使用帯域は増大する
- 各Senderの使用帯域の比は，優先度に１を足した値の比と一致する
- 例えば，Sender 1の優先度が5の時は，使用帯域の比は6:1:1になっている

各Senderの帯域占有率の変化



各Senderの帯域占有率の変化

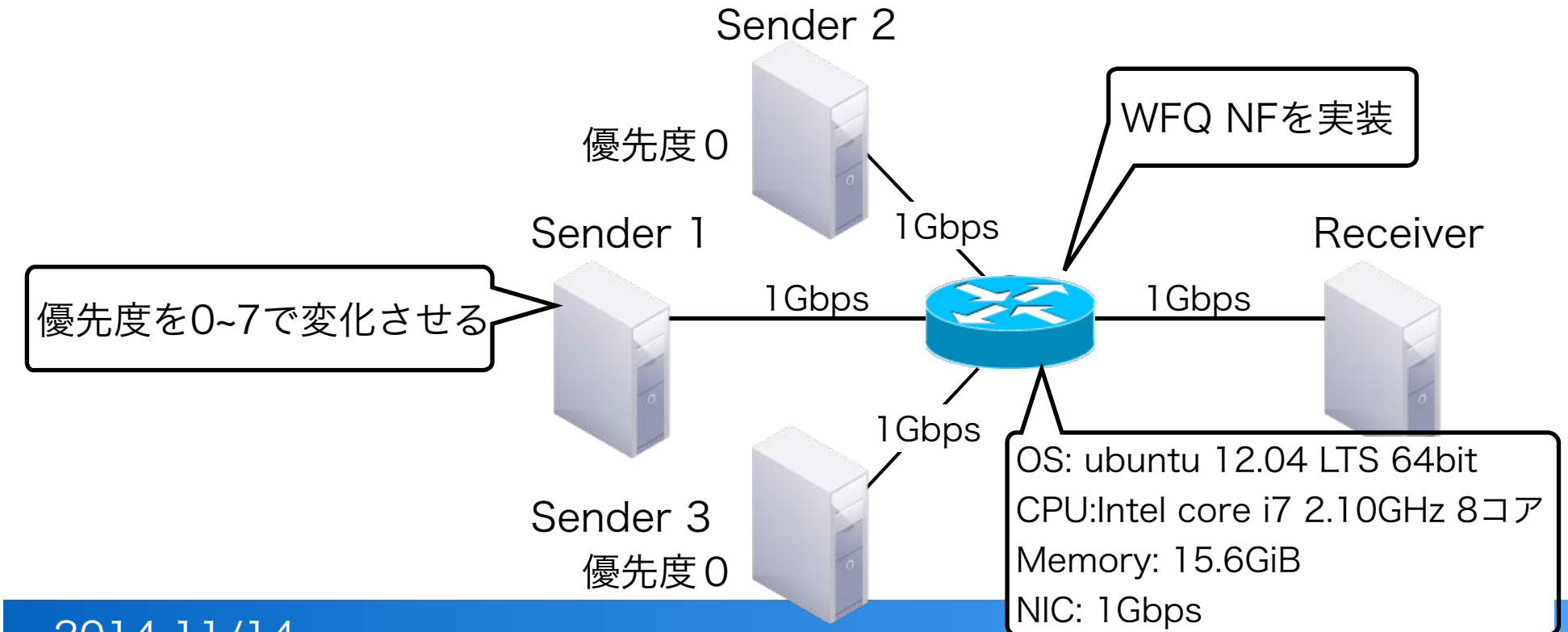
R_i をSender iの理想的な占有率,
w_i をSender iの優先度とすると,

$$R_i = \frac{w_i + 1}{\sum_{i=1}^{3}(w_i + 1)}$$

・Sender 1は優先度が上昇すると帯域占有率も増大するが,
その値は理想値と全く同じカーブを描くことがわかる
・このことから，WFQのアルゴリズムを正確に実装していることがわかる

実機によるWFQ Network Functionの帯域測定

・作成したWFQ NFを実機に実装し，評価実験を行った
・iperfを用いて，３つのSenderから
UDPパケットをReceiverに送信し，帯域を測定する
・100sの測定を三回行い，平均値を算出



iperfの使用帯域設定

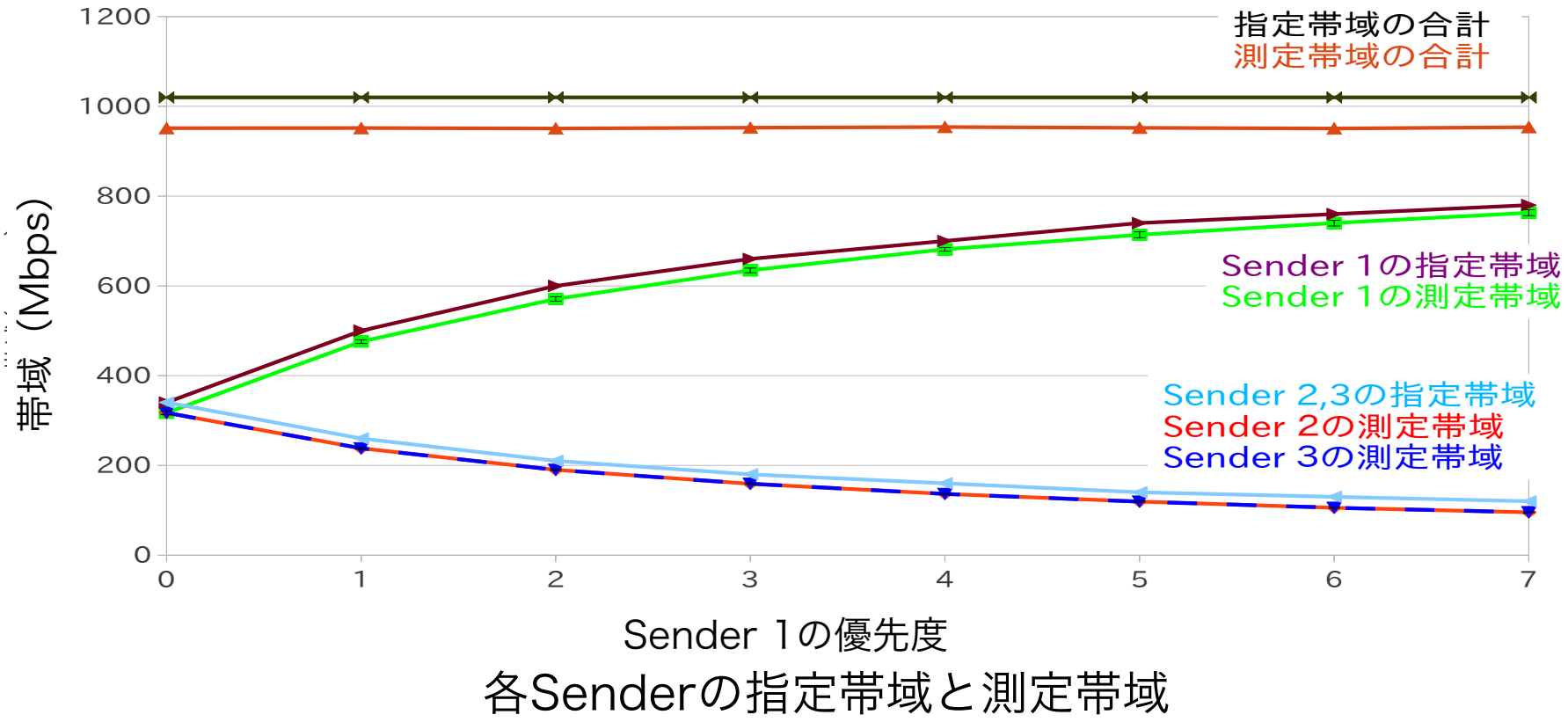
Sender 1の優先度

	0	1	2	3	4	5	6	7
Sender 1	340	500	600	660	700	740	760	780
Sender 2	340	260	210	180	160	140	130	120
Sender 3	340	260	210	180	160	140	130	120

各Senderのiperfで指定する使用帯域の表（Mbps）

- 指定帯域は，1Gbpsに各Senderの帯域占有率の理想値を掛けたものよりも10M~20Mbps程度高い値を設定している
- 指定帯域が大きすぎるとQueueが瞬時に溢れてしまい，常にパケットロスが起きるという，特殊な状態になってしまう
- 小さいとリンクの帯域が余ってしまい，優先度の影響を測定できない

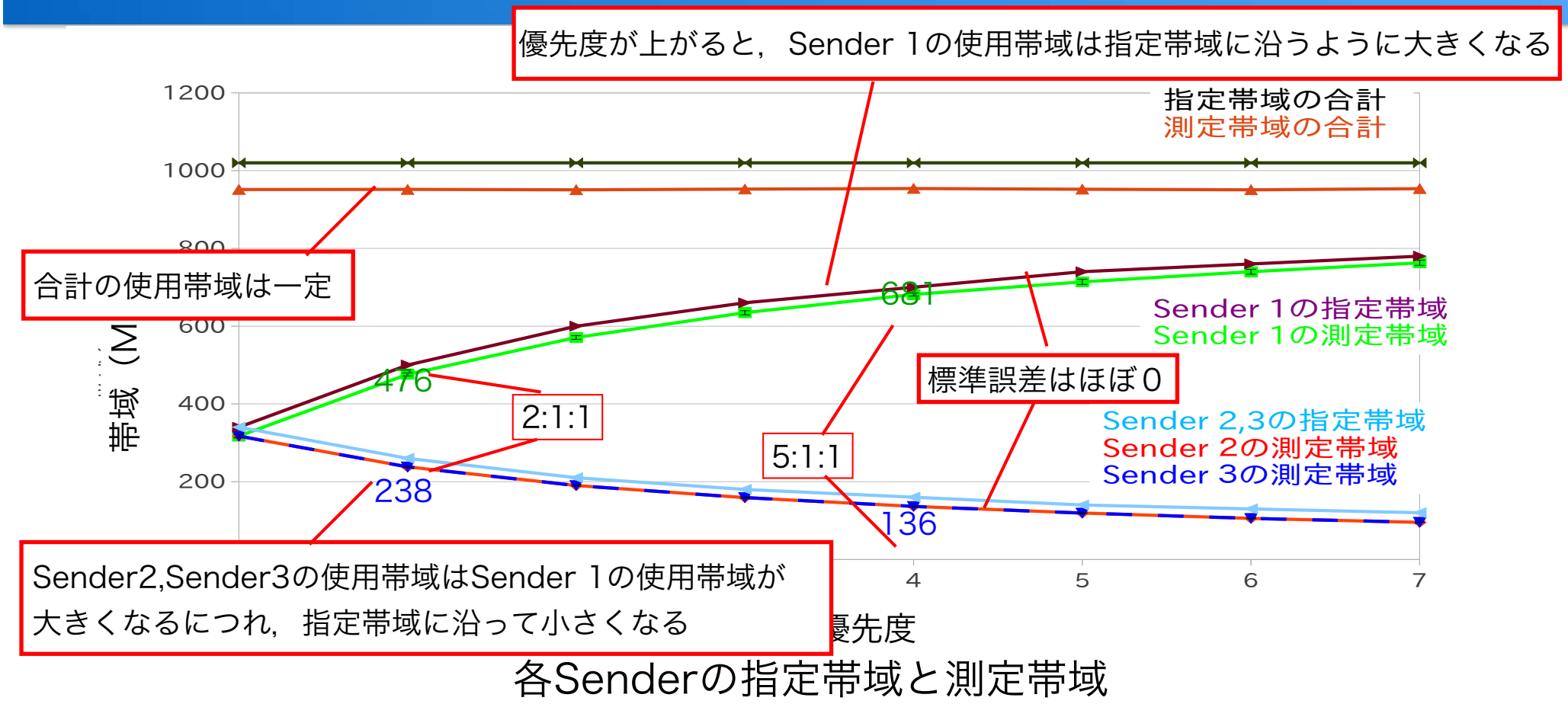
各Senderの使用帯域



各Senderの指定帯域と測定帯域

- 使用帯域の比率は，常に優先度に１を足した値の比率になっている
- 例えば，Sender 1の優先度が4の時は，使用帯域の割合は5:1:1となっている
- このことから，WFQ NFによって適切に優先制御が行われていることがわかる

各Senderの使用帯域

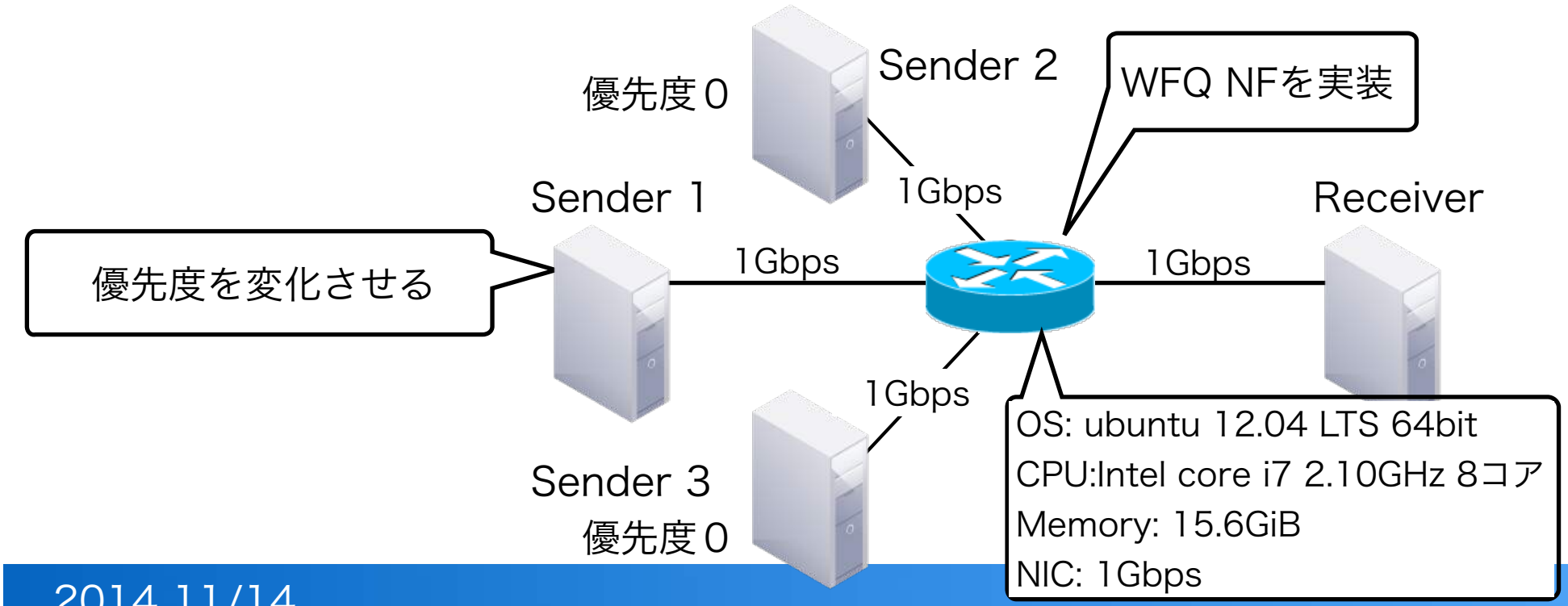


各Senderの指定帯域と測定帯域

- 使用帯域の比率は，常に優先度に１を足した値の比率になっている
- 例えば，Sender 1の優先度が4の時は，使用帯域の割合は5:1:1となっている
- このことから，WFQ NFによって適切に優先制御が行われていることがわかる

実機によるWFQ NFのパケットロス率測定

- iperfを用いて，３つのSenderからUDPパケットを Receiverに送信し，パケットロス率を測定する
- パケットロス率は，Senderが送ったパケット数に対する Receiverが受信出来なかったパケット数の割合とする
- 100sの測定を三回行い，平均値を算出した



iperfの使用帯域設定

	Sender1	Sender2	Sender3	合計
指定帯域	500	340	340	1180

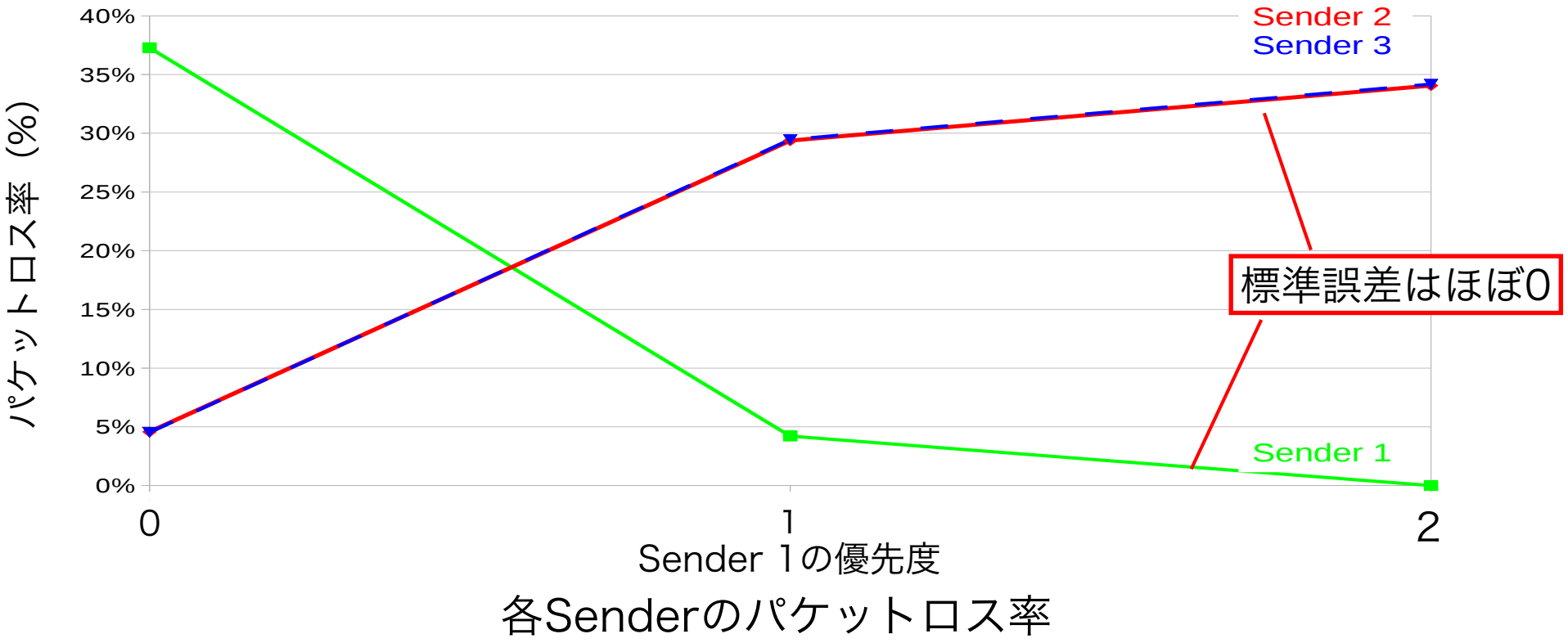
各Senderのiperfで指定する使用帯域の表（Mbps）

- 上記の帯域指定だと，Sender 1の優先度が0の場合，
  Sender 1のみ大量のパケットロスが生じる

- Sender 1の優先度の上昇に伴い，Sender 1のパケットロス率は
  低下すると予想される

- 指定した使用帯域とリンクの帯域（1000Mbps）の比率から考えると，
  優先度1～2あたりでパケットロス率が０になることが予想される

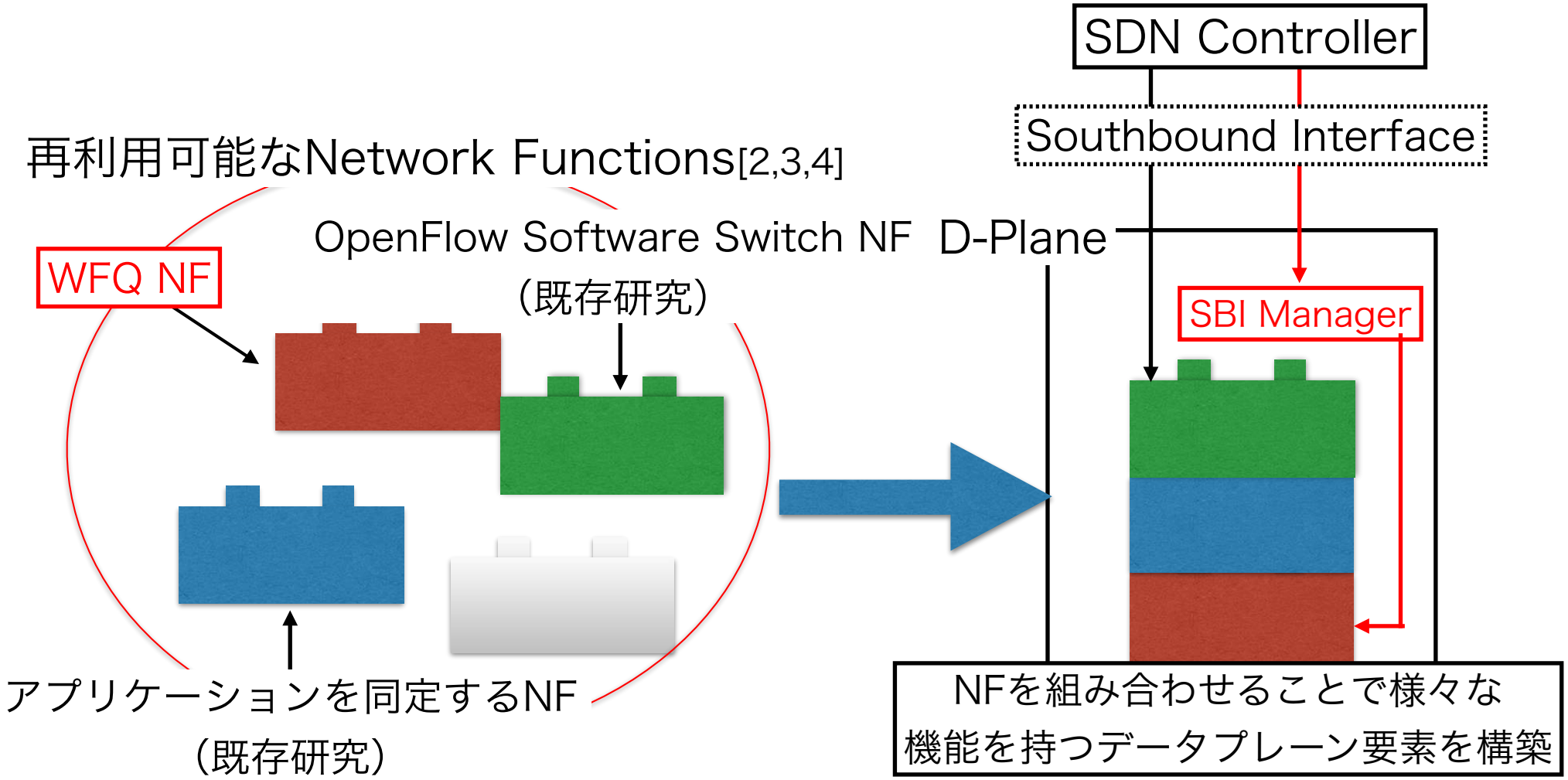
各Senderのパケットロス率



各Senderのパケットロス率

・優先度が上昇すると，Sender 1のパケットロス率は大きく低下し，
優先度2の時に０%になる
・このことから，WFQ NFによってパケットロスが低減されていると言える

Network Functionによるデータプレーン拡張



・今回の実装は，Linux上でWFQ NFをC++，SBI ManagerをPythonで行い，それぞれのソースコードは400行，230行程度である

本研究の貢献

- **WFQ優先制御のSouthbound APIを設計**
- **WFQの機能を再利用可能なNetwork Functionとして設計・実装**し、**既存のロジックと組み合わせてデータプレーン要素がソフトウェアによる拡張可能であること**を示した
- WFQ NFが，**指定したフローに対してパケットロスを低減させたり，適切に優先度を設定可能**であることを示した

将来の課題

・再利用可能なWFQ制御のNFを他のNFと組み合わせて新たな

データプレーン要素を構築・評価

→アプリケーション特化型トラヒック制御のNFとWFQ NFを組み合わせた

データプレーンを構築し，アプリケーション毎のWFQ制御の評価を行う

・ソフトウェアで定義された新たなデータプレーン要素のパフォーマンス向上

→M2Mゲートウェイや MVNOゲートウェイ上での実用に耐える10-100Gbps程度の

ラインレートの性能を目標

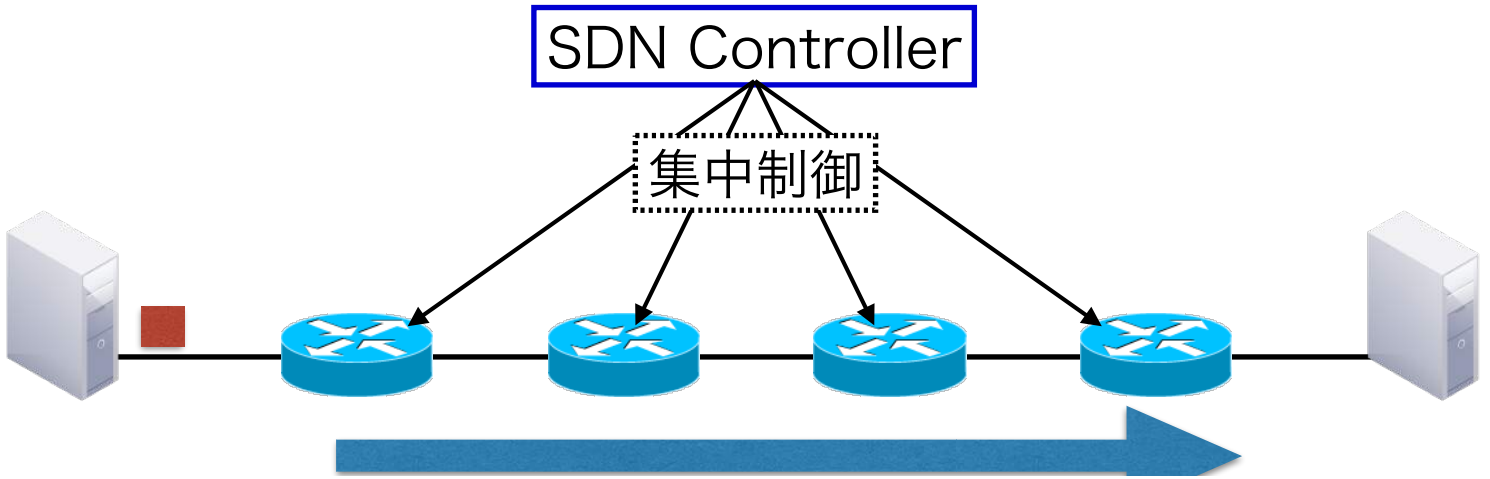
・データプレーン要素の定義がもたらす新しいアプリケーションユースケースの検証

→SDNの集中制御による特定アプリケーションフローのEnd-to-EndのQoS制御の評価

将来課題（今回はアイデアのみ）
QoS制御Network Functionのユースケース

例1：アプリケーション特化型トラヒック制御のNFと組み合わせた
データプレーン要素を構築することで，アプリケーション毎のQoS制御を行う

例2：エンド間に多数のネットワークノードが接続されているネットワークでの
通信において，SDNの集中制御によってネットワークノードのQoS制御を
コントロールし，経由ホップ数が多いパケットの送出を優先することで，
送信先に近いノードでパケットロスを起こし，それまで使用した帯域が
無駄になってしまうことを防ぐ



特定のアプリケーションフローの優先度を経由ホップ数に応じて上げる

参考文献

[1]N.McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. peterson, J. Rexford, S. Shenker, and J. Tuner. “OpenFlow: enabling innovation in campus networks.” SIGCOMM Comput. Commun. Rev., 38(2):69-74, 2008

[2]Akihiro Nakao, “Software-Defined Data Plane Enhancing SDN and NFV”, Special Section on Quality of Diversifying Communication Networks and Services, IEICE Transactions on Communications ,to appear, 2015 Jan.

[3]Akihiro Nakao, Ping Du, “Application and Device Specific Slicing for MVNO”, 2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), to appear, 2014, October

[4] 田中康隆, Du Ping, 山本周, 中尾彰宏（2014）
“SDNにおけるアプリケーション特化型トラヒック制御”.

[5]Demers A, Keshav S, Shenker S (1989)
"Analysis and simulation of a fair queueing algorithm”
ACM SIGCOMM Computer Communication Review 19

[6]Eddie K, Robert M, Benjie C, John J, M. Frans K (2000)
“The Click Modular Router”. ACM Transactions on Computer Systems (TOCS)
Volume 18 Issue 3, Aug. 2000 263-297