10回目

パケットスケジューリング（その2）

目的

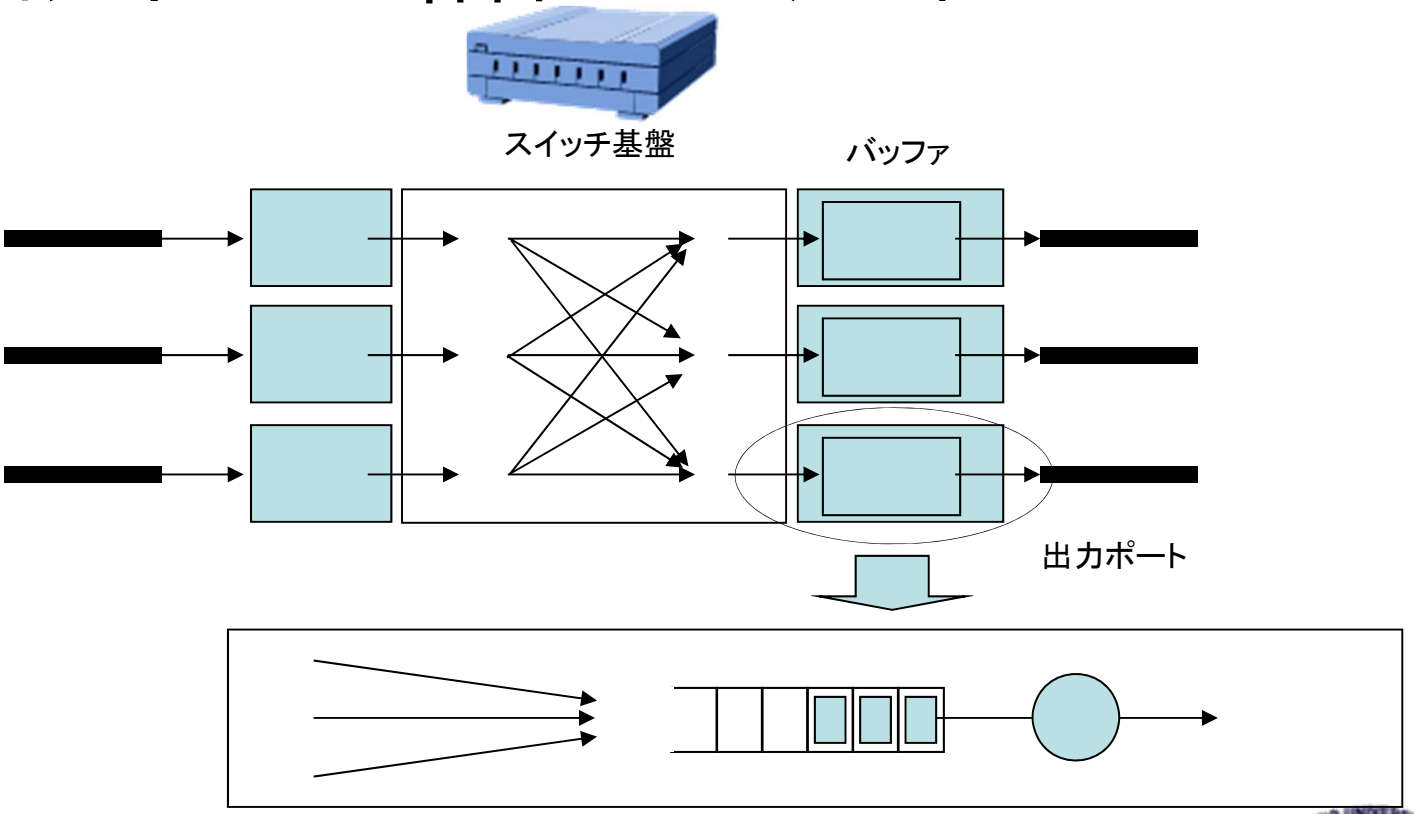
- パケットのスケジューリング手法を学ぶ
- QoS (Quality of Service)手法を学ぶ

QoSトラヒック制御

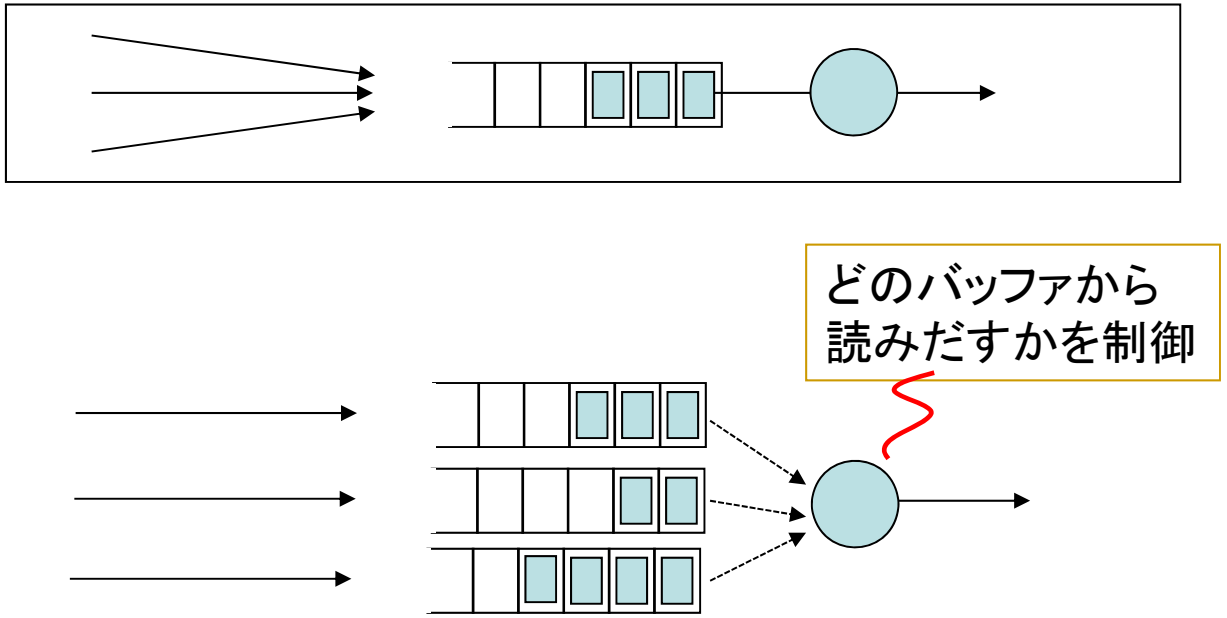
- Quality of Service
  - ネットワーク上で、ある特定のフローのための帯域を予約し、一定の通信速度、またはレスポンスタイムを確実に保証する技術
    - フロー毎の帯域比率を一定に保つ
    - 遅延時間の最大値をある値以下に保つ

イーサネットスイッチのモデル

- ある出力ポートに着目してモデル化

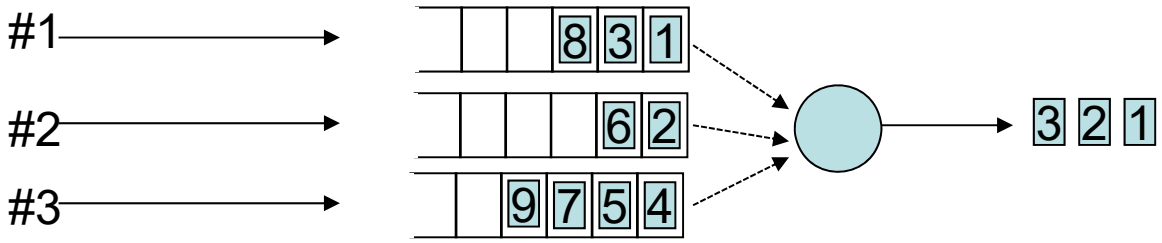


実際のバッファの構成方法



- バッファからの読み出し方を変えることで、パケットのスケジューリングができる。

FIFO (First In First Out)



- フローの到着順に従って出力する。
  - 各フローの出力レートは、入力レートに比例する。
    - サービスレートは入力レートに比例
  - サイズの大きなパケットが先着すると、長時間出力を占有する。
  - 同時に到着したパケットは、例えばポート番号の小さい方が先着とみなされる。
    - #1のサイズの大きなパケットが優先的にサービスを受けることになる。

Fairness Index

- *N*個のフローに対して Fairness Indexを次式で定義
  (x_j: ターゲットとなる項目(遅延、スループット等))

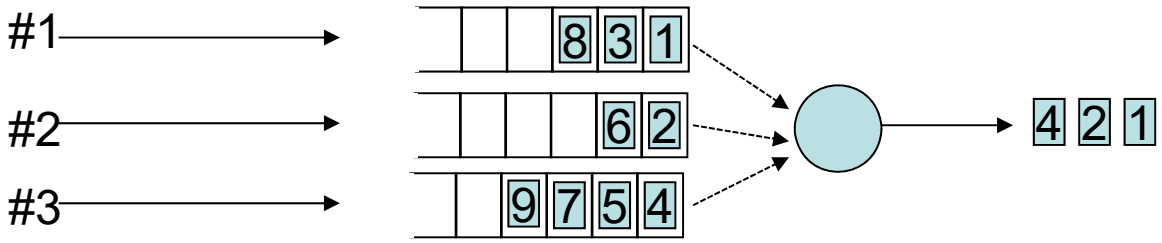
$$f = \frac{\left(\sum_{j=1}^{N} x_j\right)^2}{N\sum_{j=1}^{N} x_j^2}$$

- *f* は0以上1以下の値
  - 1に近いほど公平さが高いと言う

ネットワークシミュレータにおけるFIFOの実装

- DropTail
  - 到着順にパケットバッファにパケットを格納
    - パケットバッファからの読み出しレート (b/s や パケット/s) を指定
    - パケットバッファの大きさ (パケット数 や Byte) を指定
  - パケットバッファが一杯になると、到着したパケットを廃棄
  - 多くの場合は、リンクの属性として指定
    - リンクの長さ(遅延時間 (ms) で指定)、容量 b/s、バッファ長 (格納パケット数)

RR (Round Robin) scheduling



- フローに順番にサービスを割り当てる。
  - パケットサイズはフロー毎に固定とする。
    - 一度に整数個のパケットを読みだすことが可能。
  - フロー数が n で、一度に送出するパケットサイズが一定長で、s_j (j=1, …, n)とする。サービスレート ρ_j は

$$\rho_j = \frac{s_j}{\sum_{j=1}^{n} s_j}, \quad for \quad j = 1, \cdots, n$$

RR (Round Robin)

$$\rho_j = \frac{s_j}{\sum_{j=1}^{n} s_j}, \quad for \quad j = 1, \cdots, n$$

- 読み出し容量が 1Mbps、パケットサイズがそれぞれ 64 byte、1500 byte、9000 byteの3種類のフローに対してサービスを実行する場合
  - 64 : 1500 : 9000 = 6 kbps : 142 kbps : 852 kbps
    - 順番に読みだすのでフロー容量がパケットサイズに比例してしまう

dRR (deficit Round Robin) scheduling

- **パケットサイズが可変の場合のRR** 1995年に発明された
  - Deficit counter (不足カウンタ)を設け、送出できるデータの長さを counter 以下になるようにする。

(1) フローj ($j = 1,2,\cdots,n$)に対して、$counter_j$の初期値を$counter_j(0) = \rho_j \cdot \bar{s}$
$\bar{s}$は１ラウンド当たりのパケット平均サイズ

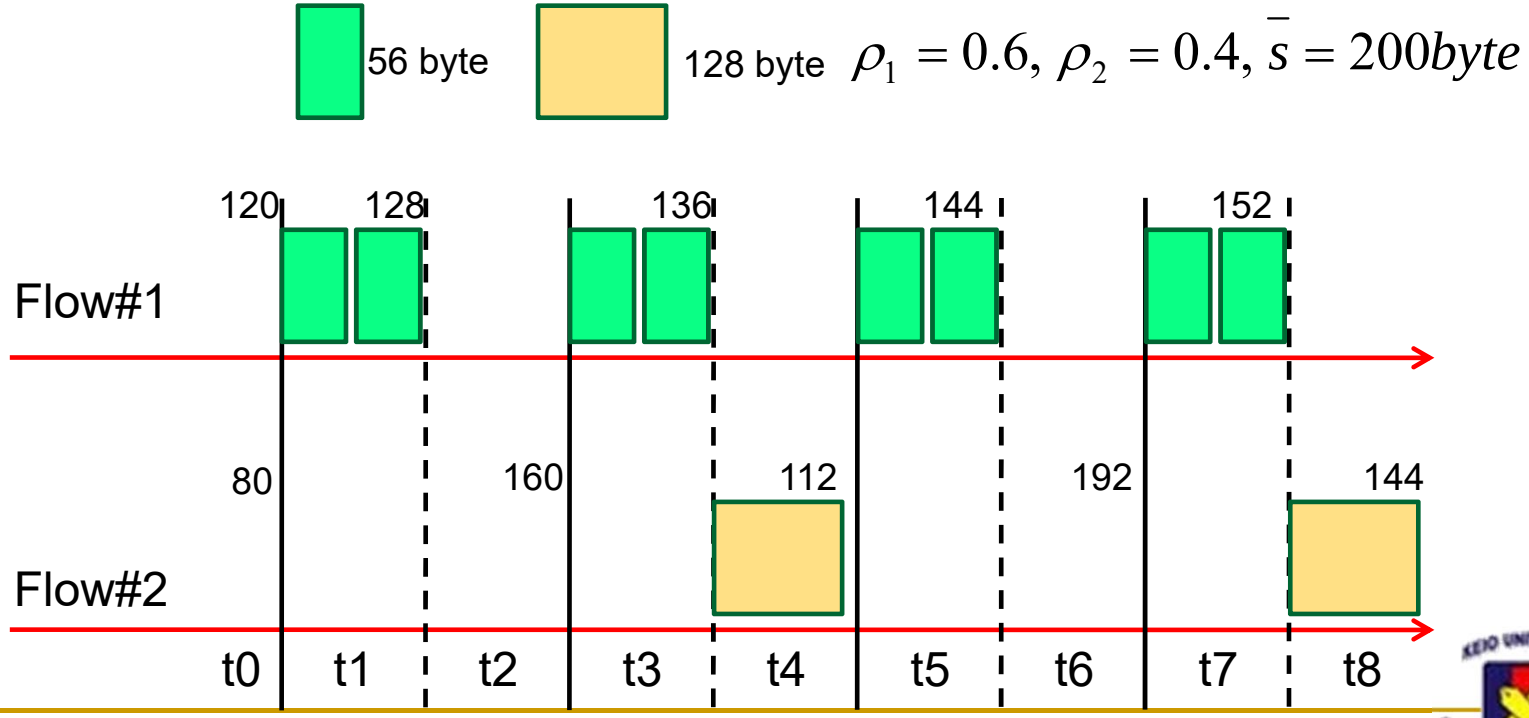
(2) 各フローに対して(3)と(4)を実行

(3) ラウンドtにおいて、フローjのデータを、合計サイズが$counter_j(t)$以下となるようできるだけ送出する。送出サイズ$s_j(t) \le counter_j(t)$
$counter_j(t) = counter_j(t) - s_j(t)$

(4) $counter_j(t+1) = counter_j(t) + \rho_j \bar{s}$

dRRの特徴

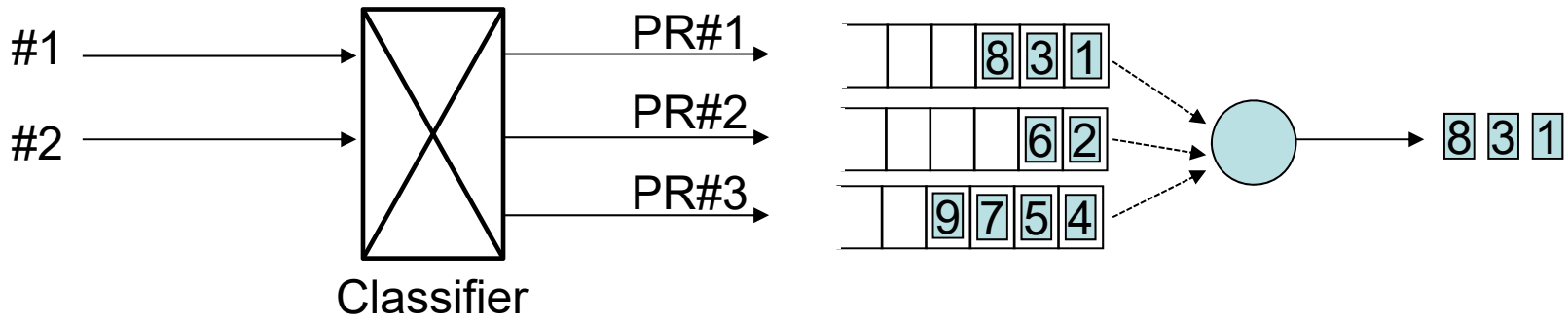
- 平均的に $\rho_j \cdot \bar{s}$ 分のデータが送出される。
- 余った分は次回に持ち越せる



ネットワークシミュレータにおけるRR/dRRの実装

- 歴史あるNS2 では、RR/dRRはきちんと実装されている
- 最近出てきたNS3 では、この辺りはまだまだ発展途上
  - 誰かが作って公開しているものを改良していくフェーズ
  - 公式には DropTailQueue と、よりプリミティブな Queue の二つの Class しかない

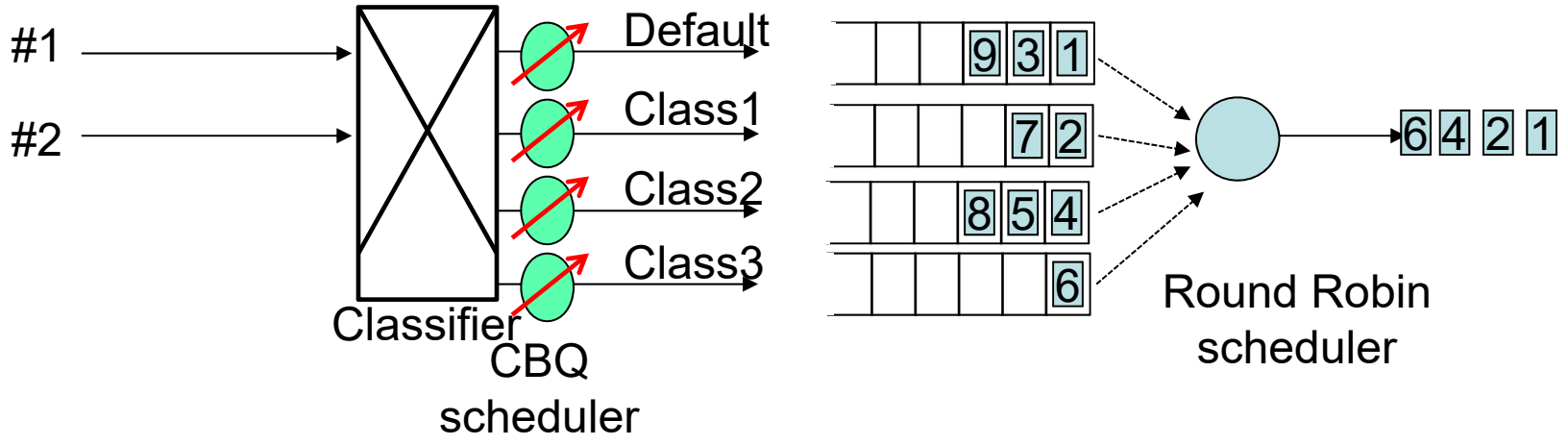
Priority Scheduling



- 優先度が高いキューから順にパケットを取りだす。
  - 優先度が高いクラスのリアルタイム性を保証できる
  - 優先度が低いクラスのパケットは出ていけなくなる
    - 最低限のレート保証を加えたものが CBQ (Class Based Queuing)

CBQ (Class Based Queuing)

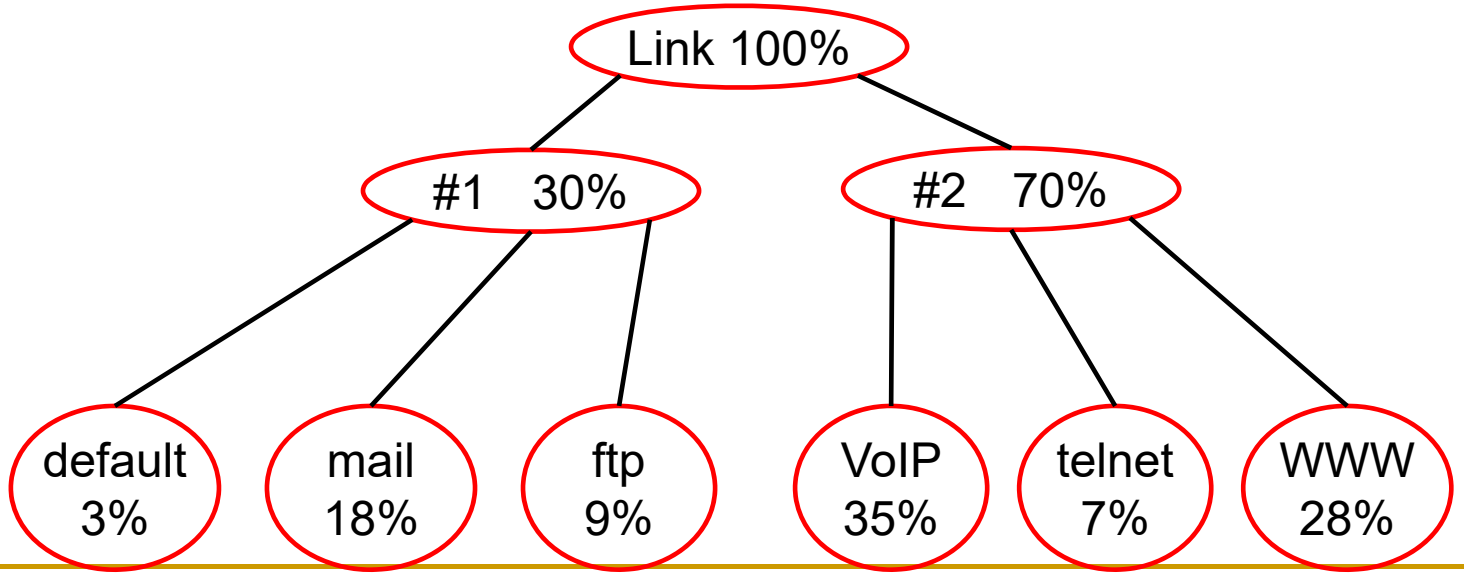
1991年に発明された



- フローをクラスに分類
  - クラス毎にサービスレートを割り当てる
    - フローに所定をレートを割り当てて確保 → RR (重み付)
    - クラスに所定のレートを割り当てるように入力フローを制御
      - 制御しないとバッファ溢れ

CBQ Link-share Tree

- CBQリンクの容量の分け方を示すグラフ
  - From #1, ftp 30%, mail 60%, default 10%
  - From #2, VoIP 50%, telnet 10%, WWW 40%
  - From #1 : From #2 = 30 : 70



ネットワークシミュレータにおけるCBQの実装

- NS2、あります
- NS3、自分で作るしか無い
  - CBQ Link-share Tree を作成する
    - アプリケーション/フロー毎の容量を決定
  - CBQ Link-share Tree をCBQ内部リンクに割り当て
    - CBQ内部リンクに対してRR/dRRで読み出し
  - フローにフローIDを与え、フローIDがCBQのどのキュークラスに収容されるのかを対応づける

FQ (Fair Queuing)

1985年に発明された

- Max-Min法
  - ユーザの需要レベルに応じてリソースを割り当てる
  - どのユーザに対しても需要より多いリソースは割り当てない
  - リソースが不足した場合、需要が完全に満たされないユーザには、残りのリソースを平等に割り当てる
- 物理キューを論理キューに分割し、フロー毎に論理キューを割り当てる
  - フローの数だけ論理キューを作成する必要がある

FQの動作アルゴリズム

$N_s(p)$: パケットpの送出シーケンス番号

N_r : パケットが到着した時 のラウンド番号

$N_{\max}(F)$: パケットpが到着した時に、フロ ーF$に割り当てられている論理キュー内の最大シ ーケンス番号

$L(p)$: パケットpのサイズ

$N_s(p)$はフローF$に割り当てられている 論理キューが空かどう かで場合分け

空の時 : $N_s(p) = L(p) + N_r$

空じゃない時 : $N_s(p) = L(p) + N_{\max}(F)$

シーケンス番号は，パケット到着時に一回だけ付与される

パケット送出の際には $N_s(p)$の小さい順で送出。

送出されたパケットの シーケンス番号を新し いN_r$とする。

FQの動作例

パケットサイズが夫々 $256B$、$128B$、$64B$のフローF^1、F^2、F^3

ある時刻に、F_1^1、F_2^1、F_3^1、F_1^2、F_1^3の順番で空の論理キュ ーに到着

ラウンド番号の初期値 $N_r = 100$

$[1]\ N_s\left(F_1^1\right) = 256 + N_r = 256 + 100 = 356$

$[2]\ N_s\left(F_2^1\right) = 256 + N_{\max}\left(F^1\right) = 256 + 356 = 612$

$[3]\ N_s\left(F_3^1\right) = 256 + N_{\max}\left(F^1\right) = 256 + 612 = 868$

$[4]\ N_s\left(F_1^2\right) = 128 + N_r = 128 + 100 = 228$

$[5]\ N_s\left(F_1^3\right) = 64 + N_r = 64 + 100 = 164$

従って、F_1^3、F_1^2、F_1^1、F_2^1、F_3^1の順に送出され、 $N_r = 868$に更新

WFQ (Weighted Fair Queuing) 1990年に発明された

- FQでは、フロー毎の優先度は無い
- WFQはFQの各論理キューに重みを付け、重みwに応じた送出順番を決定

$$w = \frac{C}{precedence + 1}, \quad C : \text{定数(例32768)}$$

  - Precedence (IPヘッダのTOS (Type of Service)フィールドの優先度3bit) ~ 0-7 (7が最優先)
- シーケンス番号ベース、フローベースの二種類がある。

シーケンス番号ベースのWFQ

$N_s(p)$: パケットpの送出シーケンス番号

N_r : パケットが到着した時 のラウンド番号

$N_{\max}(F)$: パケットpが到着した時に、フロ ーF に割り当てられている論理キュー内の最大シ ーケンス番号

$L(p)$: パケットpのサイズ

$w(p)$: パケットpの重み

$$w = \frac{C}{precedence + 1}, \quad C: \text{定数(例32768)}$$

$N_s(p)$はフローFに割り当てられている 論理キューが空かどう かで場合分け

空の時 : $N_s(p) = w(p) \cdot L(p) + N_r$

空じゃない時 : $N_s(p) = w(p) \cdot L(p) + N_{\max}(F)$

シーケンス番号は，パケット到着時に一回だけ付与される

パケット送出の際には $N_s(p)$の小さい順で送出。

送出されたパケットの シーケンス番号を新し いN_rとする。

フローベースのWFQ

- 重みをフローに割り当てる
  - 同じフローでprecedenceが異なるパケットは、同じ重みとなる。

$N_s(p)$: パケットpの送出シーケンス番号
N_r : パケットが到着した時 のラウンド番号
$N_{\max}(F)$: パケットpが到着した時に、フロ ーF に割り当てられている
論理キュー内の最大シ ーケンス番号
$L(p)$: パケットpのサイズ
$w(F)$: フローFの重み

$N_s(p)$はフローFに割り当てられている 論理キューが空かどう かで場合分け
空の時 : $N_s(p) = w(F) \cdot L(p) + N_r$
空じゃない時 : $N_s(p) = w(F) \cdot L(p) + N_{\max}(F)$

パケット送出の際には $N_s(p)$の小さい順で送出。
送出されたパケットの シーケンス番号を新し いNrとする。

シーケンス番号は，パケット到着時に一回だけ付与される

フローベースWFQの動作例

パケットサイズが夫々 $256B$、$128B$、$64B$のフローF^1、F^2、F^3

重み200、2,000、2,000

ある時刻に、 F_1^1、F_2^1、F_3^1、F_1^2、F_1^3の順番で空の論理キュ ーに到着

ラウンド番号の初期値 $N_r = 100$

$[1]\ N_s\left(F_1^1\right) = 256 \cdot w\left(F^1\right) + N_r = 256 \times 200 + 100 = 51{,}300$

$[2]\ N_s\left(F_2^1\right) = 256 \cdot w\left(F^1\right) + N_{\max}\left(F^1\right) = 256 \times 200 + 51{,}300 = 102{,}500$

$[3]\ N_s\left(F_3^1\right) = 256 \cdot w\left(F^1\right) + N_{\max}\left(F^1\right) = 256 \times 200 + 102{,}500 = 153{,}700$

$[4]\ N_s\left(F_1^2\right) = 128 \cdot w\left(F^2\right) + N_r = 128 \times 2{,}000 + 100 = 256{,}100$

$[5]\ N_s\left(F_1^3\right) = 64 \cdot w\left(F^3\right) + N_r = 64 \times 2{,}000 + 100 = 128{,}100$

従って、 F_1^1、F_2^1、F_1^3、F_3^1、F_1^2の順に送出され、 $N_r = 256{,}100$に更新

ネットワークシミュレータにおけるFQ/WFQの実装

- NS2、FQはある
- NS3、もちろん FQもWFQも無い

- 最近の機器は WFQを頑張って実装

演習10

パケットサイズがそれ ぞれ256 byte、128 byte、64 byteのフローF^1、F^2、F^3がある。

$t=0: F_1^1, F_2^1, F_3^1, F_1^2, F_1^3$

$t=1: F_2^2, F_4^1, F_3^2, F_2^3, F_5^1$

$t=2: F_6^1, F_3^3, F_4^3, F_4^2, F_7^1, F_5^2$

$t=3: F_5^3, F_6^3, F_8^1, F_6^2$

フローベース $WFQ, w(F^1)=10, w(F^2)=20, w(F^3)=100$,出力ポート容量 900 byte

各時刻の出力パケット を求めよ。 N_r の初期値は100とする。

ヒント

900 byteを超える順番のパケッ トは、その時刻には出 せず、

次の時刻に持ち越す。

N_rはその時刻に最後に出 せたパケットのシーケ ンス番号に更新。

同じシーケンス番号の パケットは、ここでは 入力の早い順で出力。