次世代ネットワークとQoS

大阪大学サイバーメディアセンター
先端ネットワーク環境研究部門
村田正幸
e-mail: murata@cmc.osaka-u.ac.jp
http://www-ana.ics.es.osaka-u.ac.jp/

Advanced Network Architecture Research

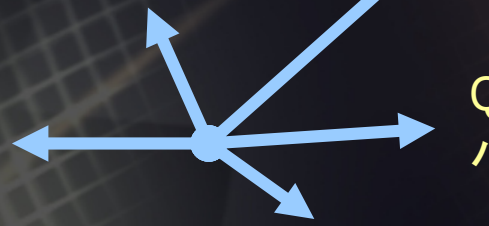
目次

- 背景
- インターネットフロー間の不公平
- 公平性向上のための手段
  - プロトコルの改善
  - ルータバッファでの制御
  - エンドホストでの資源管理
- まとめ

背景

- これまでのインターネット
  - ベストエフォートネットワーク
  - QoS保証は行われない
    - 帯域、遅延等
    - TCPを使えば、「確実に相手に届く」だけは何とかなる
  - 繋がることが重要だった
    - 当然、ユーザー間、フロー間の公平性は期待できない
- バックボーン、アクセス回線速度の飛躍的な向上
  - 「繋がる」以上のサービスへの要求
  - QoS保証
- 「公平性」が重要な要素になりつつある
  - 同じ料金なのに速度が違う
  - 2倍のアクセス速度を2倍の料金で使っているのにスループットは同じ

ネットワークQoSのための要素



QoS目標:
保証、差別化、無保証

適用対象:
パケット、フロー、クラス、、、

時間粒度:
パケット、フロー、
コネクション、
プロビジョニングレベル

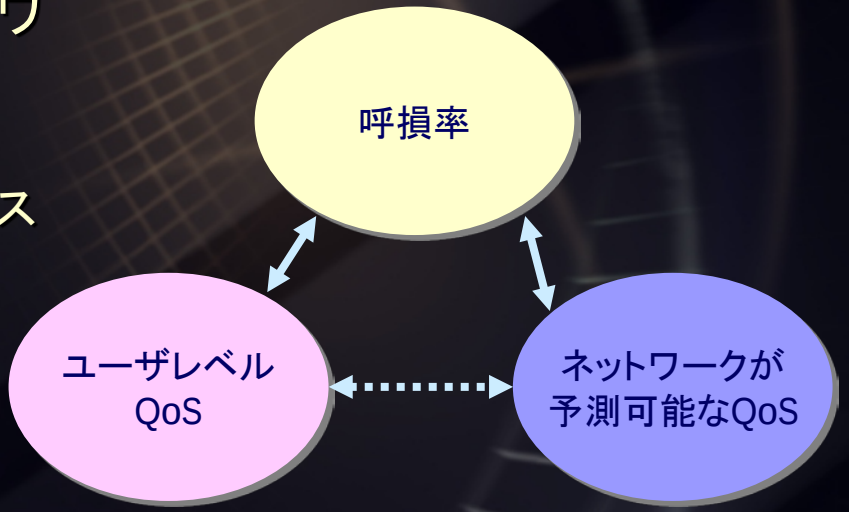
QoSパラメータ:
パケット遅延、棄却率、スループット、、、

対象トラヒック:
実時間系、データ系、VPN

- 例
  - これまでのインターネット:全トラヒックを対象、無保証
  - IntServ:フローを単位とした遅延保証(実体はスループット保証)
  - DiffServ:クラスを単位とした遅延・スループットの差別化

電気通信網におけるQoS

1. 過去の統計量に対する蓄積
   トラヒック特性
2. （古くは）単一キャリア、単一ネットワーク
3. アーラン呼損式
   ローバスト（ポアソン到着、一般サービス時間分布）
4. QoS測定　＝　呼損率
   キャリアが測定可能
5. 実時間転送；音声、動画像
   帯域保証のみで十分
   エンド間保証が前提



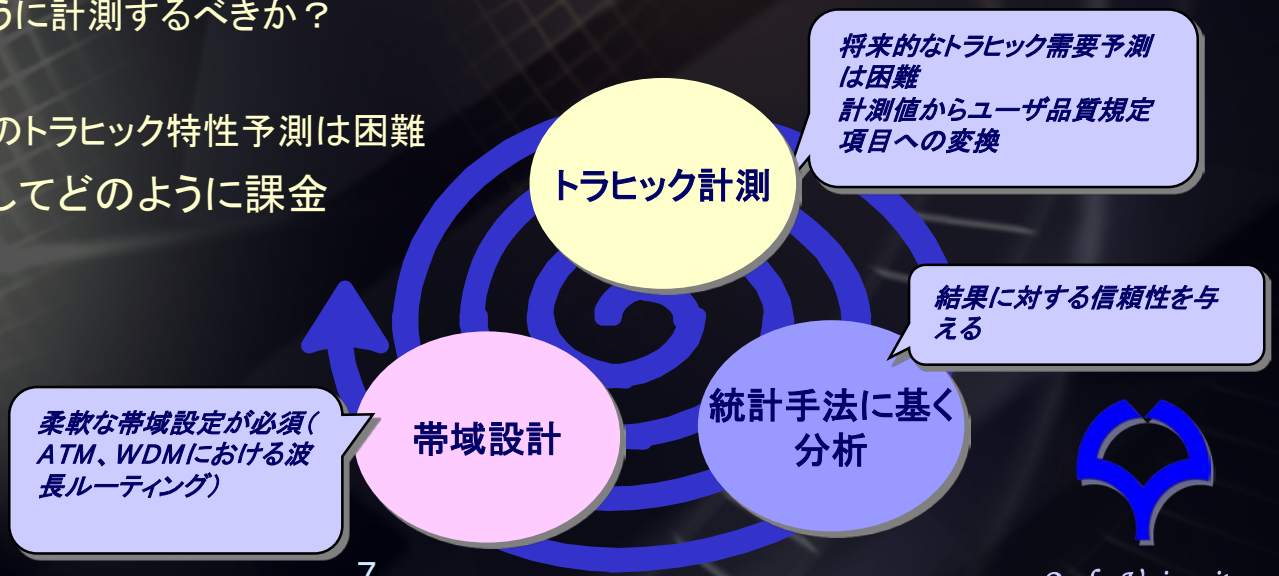
データ系アプリケーションにおけるQoSとは？

- データ系は帯域を食い尽くすアプリケーション
  - アクセス回線、エンドホストの高速化
  - TCP（＝エンドホスト）が輻輳制御を行う
  - バックボーンの高速化は解決策にならない
    - これまではアクセス回線がボトルネックになっていただけ
- データ系に適したQoS機構？
  - IntServによるQoS保証
    - パケット棄却率、パケット遅延を「保証」できるか？
    - トラヒック契約の考え方（QoSパラメータ、トラヒック特性を事前に申告）とマッチしない
    - RSVPのスケーラビリティに対する限界
  - DiffServによるクラスに対するQoS差別化
    - 実現はHOL優先権制御で十分（AFクラス）
    - 相対的なQoSはユーザのQoS要求とマッチするか？
  - QoS「保証」「差別化」なし
    - ただし、ネットワークプロビジョニングレベルでのQoS監視は重要
    - 帯域切り出し（VPN）は意味がある
- データ系アプリケーションQoSの3原則

1. Data applications try to use the bandwidth as much as possible.
2. Neither bandwidth nor delay guarantees should be expected. Only network provisioning can satisfy user's QoS requests.
3. Competed bandwidth should be fairly shared among active users.

データ系アプリケーションに適したQoS制御：スパイラルアプローチ

- 少なくともプロビジョニングレベルにおけるQoS予測が必要
- 電気通信網ではなかった新たな問題
  - データ系QoSとは何か？
    - パケット遅延、棄却率はエンドユーザレベルの性能指標ではない
    - エンド間QoSはユーザしかわからない（エッジルータの可能性はありうる）
    - QoSをどのように計測するべきか？
    - トラヒック変動
    - エンドユーザのトラヒック特性予測は困難
  - 「サービス」に対してどのように課金するべきか？



トラヒック計測の2つのアプローチ パッシブ／アクティブ

- パッシブな計測
  - OC3MON, OC12MON, ...
  - 点観測
    - 経路制御による経路の不安定性
    - TCPの誤り制御によるセグメント再送
    - 例：利用率が低いのは輻輳制御のため？　エンドユーザのアクセス回線が細いため？　エンドホストのパワー不足？
    - ストリーミングメディアのレート制御
  - ユーザQoSは不明
- アクティブな計測
  - Pchar, Netperf, bprobe, ...
  - エンド間ユーザQoSの計測
  - ある特定のユーザのQoSがわかったとしてもネットワーク設計ができるわけではない
  - ネットワークトラヒックの変動への対処

トラヒック計測によるボトルネック特定

- トラヒック計測によるボトルネック特定
  - ネットワーク提供者による容量設計の限界
  - エンドユーザ（エッジルータ）による性能向上策
    - ボトルネックの特定
    - 回線増強へのフィードバック
  - TCP Calculus
    - RTT、パケット棄却率を用いたTCP性能特性の定式化
- エンドユーザによるQoS測定
  - アクティブ計測ツール（帯域推定ツール）pchar, pathchar
    - 計測ホストからProbeパケットを送出
    - RTTを計測
    - 最小RTTから各リンクの帯域（回線容量）を推定

TCP Calculus：TCPの理論値と測定値の比較

- TCP コネクションのウィンドウサイズの期待値 E[W]

$$E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}$$

$E[W] \geq W_{max}$ なら受信側のバッファサイズが不足

- RTT、パケットロス率、ウィンドウサイズの最大値、タイムアウト時の再送間隔を用いて予測

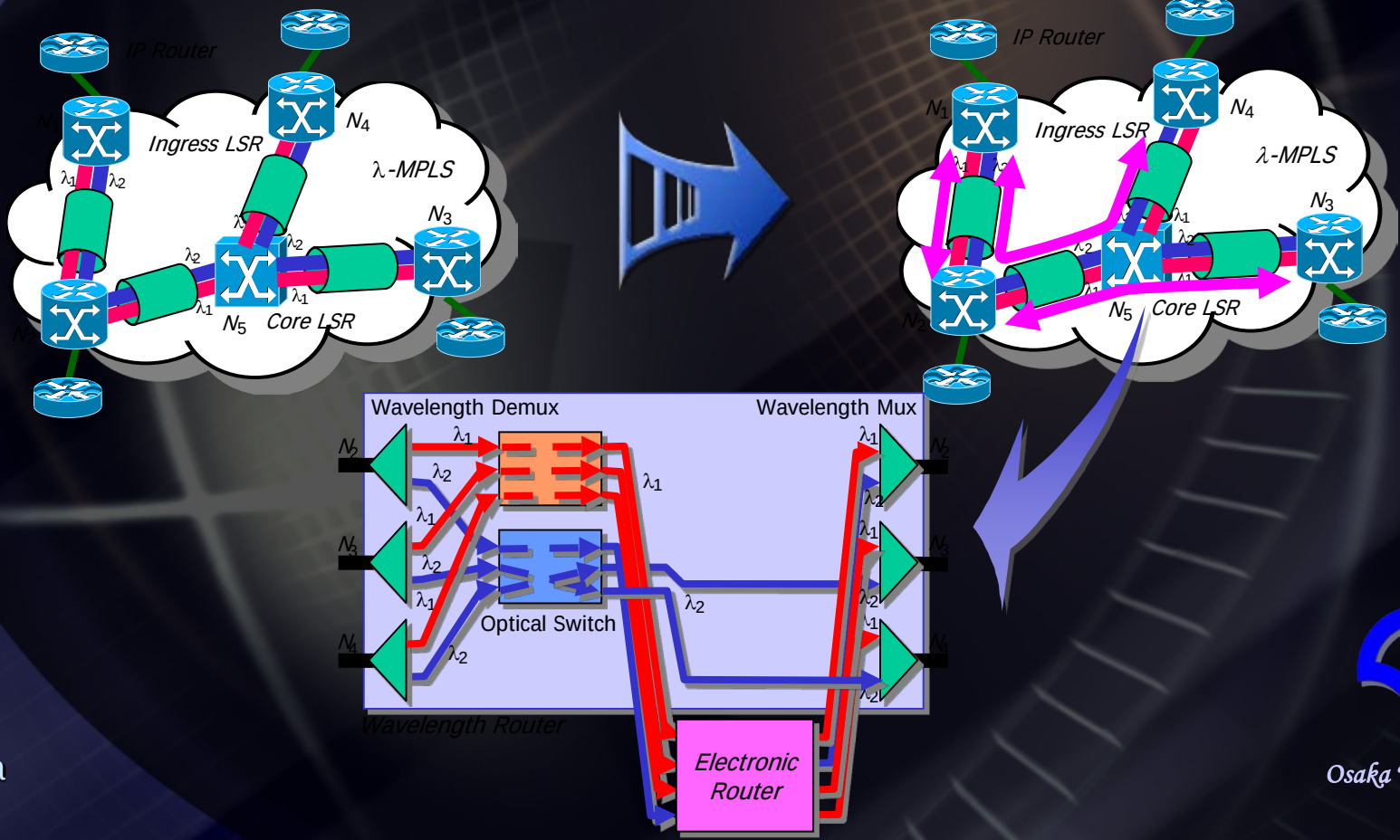
$$\frac{\frac{1-p}{p} + W_{max} + \hat{Q}(W_{max})\frac{1}{1-p}}{RTT\left(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2\right) + \hat{Q}(W_{max})T_0\frac{f(p)}{1-p}} \qquad E[W] \geq W_{max}$$

- *実測値と予測値からボトルネックを特定*
  - 回線容量増大へのフィードバック

柔軟な帯域割当を可能とするインフラ: WDM波長ルーティングの利用

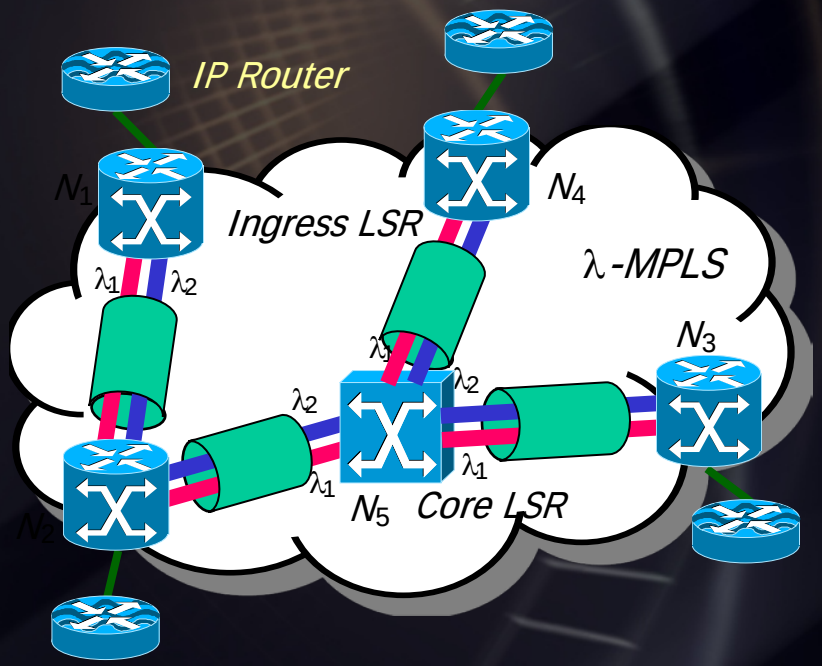
物理トポロジー

論理トポロジー



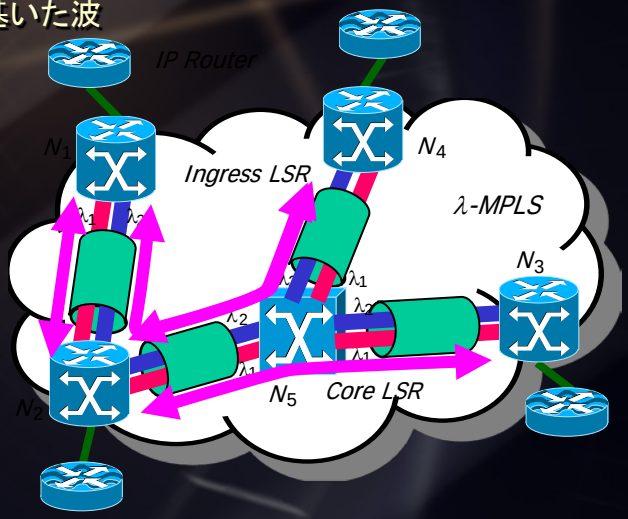
これまでの研究

- 経路／波長割り当て問題 (Routing and Wavelength Assignment: RWA)の例
  - 与条件:
    トラヒック量既知
  - 目的関数:
    利用可能波長を使い切って各波長ごとのトラヒック量を最小化



スパイラルアプローチの実現

- 段階的ネットワーク設計
  - 初期ステップ
    - 与えられたトラヒック量に基いたトポロジー設計；従来の設計手法が適用可能、ただし、トラヒック予測が間違っていたとしても、追加ステップで修正可能
  - 追加ステップ
    - トラヒック測定（パッシブ）、エンドユーザ品質測定（アクティブ）に基いた波長設定
    - 波長の追加、削減のみ
    - バックアップパスの有効利用
  - 調整ステップ
    - 全体の波長有効利用を考慮したトポロジー再設計
    - サービスの継続性を考慮した1波長ルートごとの変更
- WDM技術による高信頼化：IP＆WDM Integration
  - 共有プロテクション方式
    - 複数の障害には対応不可、必要波長数小
  - 追加ステップにおけるバックアップパスの有効利用
  - QoP (Quality of Protection)
- 今後の課題
  - 複数の設定要求をいかに調整するか？
  - 波長不足により設定失敗に終わった時の処理
  - 処理量の把握

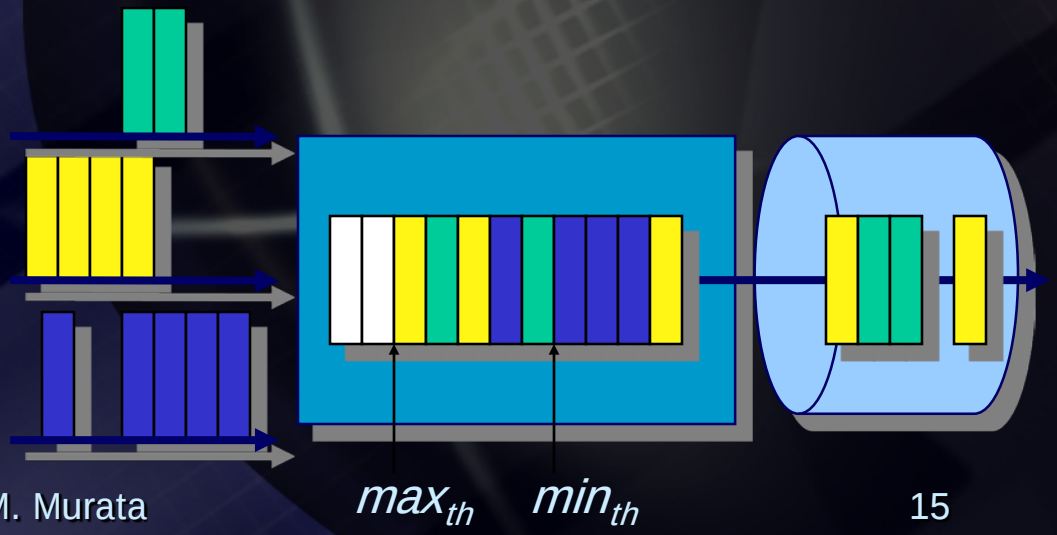


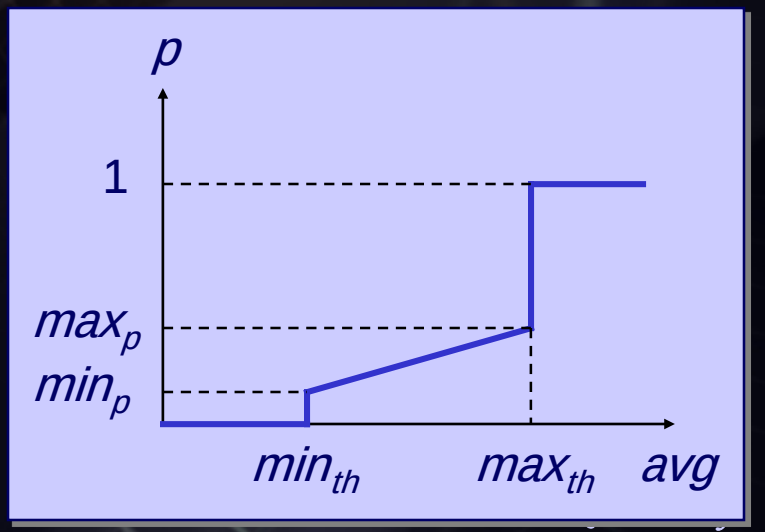
ネットワーク機能の再配分

- エンドホストに頼りすぎ
  - TCPが輻輳制御に責任を持っている
    - 輻輳制御は本来ネットワーク機能
  - 公平なサービスの実現を困難にしている
    - ソフトウェアのバグやコードの書き換えによって、輻輳制御を行わないホストも存在する
    - サービスの有料化を阻害する
- どの機能をネットワークに再配分するべきか？
  - フロー制御、誤り制御、*輻輳制御*、経路制御
  - RED, DRR, ECN, diff-serv, int-serv (RSVP)
  - ネットワークに過度に頼るのはインターネットのメリットを失う

REDのメカニズム

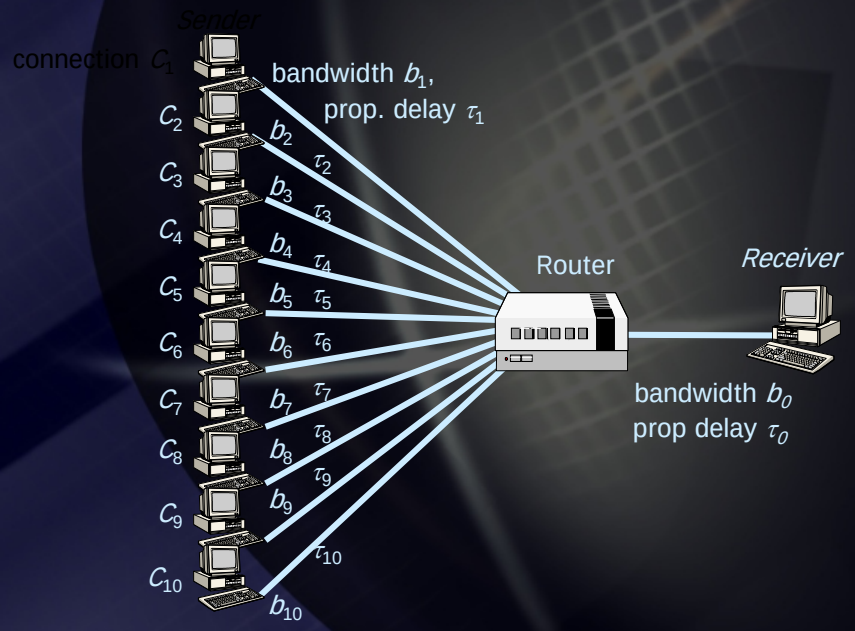
- 平均待ちパケット数に基いてパケット廃棄率を制御する
  - $avg < min_{th}$の時、すべてのパケットを受け付ける
  - $min_{th} < avg < max_{th}$の時、確率 $p(x)$でパケット廃棄
  - $max_{th} < avg$の時、すべてのパケット廃棄
- バースト的に到着する同一フローのパケットのすべてを失うことを予防する：Fast retransmitへの対応



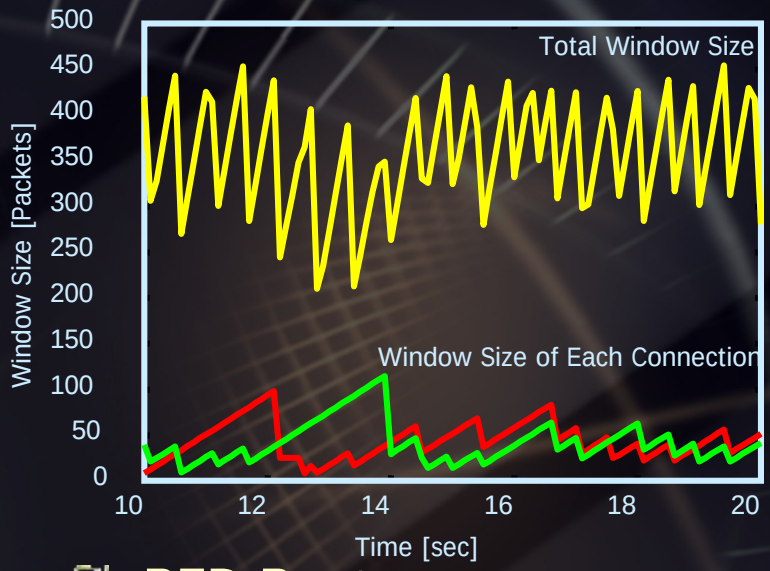


REDの効果

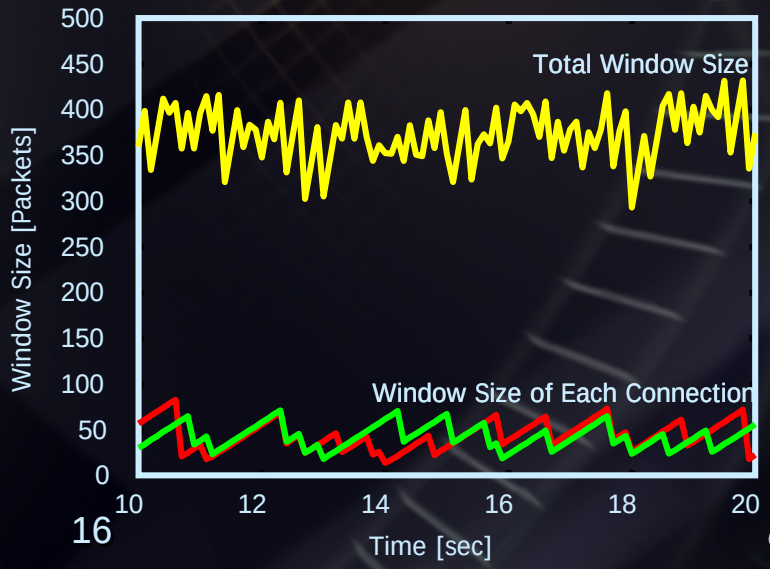
- fast retransmitの起動によってパケット損失の影響を最小限にとどめる
- 一時的に発生する不公平性を緩和



Drop-Tail Router



RED Router



コネクション間の公平性

- TCPは常に帯域に対して貪欲
- 帯域の公平な分配はTCPでは困難
  - ウィンドウサイズの変化による短期的な不公平性
  - 帯域、距離の異なるコネクション間の長期的な不公平性
    TCP Renoの場合

$$S = \frac{1.3 \times MTU}{RTT \times \sqrt{p}}$$

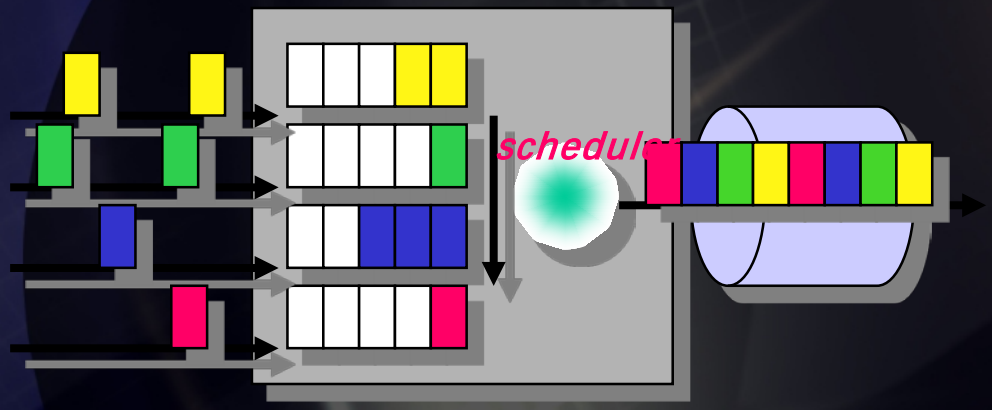
    TCP Reno + REDの場合：RTTにのみ反比例
- ルータバッファでのパケット処理方式
  - プロトコルを意識しないで実現できるか？
    - TCP、UDP間
    - 異なるTCPバージョン間
  - ルータにおけるバッファ管理、スケジューリング; RED, DRR
- 既存プロトコルとの親和性を考慮したプロトコル設計
  - ルータに頼ることなく実現できるか？
  - 実時間アプリケーション (UDP) とデータ系アプリケーション (TCP)の公平性？
    - TCP-Friendlyな輻輳制御

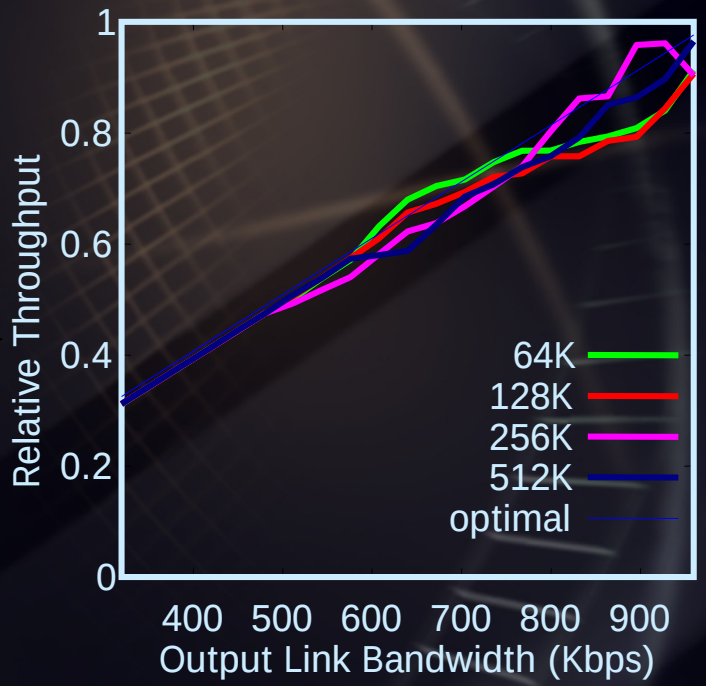
公平性を実現するための第4層スケジューリング

- ステート情報をフルに持つ
  - DRR
- ステートは保持せずにフロー識別を実現（フロー単位のバッファリングではない）
  - FRED
    - アクティブフローごとに到着／処理パケット数をカウントし、バッファ内パケット数を計算。それをREDのパケット棄却率に反映させる
  - Stabilized RED
  - Core Stateless Fair Queueing
    - エッジルータではフローごとのレートを計算、パケットヘッダに書き込む。コアルータでは、アクティブなフロー数を計測、パケットヘッダのレートに基いて、公平な帯域割り当てを計算。それに基いてパケット棄却・処理を決定する

DRRのメカニズム

- ウエイトを考慮したビットレベルでのプロセッサシェアリングを模倣

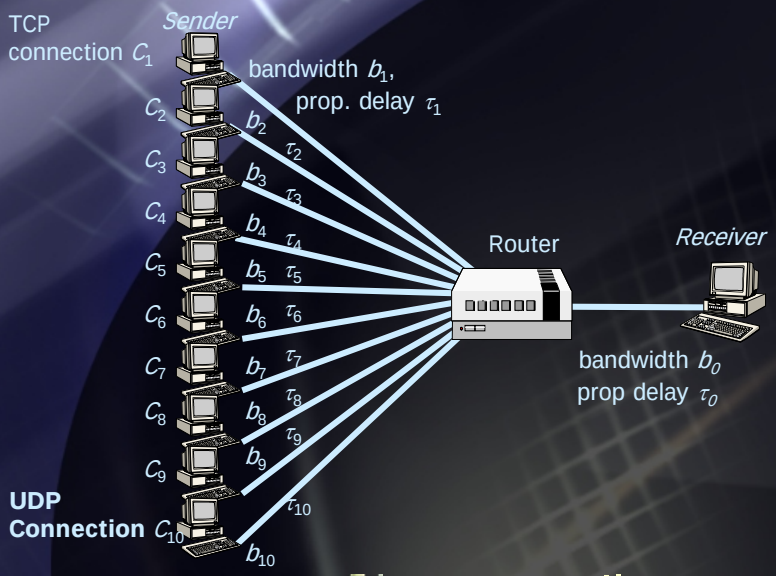




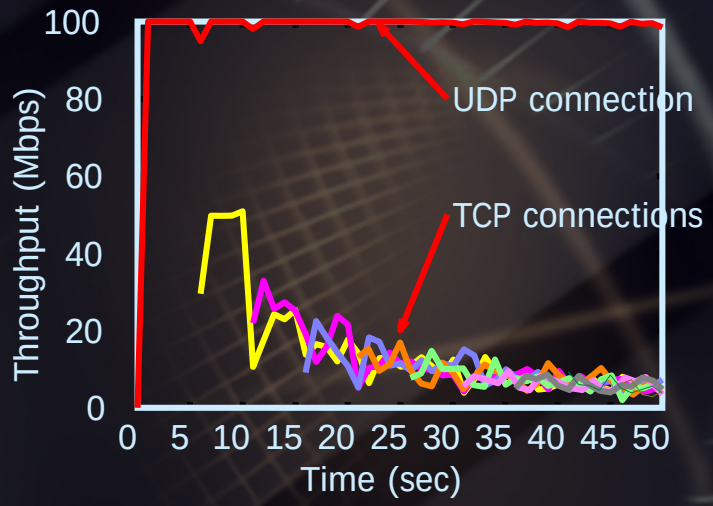
- 物理的(または論理的)にフローごとのパケットバッファリング
- IPアドレスとアクセス回線容量のマッピングを知っておく必要がある

Advanced Network Architecture Research

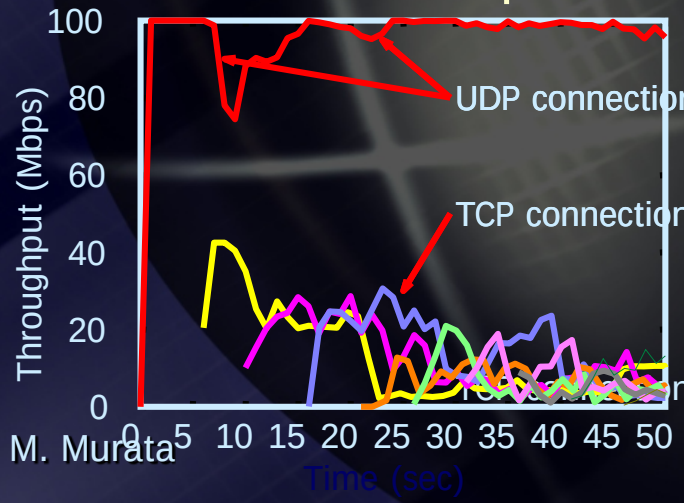
FREDの効果



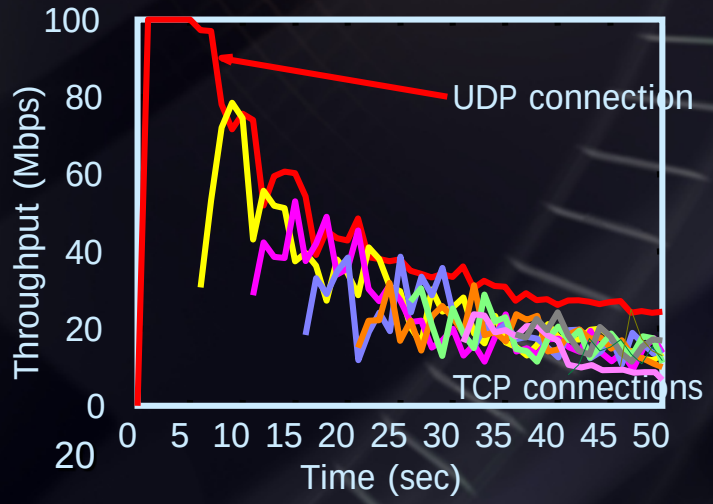
RED Router



Drop-Tail Router



FRED Router



今後？

- QoSに関して
  - バックボーンの高速化：フォトニックインターネット
    - GMPLSに基づくルーティング＋MPλSに基づく波長ルーティング
    - GMPLSに基づくルーティング＋フォトニックパケットスイッチに基づく波長ルーティング
    - フォトニックIPルータ
    - 高機能フォトニックIPルータ
  - エッジルータ、ゲートウェイにおける高品質化
    - プログラマブルルータの活用
  - エンドホストの高速化
    - ムーアの法則：CPUのコストパフォーマンスは18ヶ月で2倍に向上（10年で100倍）
    - ビルジョイの法則(?)：回線容量は9ヶ月～1年で2倍に向上（10年で1,000倍）
- インターネットが目の前にあったからこそ、それに適したWebというアプリケーションが生まれた
  - 背景：画像圧縮技術、GUI、画像表示能力
  - にわとりと卵（？）
  - napster, gnutella
- 波長の有効利用
  - 波長をどれだけエンドユーザの近いところに持ってこれるか？
  - 多重波長数に依存

今後？

- エンド間QoSを保証、差別化することに意味があるか？
  - IntServ、DiffServの前提；回線固定、ノード固定、サーバ固定
  - P2P；サーバが突発的に現れる
  - モバイル環境；情報源が突発的に現れる
- ネットワーク資源の変動を前提とした、アダプティブなエンドホストによるQoS制御
  - 例：ストリーミングサービス vs. リアルタイム動画配信サービス
  - エンドシステムにとって利用可能な資源の実時間推定
  - ネットワークの資源管理は補助的な役割

参考文献

- Masayuki Murata, “Challenges for the Next Generation Internet and the Role of IP over Photonic Networks,” *IEICE Transactions on Communications*, Vol. E83-B, No. 10, October 2000.
- Go Hasegawa and Masayuki Murata, “Survey on Fairness Issues in TCP Congestion Control Mechanisms,” to appear in *IEICE Transactions on Communications*, January 2001.
- Masayuki Murata, “On A Network Dimensioning Approach for the Internet,” to appear in *IEICE Transactions on Communications*, 2001.
- Masayuki Murata and Ken-ichi Kitayama, “A perspective on photonic multi-protocol label switching,” to appear in *IEEE Network Magazine*, August 2001.
- Shin’ichi Arakawa and Masayuki Murata, “Lightpath Management of Logical Topology with Incremental Traffic Changes for Reliable IP over WDM Networks,” to be presented at *OptiComm*, 2001.
- Go Hasegawa, Tatsuhiko Terai and Masayuki Murata, "Scalable socket buffer tuning for high-performance Web servers," to be presented at *IEEE ICNP 2001*.
- Kazumine Matoba, Shingo Ata, Masayuki Murata, “Capacity Dimensioning Based on Traffic Measurement in the Internet,” to be presented at *IEEE Globecom*, 2001.

背景

- バックボーン、アクセス回線速度の飛躍的な向上
  - 「繋がる」以上のサービスへの要求
  - QoS保証
- 「公平性」が重要な要素になりつつある
  - 同じ料金なのに速度が違う
  - 2倍のアクセス速度を2倍の料金で使っているのにスループットは同じ

公平性の定義

- 場所によって異なる
  - 同じ輻輳ルータを通るフローは同じスループット
  - アクセス回線速度に応じたスループット
- フローとは？
  - 個別のTCPコネクション
  - ユーザ単位

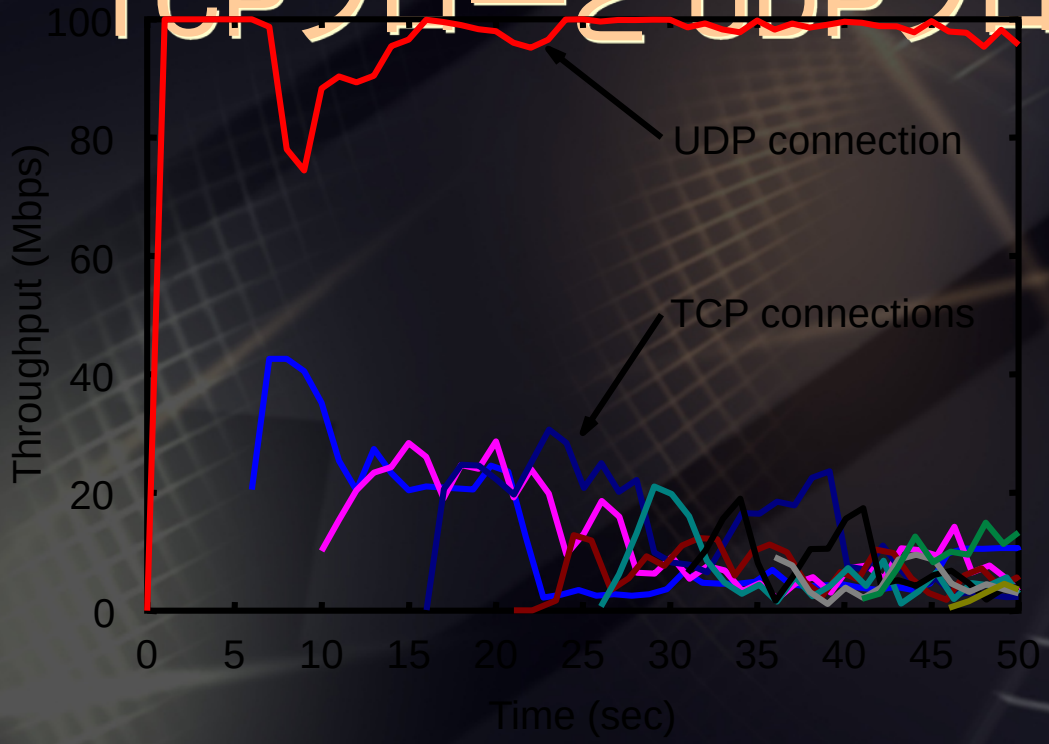
インターネットフロー間の不公平

- プロトコルの違い
  - TCPとUDP
  - TCPのバージョン
- TCPコネクションの環境の違い
  - 遅延（距離）
  - 帯域
  - …

TCPフローと UDPフロー

- UDPは輻輳制御を行わない
  - ネットワークの輻輳に関係なく一定のレートでパケットを転送
- TCPフローと UDPフローが混在すると
  - 輻輳時にTCPフローは転送レートを下げる
  - UDPフローが帯域を占有
- TCPが輻輳制御を行わないように改良（改悪？）することも容易

TCPフローとUDPフロー



- TCPフローが増えても、UDPフローのスループットは下がらずに帯域を占有

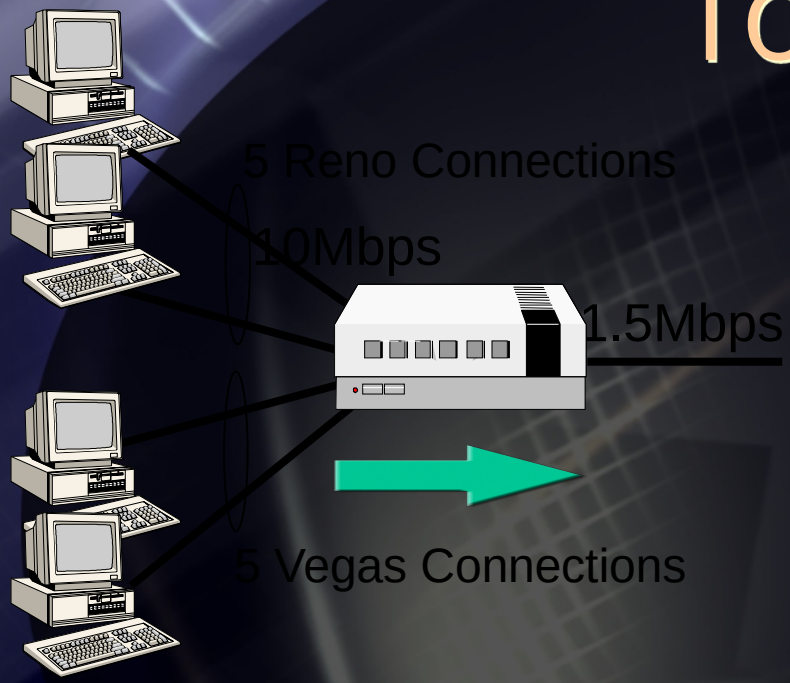
TCPのバージョン

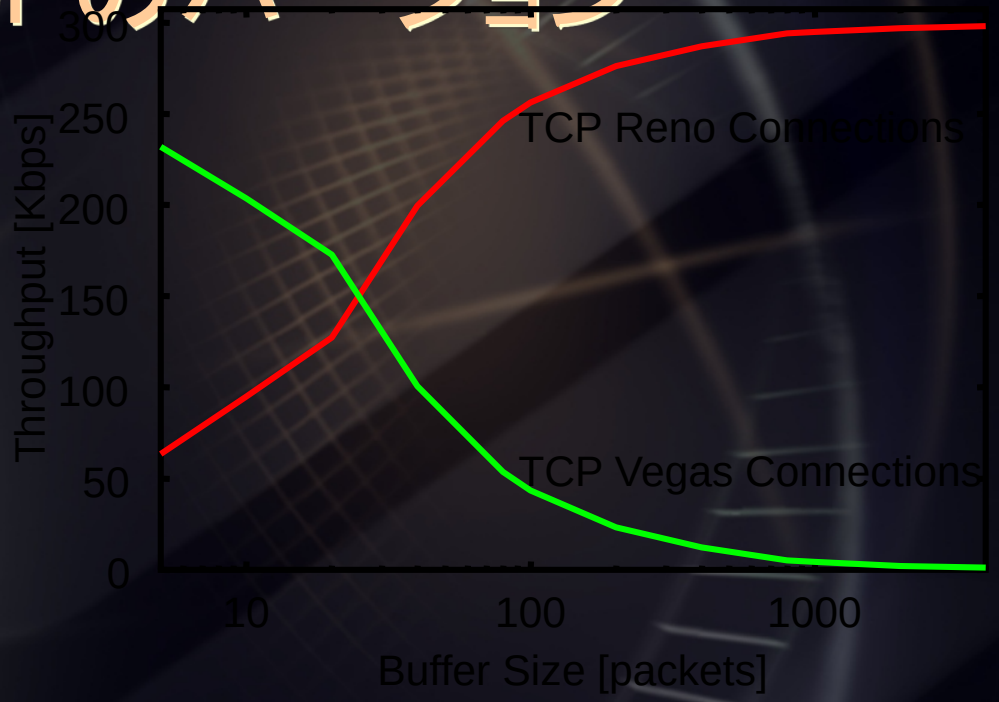
- TCP Tahoe, TCP Reno
  - 現在のほとんどのOSの実装ベース
  - パケットロスのみで輻輳を検知
- OSによって実装はバラバラ
  - タイマ処理
  - ウインドウサイズの初期値
  - スループットに大きく影響

TCPのバージョン

- TCP Vegas
  - RTT (Round Trip Time) の大小で輻輳を検知
  - Tahoe/Renoに比べて高いスループットが得られる
- RenoとVegasの差異
  - Renoは積極的に転送レート (ウィンドウサイズ) を上げ、パケットロスを前提とした制御を行う
  - VegasはRTTの増加をパケットロスの前兆ととらえ、パケットロスが発生しないように制御を行う

TCPのバージョン





- 混在すると、性能がいいはずのVegasが積極的なRenoに負ける

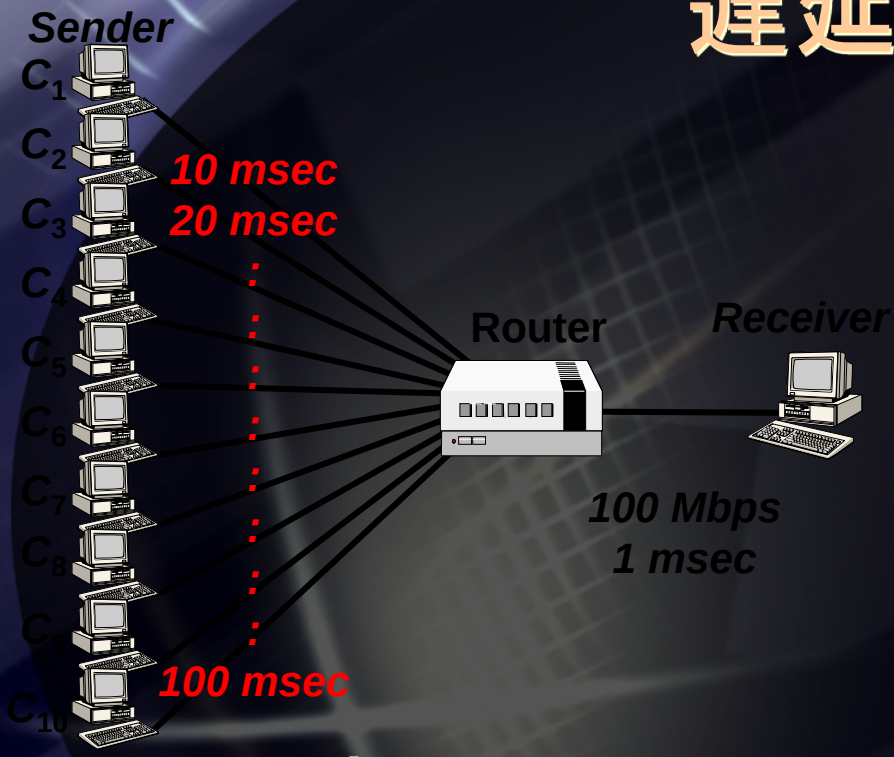
距離（伝播遅延時間）の違い

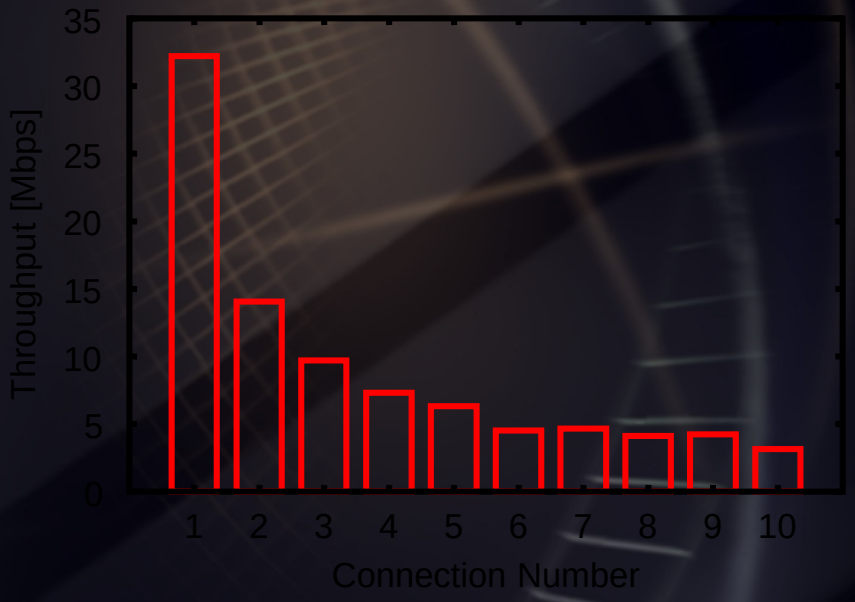
- TCPのスループットは送受信間の距離に大きく影響される

$$\rho = \max\left(\frac{W_{\max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p\left(1+32p^2\right)}\right)$$

J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP throughput: a simple model and its empirical validation,” in *Proceedings of ACM SIGCOMM’98*, pp. 303–314, August 1998.

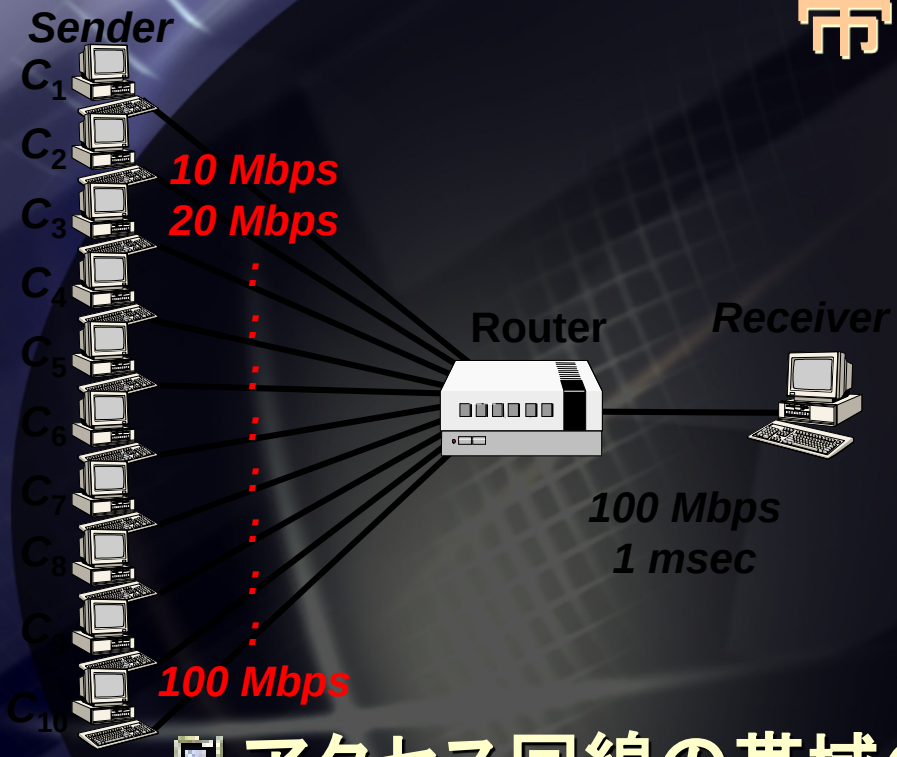
遅延時間の違い

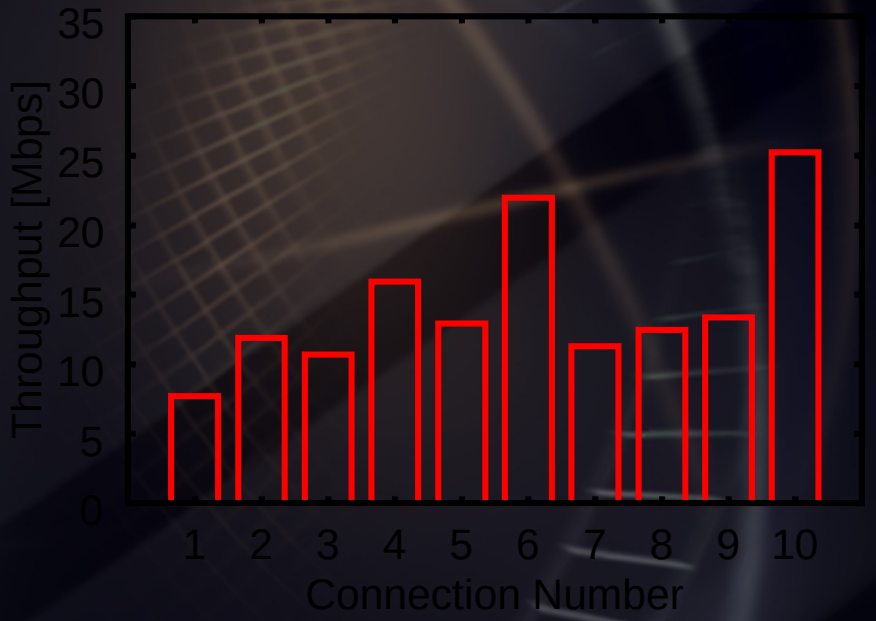




- 距離の大きいコネクションは非常に不利

帯域の違い





- アクセス回線の帯域の違い
  - スループットは等しくならない
  - アクセス回線の帯域に比例もしない

その他

- ホップ数
  - 通過する輻輳ルータの数が多いほどスループットは下がる
- 転送するファイルサイズの差
  - 小さいファイルを転送するコネクションのスループットは小さくなる
  - Slow Startが原因
- 送受信端末のバッファサイズ
  - コネクションの状態に関係なく固定値が割り当てられる
- …

公平性改善の方法

- 送信側端末での制御
  - UDPフローのTCP-friendly制御
  - TCPプロトコルの改善
- ルータバッファでの制御
  - AQM (Active Queue Management)
- エンドホストでの資源管理

TCP-friendly制御

- TCP-friendlyとは
  - UDPフローは、同一パスを通るTCPフローと平均のスループットが同じであるべき
  - アプリケーションが制御を行う
  - 2つの制御方法
    - TCPスループット推測の式を利用する
    - TCPと同様のAIMD (Additive-Increase Multiplicative Decrease) 制御を行う
- TCPとの公平性を向上

TCPプロトコルの改善

- TCPの本質的な欠点
  - フロー制御とエラー制御（再送制御）の区別が非常にあいまい
  - パケット単位の制御と、バイト単位の制御が混在している
- 改善方式
  - TCP Vegasなどさまざまな方式が提案
  - TCPに変わる全く新しいプロトコルを考えるのは非現実的
  - 段階的な実装、新旧TCP間の公平性を考慮する必要がある

ルータバッファでの制御

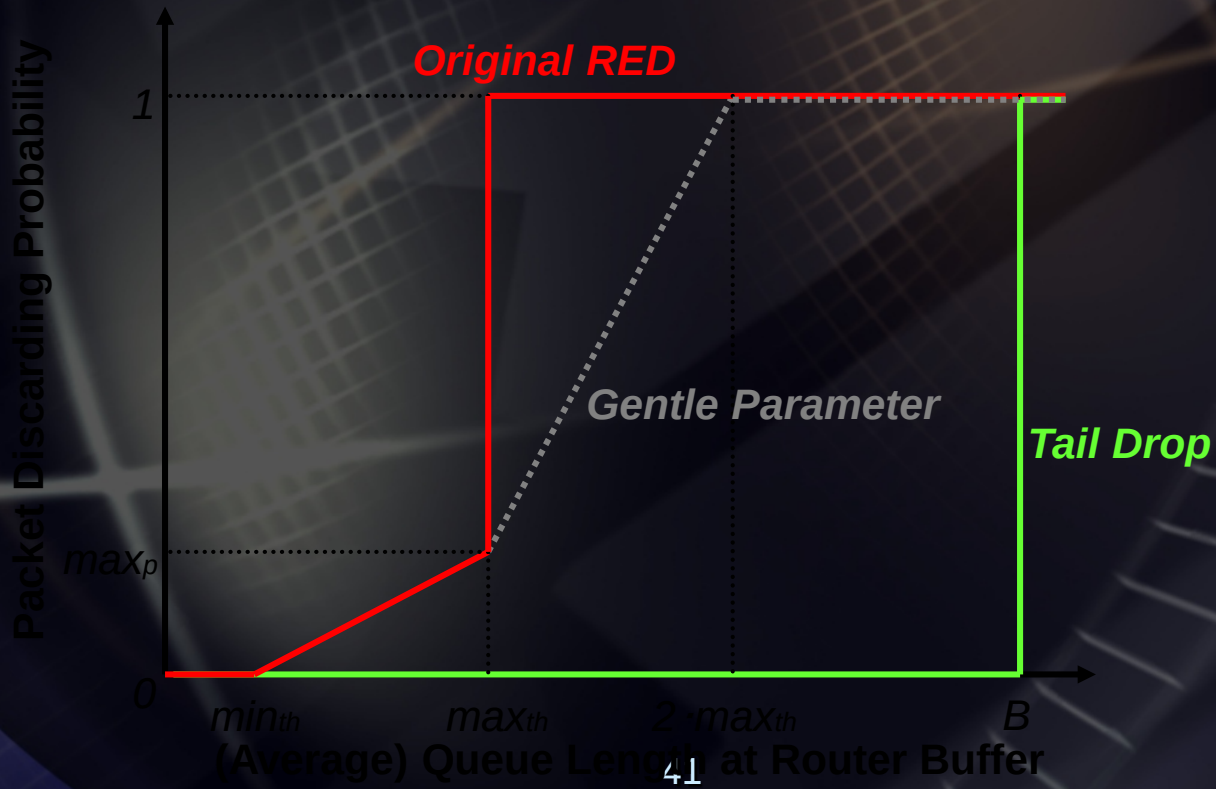
- Tail-Drop Router
  - 単純なFIFOバッファで、実装が容易
  - バッファが一杯になると到着パケットを廃棄
  - 問題点
    - バースト的なパケットロス
    - TCP のタイムアウトを引き起こしやすい
  - 公平性は全く期待できない

AQM (Active Queue Management)

- TCPはネットワークをブラックボックスと考えている
  - 異環境の下での高い公平性は期待できない
- ルータバッファで積極的な制御を行う
  - パケットの確率的な廃棄
  - 複数キューを用いたフロー管理

RED (Random Early Detection)

- バッファが一杯になる前に、パケットを確率的に廃棄
- 廃棄率はキュー長によって変動



RED (Random Early Detection)

- TCPのスループット、公平性向上
  - バースト的なパケット廃棄が減少
  - TCPのタイムアウトが減少
- 問題点
  - パラメータ設定の難しさ
  - TCPの本質的な不公平性は解消されない
- 非常に多くの改善方式が提案されている

REDの改善方式

- FRED (Flow Random Early Detection)
  - ルータ内パケットをフロー毎にカウント
- SRED (Stabilized RED)
  - 確率的な動作をコネクション数予測に利用
- CSFQ (Core-Stateless Fair Queuing)
  - エッジルータとコアルータで協力
- DRED, ARED, BLUE, GREEN,…
- 制御の複雑さと公平性はトレードオフ
  - どれを使うかは適用箇所の要求条件による

ECN (Explicit Congestion Notification)

- ルータバッファでのパケット廃棄を避ける
  - 輻輳時にルータでパケットを廃棄するのではなく、マーキングを行う
  - 送信側TCPはマークされていればパケットロスと同様に扱う
- TCPの不公平性が解消されるわけではない
  - 送信側の動きは定義されていない
  - すべてのルータ、エンドホストが対応する必要がある

複数キューを用いた制御

- 古くから提案されている
  - RR (Round Robin), FQ (Fair Queuing)
- 高い公平性
- 問題点も多い
  - 制御の重さ
  - フロー衝突
- DRR (Deficit Round Robin)
  - O(1)でできる
  - パケットサイズの違いも考慮
- やはり、制御の度合いと公平性はトレードオフ

エンドホストでの資源管理

- TCPの送受信バッファ
  - 固定値が割り当てられている
  - バッファサイズで最大レートは制限
  - 小さすぎるとネットワークに余裕があるときに帯域を使い切れない
  - 大きすぎるとネットワークが混雑しているときにバッファが無駄
- 特に繁忙なWWWサーバ等で無駄が生じやすい

エンドホストでの資源管理

- 割り当て資源量を動的に制御する
  - ATBT
    - バッファサイズをウインドウサイズに応じて増減
  - SSBT
    - バッファサイズを推定TCPスループットに応じて増減
  - 公平に、かつ必要なところへ必要なだけ資源を配分
- 資源の浪費の回避、サーバ能力の向上

まとめ

- TCPは既にインターネットに広く普及しているため、全く新しいトランスポート層プロトコルを提案することは非現実的
- 既存のTCP との親和性、及び提案方式を段階的に適用した場合の影響を考慮すべき

まとめ

- フロー間の公平性を向上させるためには、エンドホストにおける輻輳制御だけではなく、ネットワークルータにおける何らかの制御が必要
- ルータにおいて公平性を向上させる機構を持たせる際には、公平性の定義とその程度、収容するフロー数等によって、適切な複雑さを持つアルゴリズムを選択して適用すべき