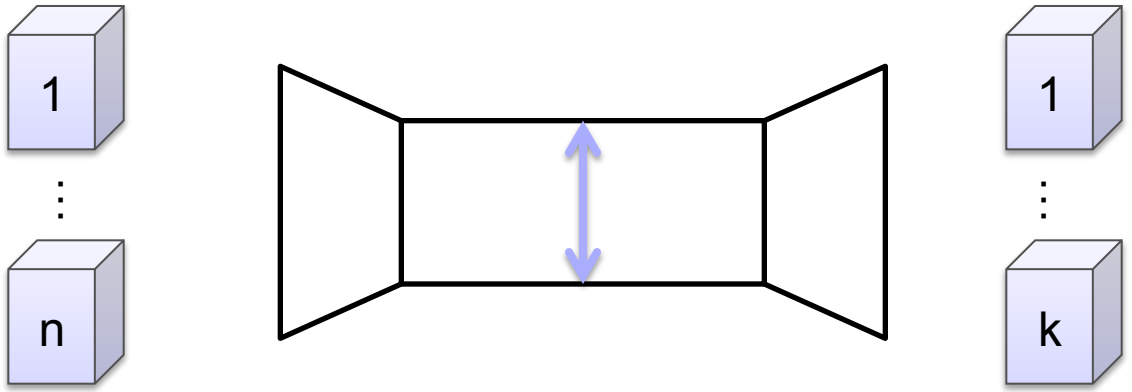
情報ネットワーク論I
TCP 2/2

門林　雄基
奈良先端科学技術大学院大学

トランスポートプロトコルの挑戦

- 未知のパラメータが多い
  - 行われている通信の数
  - 帯域幅のボトルネック
  - エラー率
- 我々はどうやってネットワークを壊さずにできるだけ多くの通信させるか？



Key ideas: 調査, 判断, 自己警備, 大局的な安定

ネットワーク資源の競合と調停

- フロー制御 (flow control)
  - 速度の違いを吸収
  - 配送順序の入れ替わりからの回復
  - 重複、パケット廃棄、ビット誤りからの回復

- 輻輳制御 (congestion control)
  - 輻輳を抑えつつ帯域を共有
  - 帯域の公平分配

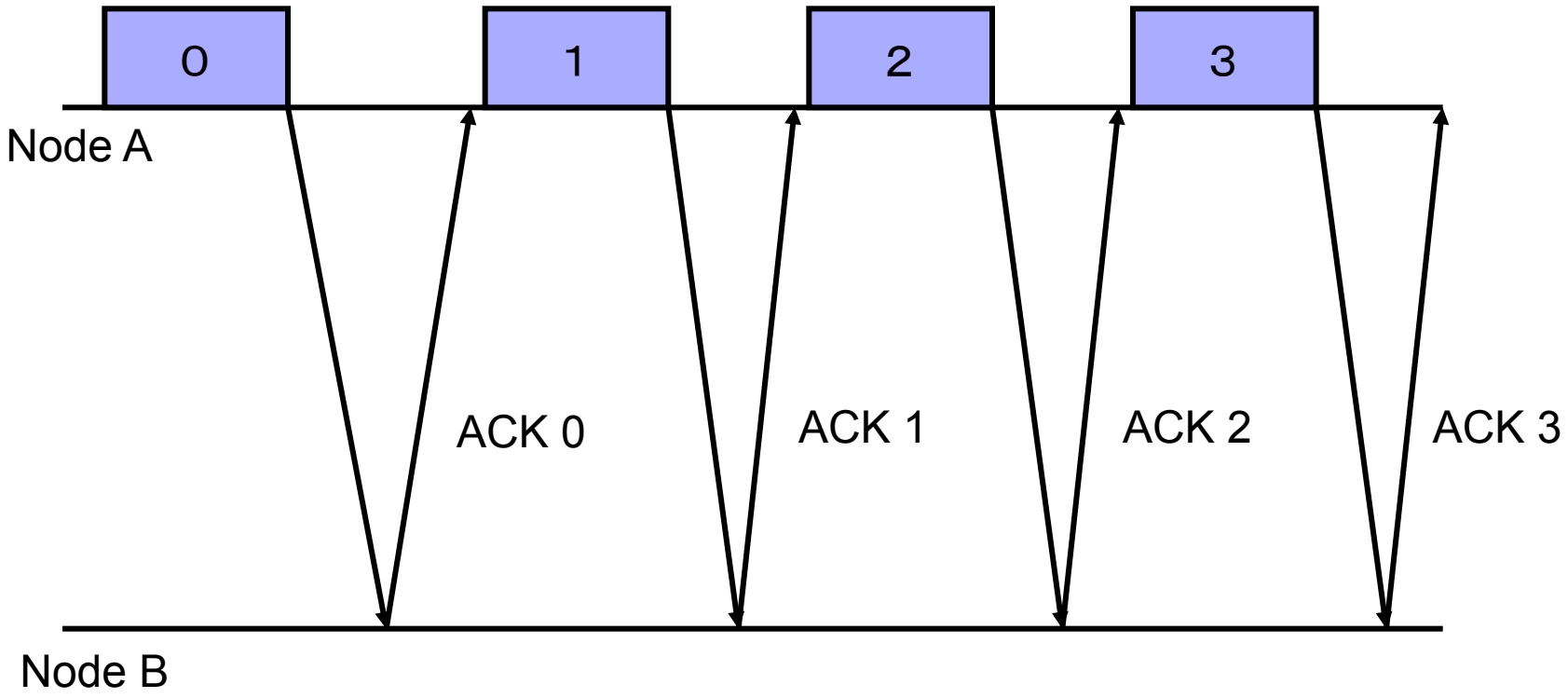
フロー制御

- Stop-and-Wait
- Go Back n

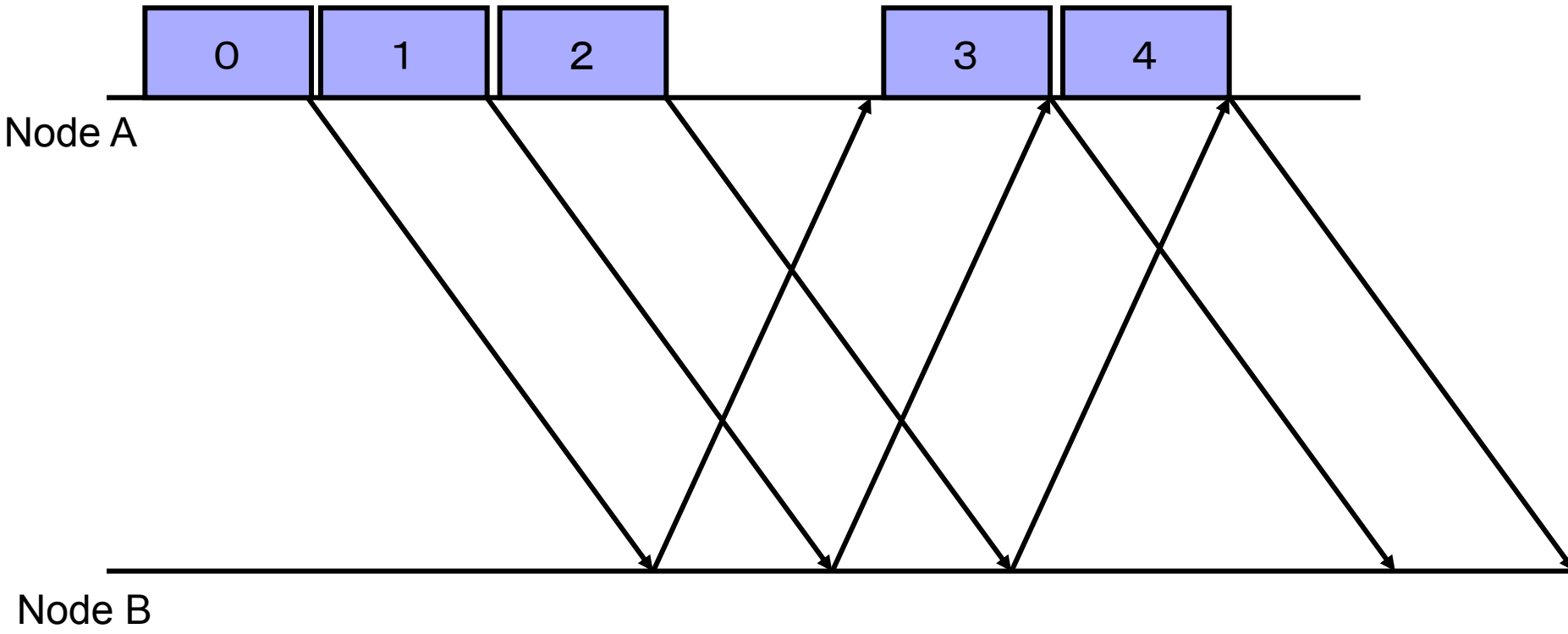
- さまざまな方式が存在
  - ネットワーク側が速度等を申告するもの
  - ホスト側で速度を推定しながら動くもの

  - ARQ (Adaptive Repeat reQuest)

Stop-and-Wait



Go Back n



TCP におけるフロー制御の特徴

- End to end
  - グローバルな資源割り当てはしない
  - 各ホストが利用可能帯域を推定しながら動作
  - ルータは介在しない – パケット廃棄のみ

- Scalable
  - ホスト側のみで自律的に動く方式はネットワーク側での状態管理を必要としないため、より大規模な運用が可能
    - Autonomous → Less state management → Scalable

Questions?

TCPから学ぶべき点

- 非常に単純なアルゴリズム
  - マクロな自己安定性
- Greedyなノードの存在を仮定していない
  - グローバルな制御システムの排除
- データリンクを選ばず、そこそこ動く
  - 特定の環境で最適な性能を発揮するわけではない
- 30年にわたる段階的改善

TCPにおけるフロー制御

- 利用帯域を調整
  - Sliding window
- パケット番号ではなくシーケンス番号で
  - Window size
- パケット送出間隔を調整
  - ACK clocking

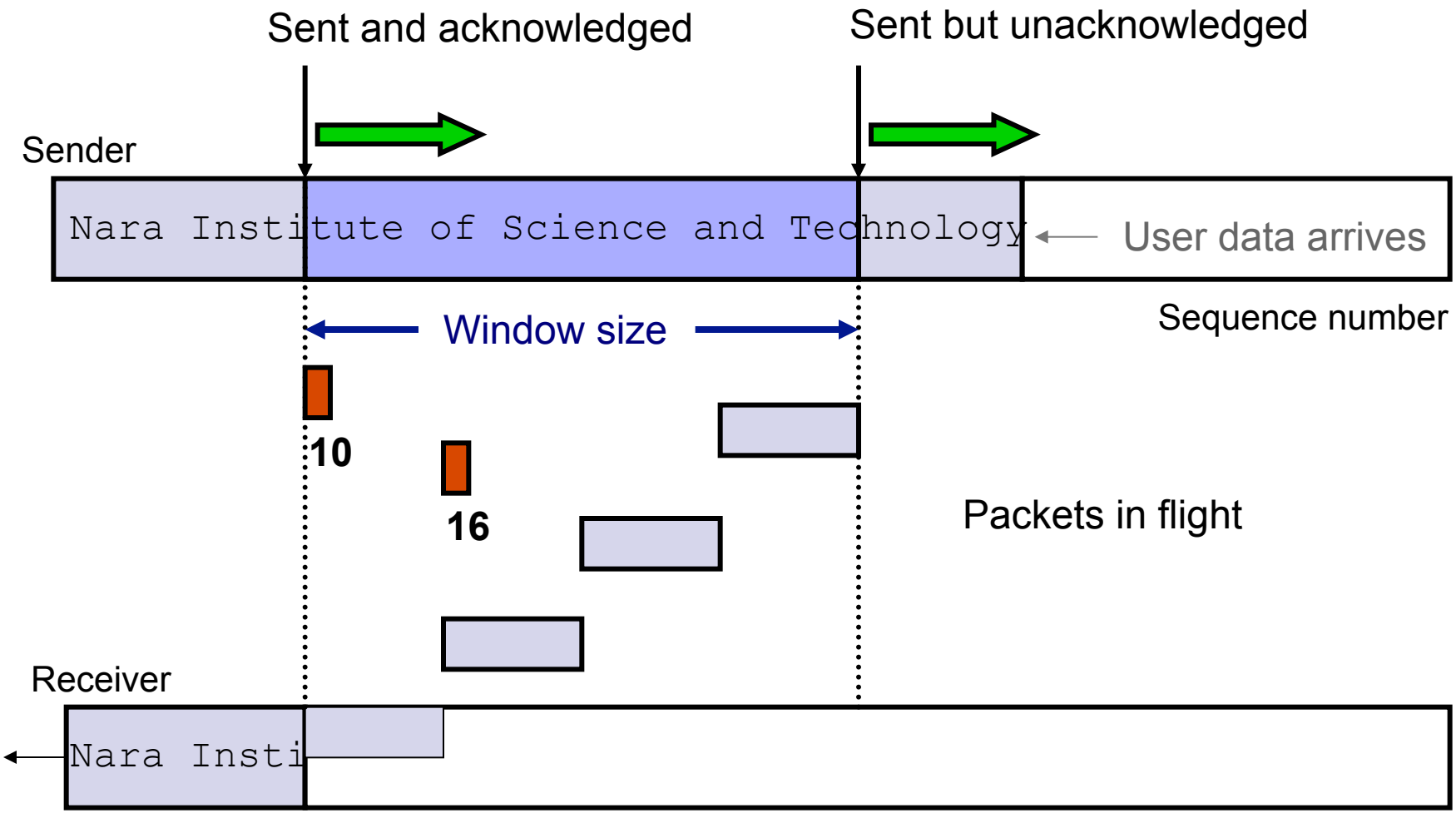
- その他
  - 誤り検出 – TCP checksum
  - パケット廃棄 – duplicate ACK, timeout

パケット送出間隔の調整 (ACK clocking)

- namによるデモ

- ACK clocking のデモ
- Duplicate ACK のデモ

Sliding window



Questions?

TCP における輻輳制御

- Fair-share model: 帯域の公平分配
  - End to end
- window size の増減
  - additive increase
  - multiplicative decrease
    - 自己安定的であることが知られている (R. Jain, et al., “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks”, 1989)
- window size 増加戦略の切り替え
- パケット廃棄で輻輳が起きたことを知る

Fair-share: 帯域の公平分配

- namによるデモ

Window size の増やし方

- slow start threshold (ssthresh) を基準としてcongestion window (cwnd) の増やし方を変える

(アルゴリズムの概要)

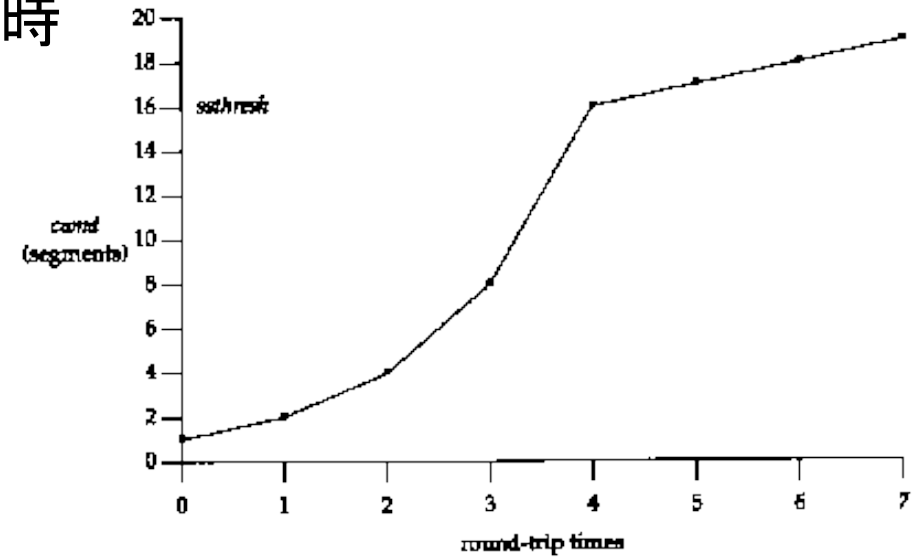
- On receiving an ACK:

```
if (cwnd < ssthresh) {
        /* slow start: exponential increase */
        cwnd += 1;
} else {
        /* congestion avoidance: additive increase */
        cwnd += 1 / cwnd;
}
```

Window size の増やし方

- Slow startの時
  - exponential increase
- Congestion avoidanceの時
  - additive increase
- (Illustrated p. 311)

Source: TCP/IP Illustrated, Vol.1



- 効用: V. Jacobson, “Congestion Avoidance and Control”, SIGCOMM’88.

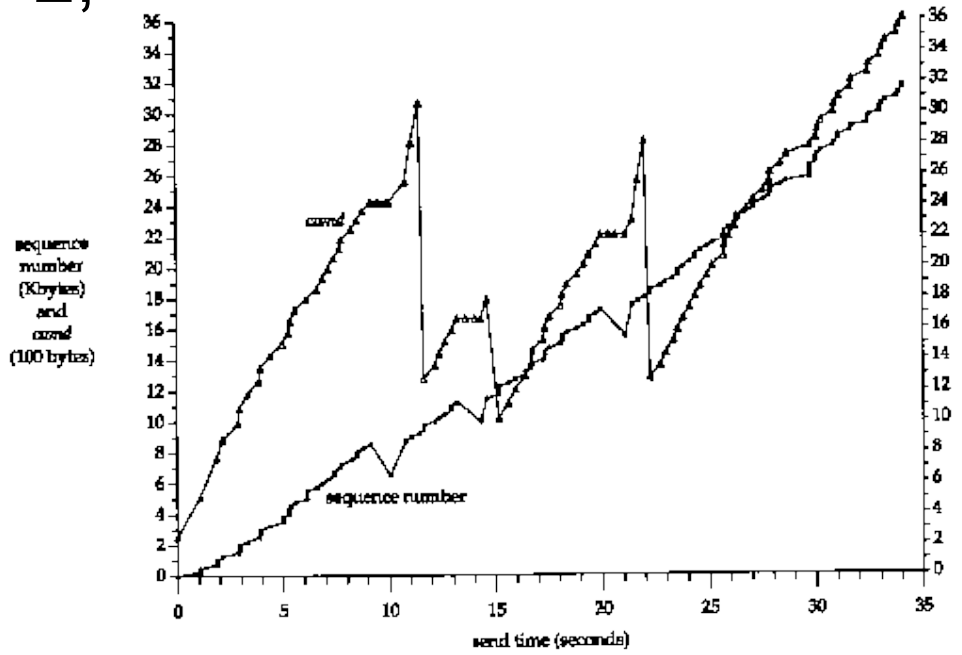
Window size の減らし方

- (アルゴリズムの概要)
- On detecting packet drop:

```
ssthresh = cwnd / 2;
if (timeout) {
        cwnd = 1;
}
```

- (Illustrated p. 315)

Source: TCP/IP Illustrated, Vol.1



Questions?

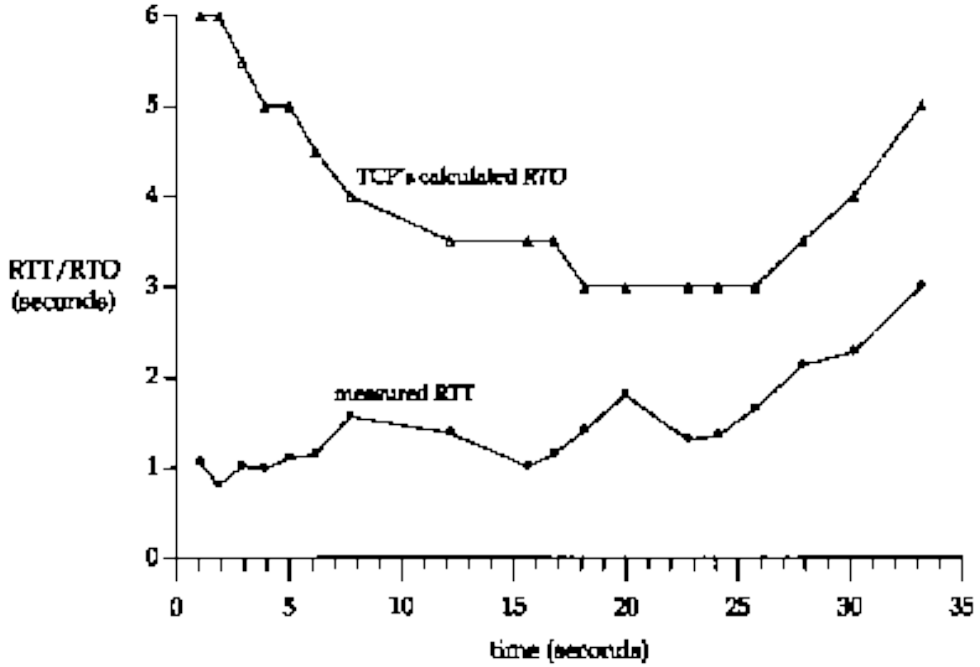
Packet drop

- 2つの方法で検出:
  - Duplicate ACK
  - Retransmission Time Out (RTO)

- タイムアウトはどのように判定すればよいか？
  → RTO ~ RTT Estimator

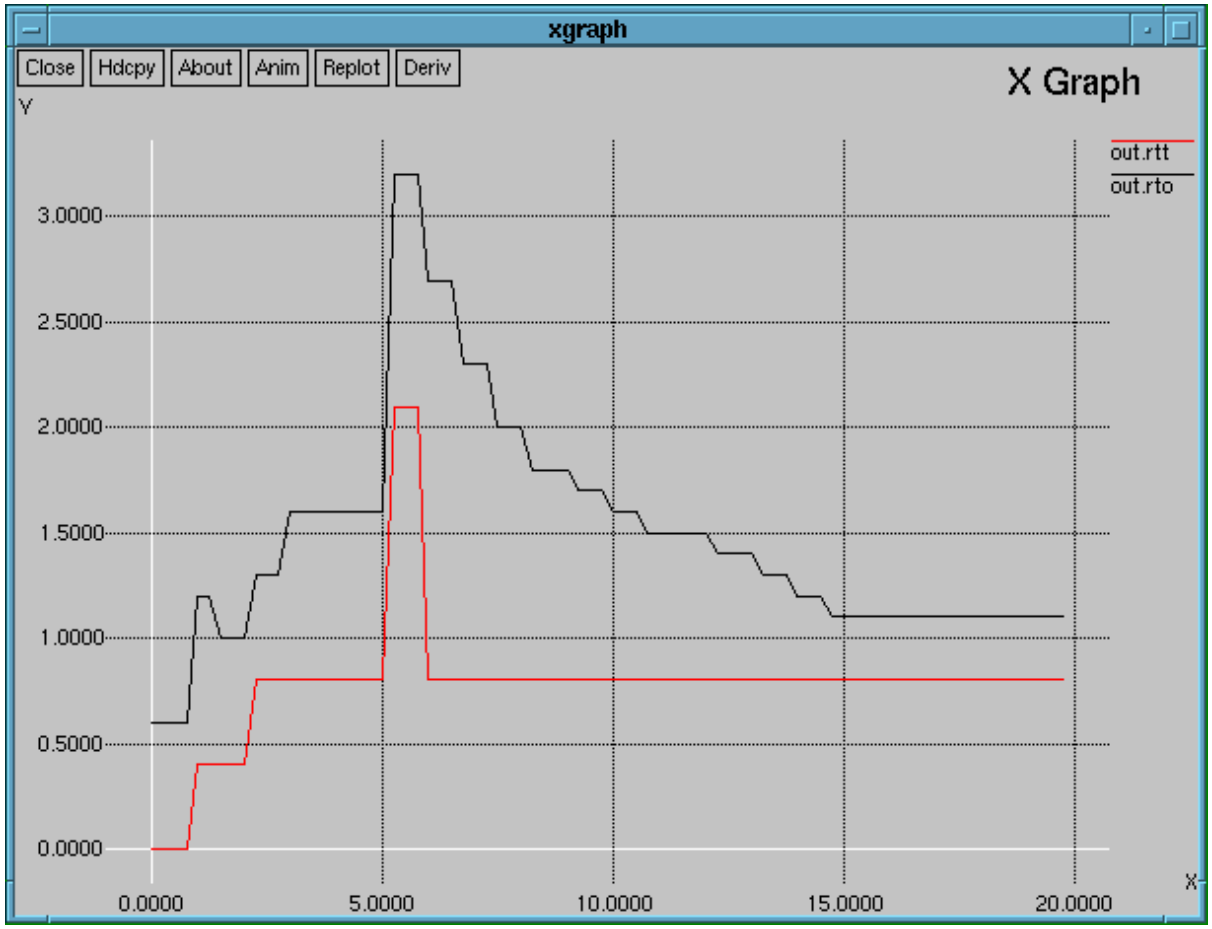
RTO Calculation

Source: TCP/IP Illustrated, Vol.1



- Err = M – A
  A <- A + gErr
  D <- D + h(|Err| - D)
  RTO = A + 4D
  - A: smoothed RTT
  - D: smoothed mean deviation
  - g: gain for the average (1/8)
  - h: gain for the deviation (1/4)
- (Illustrated p. 304)
- (Illustrated p. 305)

RTO calculation



Source: A Quick Tour Around TCP,
http://web.eecs.utk.edu/~dunigan/tcptour/

Loss sensitivity

- namによるデモ

- Timeout
- Congestion avoidance

More topics on TCP

- Fast retransmit
- Fast recovery (RFC3782)
- Selective Acknowledgment (SACK) (RFC3517)
- TCP Friendly Rate Control (TFRC) (RFC3448)
- Explicit Congestion Notification (ECN) (RFC3168)

- Interaction with RED
- TCP extensions for wireless links
- ...

TCP enhancements

- 細かな実装の違いが存在
  - Reno
  - NewReno
  - SACK
  - ...
- 拡張には2種類ある
  - 「特定の条件で最適な性能を発揮する」ような TCP拡張
    - → たくさん論文が出続けているが無視してよい
  - 「幅広い条件でそれなりの性能を向上させる」ようなTCP拡張
    - → IETFで標準化を検討
- ベンダ独自の工夫が裏目に出ることも

Questions?

まとめ

- フロー制御
  - Stop-and-Wait
  - Go back n
- TCPにおけるフロー制御
  - Sliding window
- TCPにおける輻輳制御
  - Slow start
  - Congestion avoidance

課題

- TAが用意する仮想マシンイメージを用いて遅いWebサイトを取り上げ、tcpdumpまたはwiresharkを用いて遅い理由を分析せよ。

- 提出期限 5/23（水） 17:00
- 提出先 A3F インターネット工学研究室