

情報ネットワーク論I

TCP 2/2

門林 雄基
(インターネット工学講座)
TA: 檉原、Omar,

Copyright(C)2003 Yusuke Kadibayashi. All rights reserved.

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

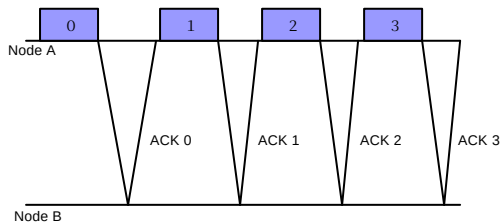
Copyright(C)2003 Yusuke Kadibayashi. All rights reserved.

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)

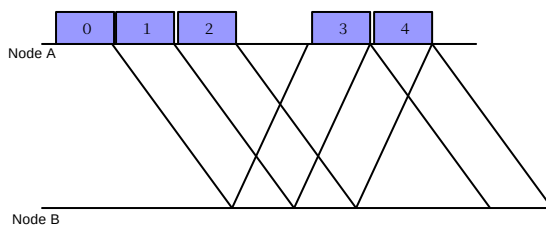
Copyright(C)2003 Yusuke Kadibayashi. All rights reserved.

Stop-and-Wait



Copyright(C)2003 Yuuki Kadibayashi. All rights reserved.

Go Back n



Copyright(C)2003 Yuuki Kadibayashi. All rights reserved.

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ではない
 - 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

Copyright(C)2003 Yuuki Kadibayashi. All rights reserved.

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
- 20年にわたるチューニング

Copyright(C)2003 Yuuki Kadibayashi. All rights reserved.

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

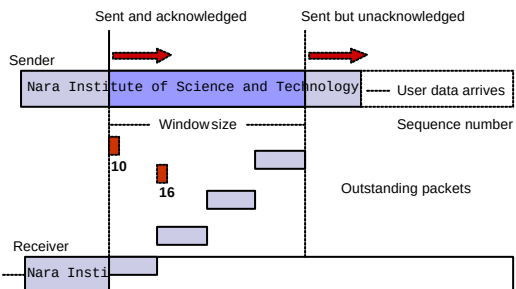
Copyright(C)2003 Yuuki Kadibayashi. All rights reserved.

パケット送出間隔の調整 (ACK clocking)

- namによるデモ

Copyright(C)2003 Yuuki Kadibayashi. All rights reserved.

Sliding window



Copyright(C)2003 Yoski Kadibayashi. All rights reserved.

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 (Jain, et al.)
- 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

Copyright(C)2003 Yoski Kadibayashi. All rights reserved.

Fair-share: 帯域の公平分配

- namによるデモ

Copyright(C)2003 Yoski Kadibayashi. All rights reserved.

Window size の増やし方

- slow start threshold (sssthresh) 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 < sssthresh) {
 - /* slow start */*
 - cwnd += 1;
 - } else {
 - /* congestion avoidance */*
 - cwnd += 1 / cwnd;

Copyright(C)2003 Youshi Kadibayashi. All rights reserved.

Window size の増やし方

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

Copyright(C)2003 Youshi Kadibayashi. All rights reserved.

Window size の減らし方

- (アルゴリズムの概要)
- On detecting packet drop:
 - sssthresh = cwnd / 2;
 - if (timeout) {
 - cwnd = 1;
- (Illustrated p. 315)

Copyright(C)2003 Youshi Kadibayashi. All rights reserved.

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

Copyright(C)2003 Yusuki Kadibayashi. All rights reserved.

RTO Calculation

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

Copyright(C)2003 Yusuki Kadibayashi. All rights reserved.

Loss sensitivity

- namによるデモ

Copyright(C)2003 Yusuki Kadibayashi. All rights reserved.

More topics on TCP

- Fast retransmit
- Fast recovery
- Selective Acknowledgment
- Explicit Congestion Notification
- Interaction with RED
- TCP extensions for wireless links
- ...

Copyright(C)2003 Yuuki Kadobayashi. All rights reserved.

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

Copyright(C)2003 Yuuki Kadobayashi. All rights reserved.

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

Copyright(C)2003 Yuuki Kadobayashi. All rights reserved.

課題

- 1回目のレポートで挙げた自らの疑問について、自分で調べて答えよ。
 - 「これまでの授業の中で、疑問に思ったこと、あるいは納得できなかったことを2つ挙げよ。」 → 答えよ。
- あるいは、下記課題のいずれかをこなしてもよい
 - 1. TCPの輻輳制御アルゴリズムが自己安定的である様子を、自らが書いた簡潔なプログラムによって示せ。
 - 2. TCPの輻輳制御アルゴリズムが帯域の公平分配を行う様子を、自らが書いた簡潔なプログラムで示せ。
 - いずれも、実装言語、グラフ形式は自由)

Copyright(C)2003 Yusuki Kadobayashi. All rights reserved.

1回目レポート課題について

- 課題
 - 「多くのアプリケーションは実際にはライブラリを利用して作成しているが、多くのライブラリはOSI 7階層参照モデルでは独立した階層ではない。どのような場合に、ライブラリが階層化プロトコルにおける階層となりうるか論じよ。」
- 以下の2条件が成立する場合、階層となりうる
 - 上位層に対するサービスが規定されている場合
 - 同一層における相手との通信を規定している場合

Copyright(C)2003 Yusuki Kadobayashi. All rights reserved.

1回目レポート課題について

- 実装のレイヤは関係ない

