ソフトウェアによる精密ペーシング方式を用いたTCP通信性能の改善

高野了成[1,2]，工藤知宏[1]，児玉祐悦[1]，
松田元彦[1]，岡崎史裕[1]，石川裕[3,1]

[1)]産業技術総合研究所，グリッド研究センター
[2)]株式会社アックス　[3)]東京大学

2006年1月27日　電子情報通信学会
ネットワークシステム研究会＠大阪工業大学

Grid Technology Research Center AIST

AIST

発表の流れ

- 背景
  - 高速長距離ネットワークにおけるTCPの問題
  - 既存のペーシング実装の問題
- ギャップパケットを用いたペーシングの実現
- PSPacerの実装方法
- 評価
- まとめ

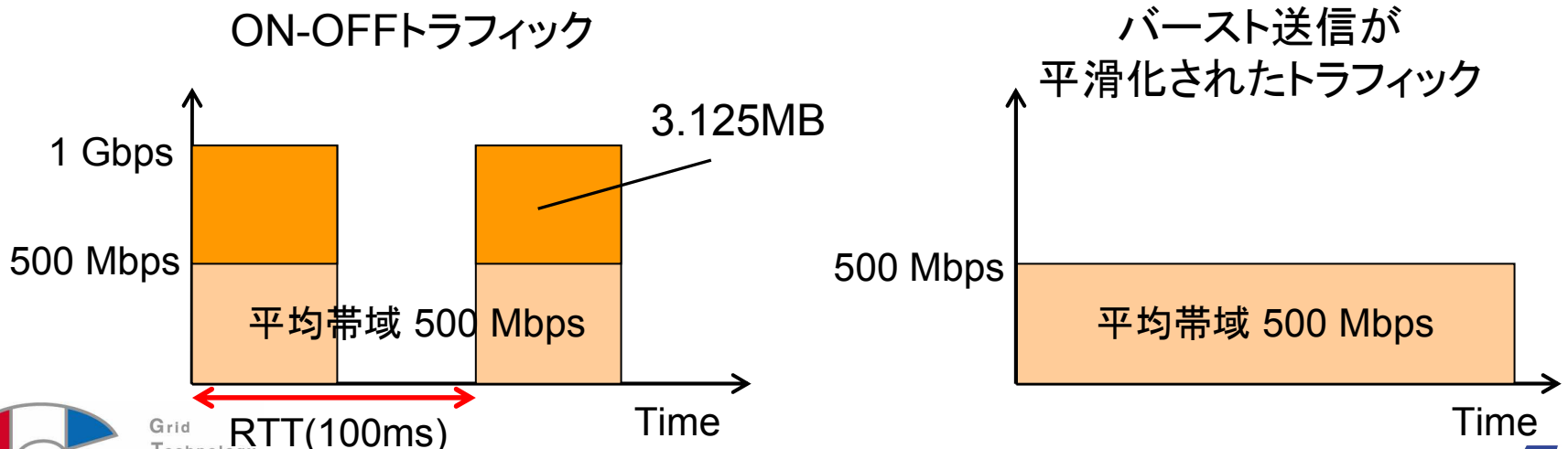
TCPのフロー制御

- ウィンドウ制御
  - RTT(Round Trip Time)内に送出するパケット量の決定
  - 輻輳ウィンドウ(cwnd)がネットワークの帯域遅延積(BDP)に一致することが理想

- バースト制御
  - パケット送出タイミングの決定
  - ACK受信をパケット送信のトリガとすることで，RTT内で均等にパケットを送出する（ACKクロッキング）

ACKクロッキングとバースト送信

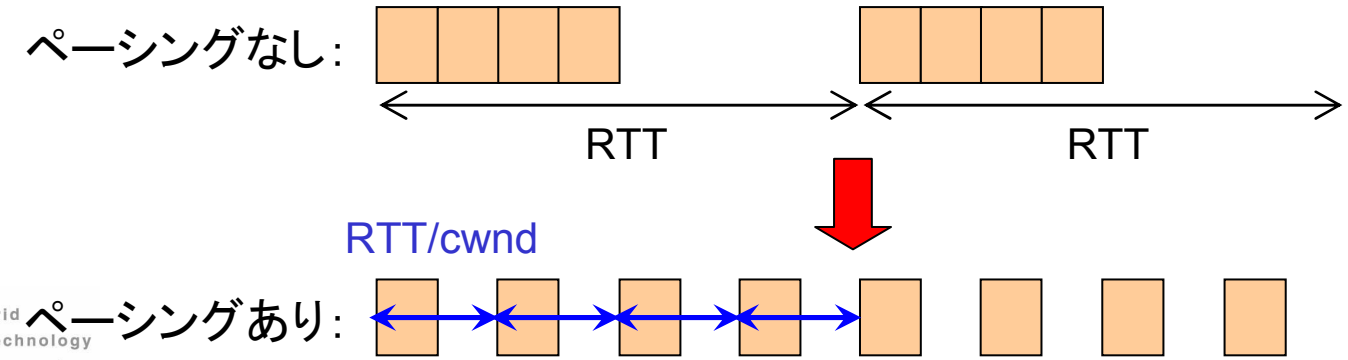
- ACKクロッキングによりバースト送信を平滑化する
  - 入出力帯域の差分が，ボトルネックルータのバッファサイズを超えると，パケットが破棄される
- 単純で効果的な方法だが，常に有効とは限らない
  - 例えば，スロースタート，ACK圧縮



（GbE，ボトルネック帯域 500Mbps，RTT 100msの場合）

ペーシング

- ACKクロッキングの代わりに，目標帯域に基づいてパケット送信間隔(IPG)を調整する
  - TCPでは，RTT/cwndごとにパケット送信すればよい
- 精密なIPG制御には高精度タイマが必要
  - GbEの場合，1500バイトのパケット送信に12us要する
  - タイマ割込みの負荷増大により，実現は困難

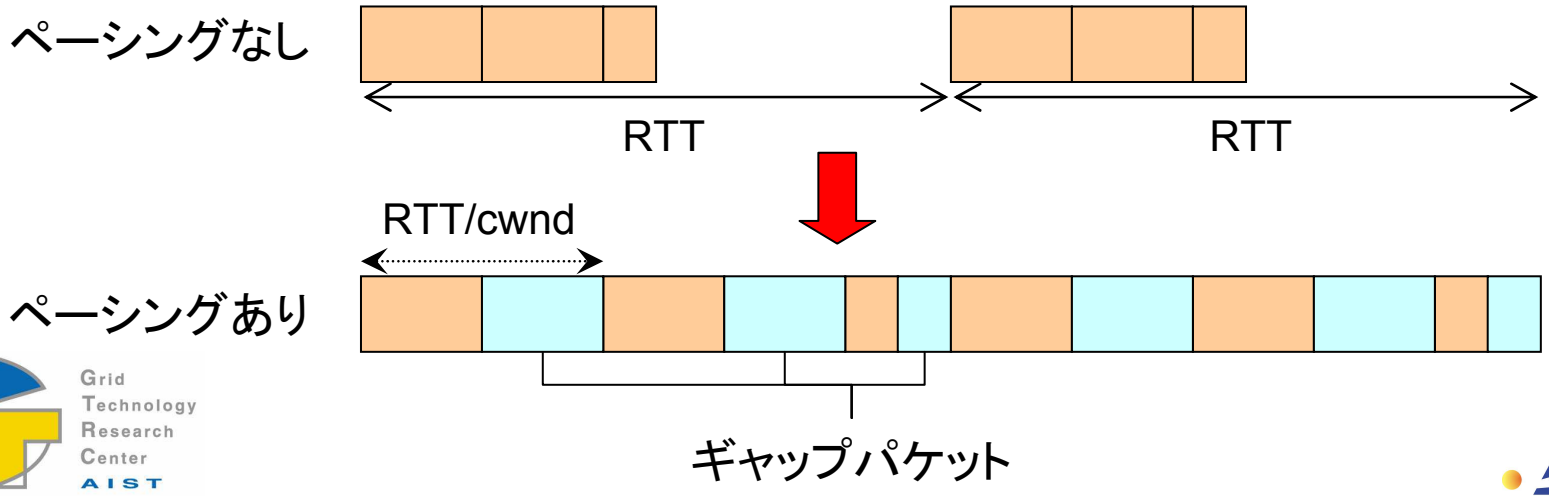


発表の流れ

- 背景
- ギャップパケットを用いたペーシングの実現
- PSPacerの実装方法
- 評価
- まとめ

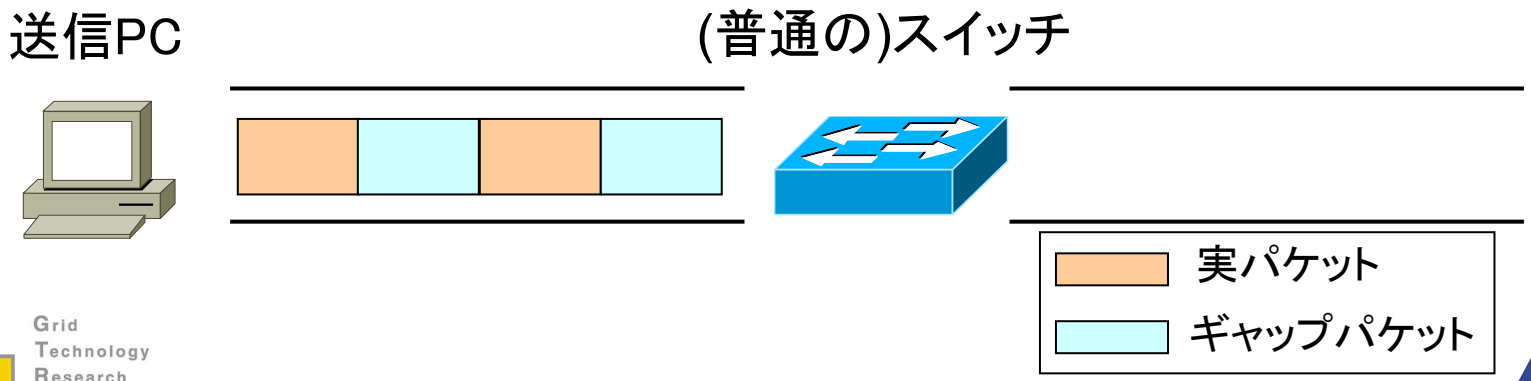
ギャップパケット（1）

- タイマ割込みを利用しない，ソフトウェアによる精密なパケットスケジューリングを実現したい
- 実パケット間にギャップパケット（ダミーパケット）を挿入することでIPGを調整する
  - 物理的なパケット送信時間は正確なので，パケットサイズを変更することで，IPGを精密に制御可能



ギャップパケット(2)

- ギャップパケットは実際に送信PCから送信する必要がある
- PAUSEフレーム(IEEE 802.3x)を利用する
  - 直近のスイッチ/ルータの入力ポートで破棄されるので，外部ネットワークへの影響はない



ギャップパケットサイズの計算

- 基本アルゴリズム
  - 物理帯域に占める目標帯域の割合を基にIPGを調整

$$ipg = \left(\frac{max\_rate}{target\_rate} - 1 \right) \times packet\_size$$

  - IPGがMTUより大きい場合は，複数のギャップパケットを挿入する
  - 例えば，
    - max_rate = 1 Gbps
    - target_rate = 500 Mbps
    - packet_size = 1500 Byte

$$ipg = \left(\frac{1000}{500} - 1 \right) \times 1500 = 1500$$

目標帯域の見積り

- 静的ペーシング
  - 静的に固定値を設定する
  - ボトルネック帯域が既知で，一定の場合に有効
- 動的ペーシング
  - ウィンドウ制御の情報（cwnd，RTT）を基に目標帯域を計算し，ギャップパケットサイズを決定する
  - ボトルネック帯域が不明の場合に有効

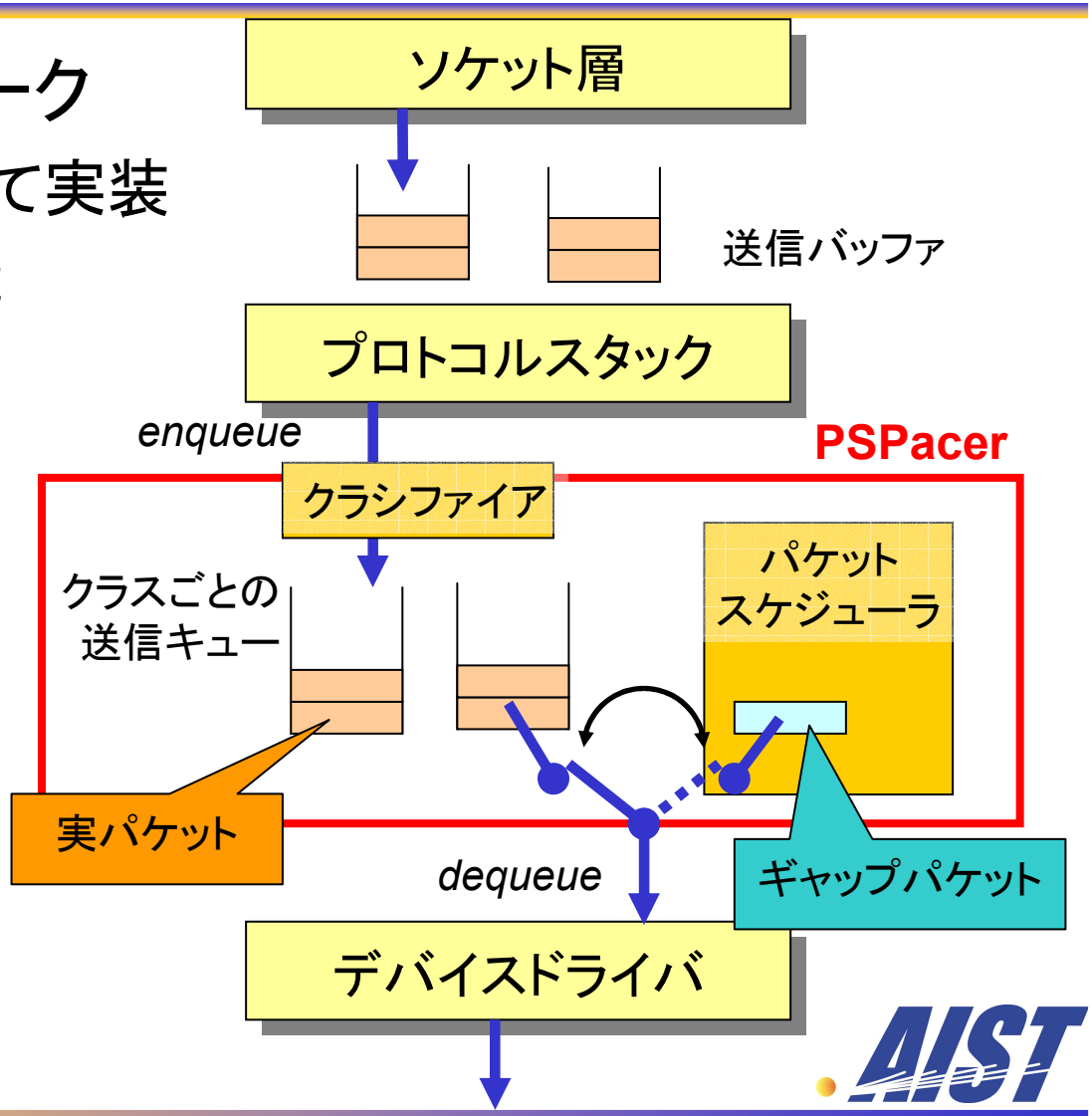
$$\text{target\_rate} = \frac{\text{packet\_size} \times \text{cwnd}}{\text{RTT}}$$

発表の流れ

- 背景
- ギャップパケットを用いたペーシングの実現
- PSPacerの実装方法
- 評価
- まとめ

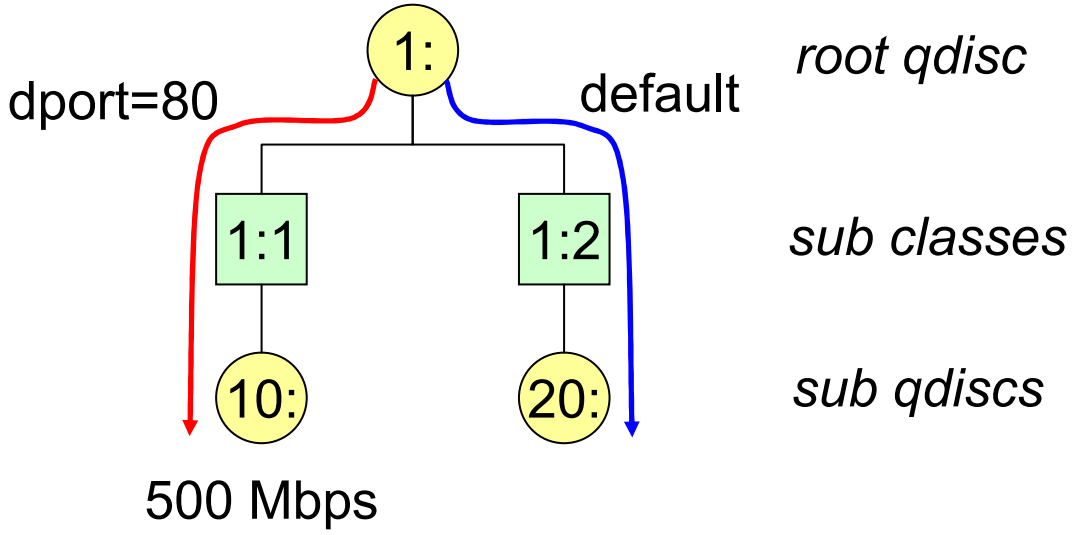
PSPacerの実装

- Iproute2フレームワーク
  - Qdiscモジュールとして実装
  - tcコマンドで設定可能
- カーネル再構築不要
- ドライバ非依存
- プロトコル非依存



PSPacerの使用例

```
# tc qdisc add dev eth0 root handle 1: psp default 2
# tc class add dev eth0 parent 1: classid 1:1 psp rate 500mbit
# tc class add dev eth0 parent 1: classid 1:2 psp mode 0
# tc qdisc add dev eth0 parent 1:1 handle 10: pfifo
# tc qdisc add dev eth0 parent 1:2 handle 20: pfifo
# tc filter add dev eth0 parent 1: protocol ip pref 1 u32 /
   match ip dport 80 0xffff classid 1:1
```



発表の流れ

- 背景
- ギャップパケットを用いたペーシングの実現
- PSPacerの実装方法
- 評価
  - ギャップパケットによる帯域制御
  - 高遅延環境でのTCP通信におけるペーシングの効果
- まとめ

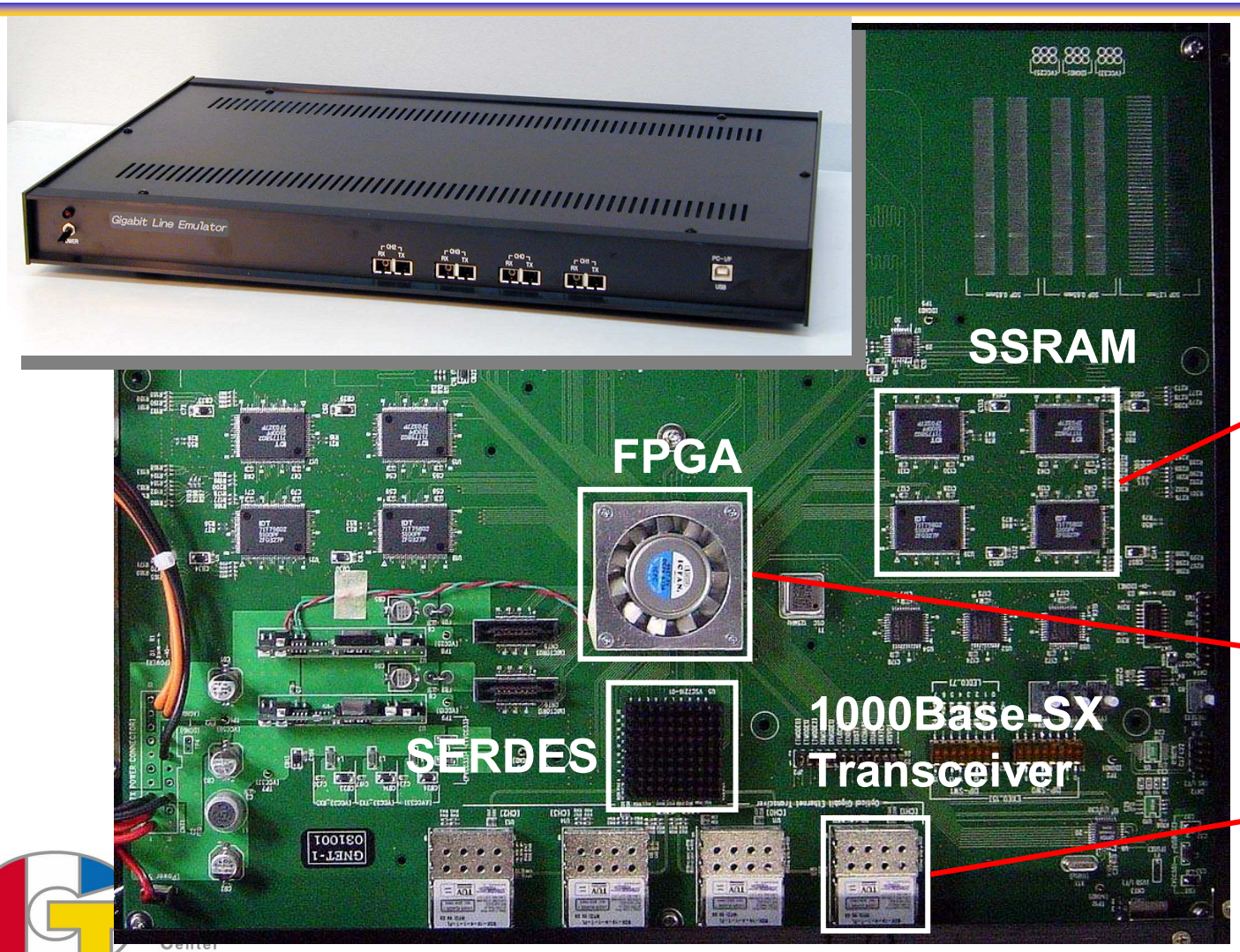
実験環境(1)

目標帯域を変えながら，1対1通信の帯域をGtrcNET-1で計測

PC環境
- CPU: Intel Xeon/2.4GHz dual
- Memory: 2GB (DDR266)
- PCI Bus: PCI-X 133MHz/64bit
- NIC: Intel PRO/1000 (82545EM)
- OS: Fedora Core 3
  Linux 2.6.14.2 + Web100 2.5.6
- BIC TCP
- Socket Buffer: 12.5MB
- txqueuelen: 10000

GtrcNET-1
帯域測定

GtrcNET-1:プログラマブル ギガビットネットワークテストベッド



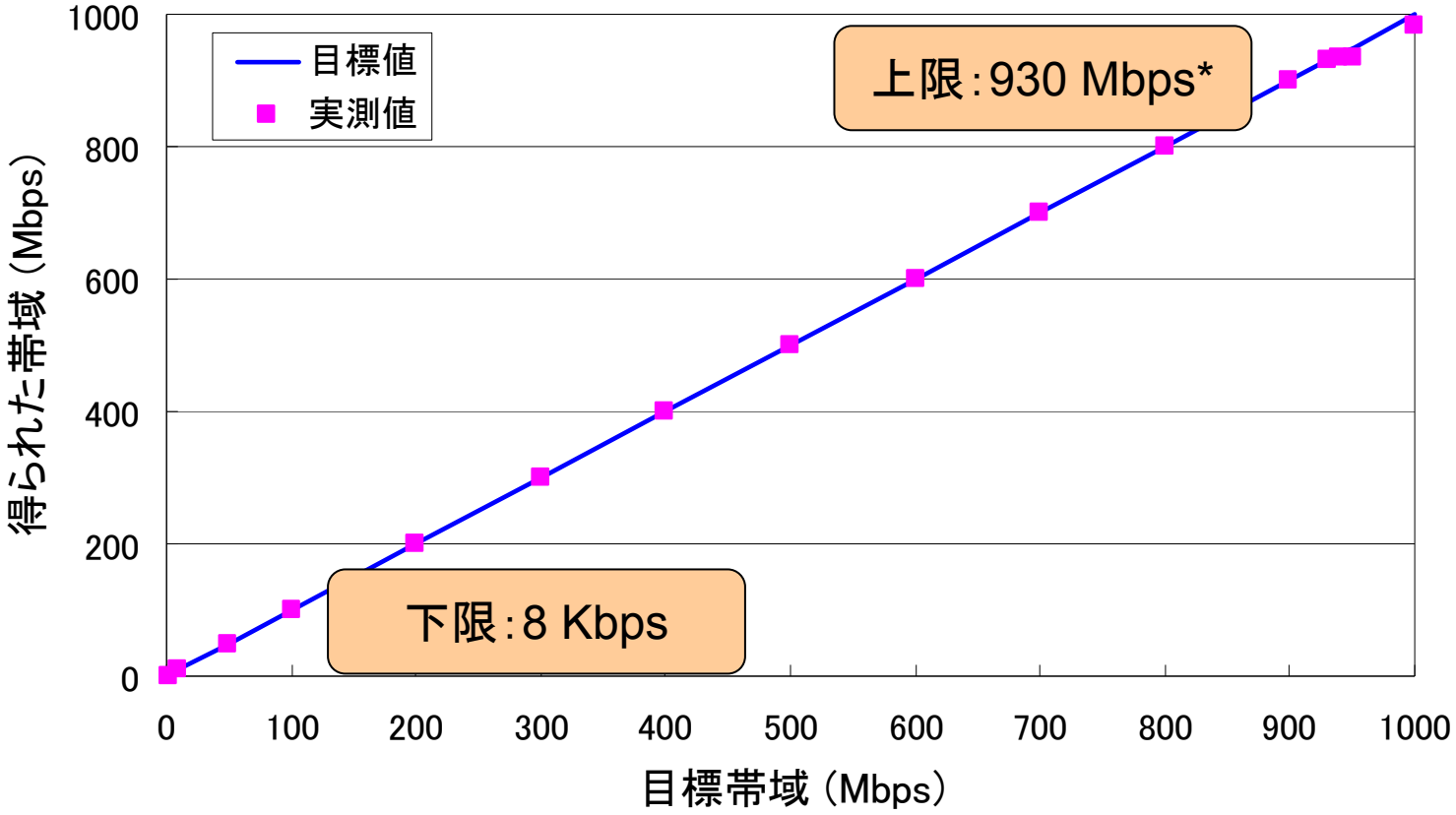
1 Gbps read and write simultaneously 144Mbits/port

Xilinx XC2V6000 (76K logic cells)

4 GbE ports

ギャップパケットによる帯域制御

8Kbps～930Mbpsでの帯域制御に対応



*) ギャップパケットが最小フレームサイズである64バイトの場合
（評価には静的ペーシングを用いた）

実験環境(2)

GtrcNET-1 で高遅延環境をエミュレートし，帯域とパケットロスを計測

PC環境
- CPU: Intel Xeon/2.4GHz dual
- Memory: 2GB (DDR266)
- PCI Bus: PCI-X 133MHz/64bit
- NIC: Intel PRO/1000 (82545EM)
- OS: Fedora Core 3
  Linux 2.6.14.2 + Web100 2.5.6
- BIC TCP
- Socket Buffer: 12.5MB
- txqueuelen: 10000

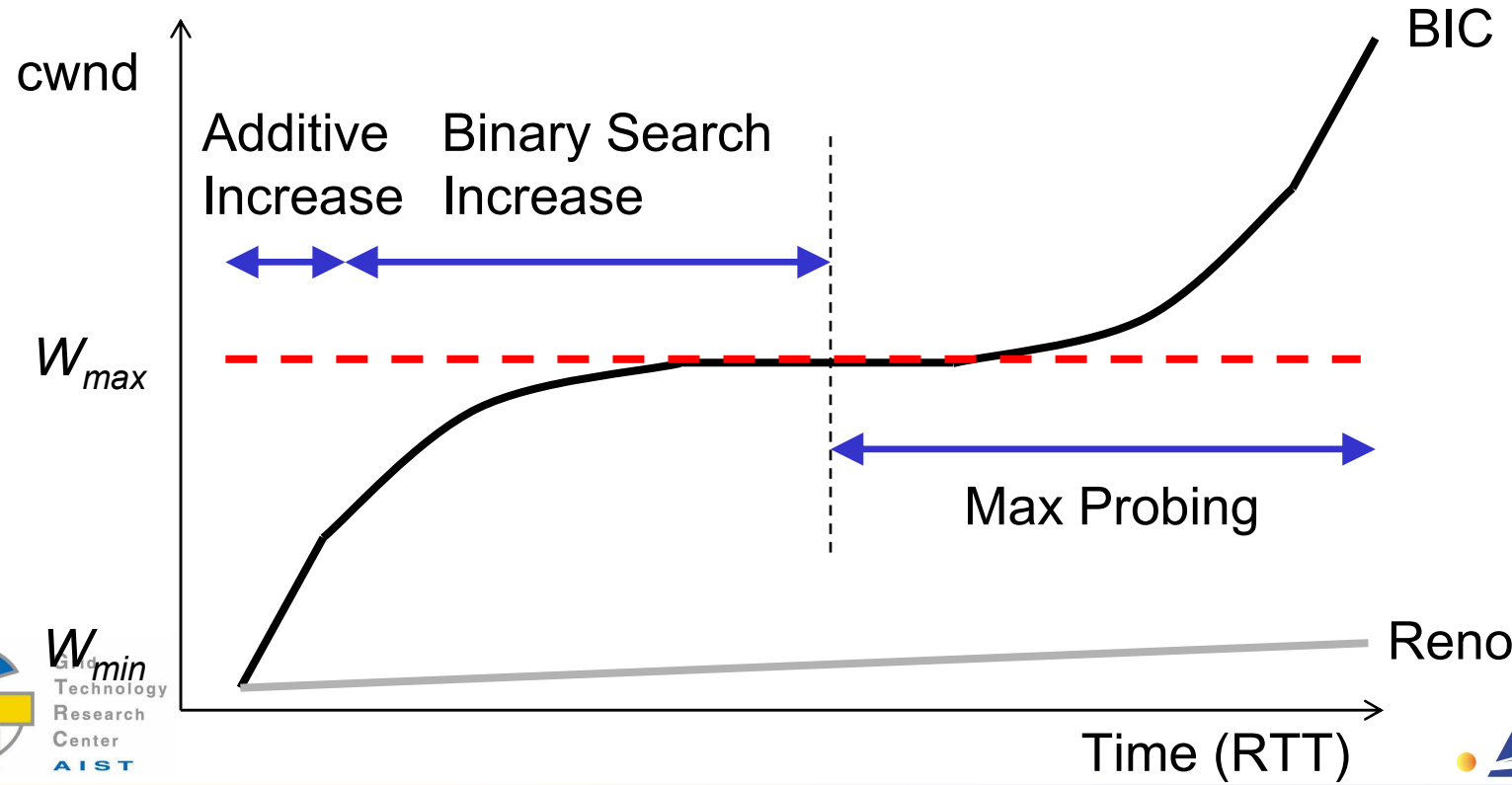
GtrcNET-1
往復遅延: 100ms
帯域: 500Mbps
Drop Tail ルータ (FIFO 1MB)
帯域測定 (500us)

Center
AIST

AIST

（参考）BIC TCP

- 帯域のスケーラビリティとTCP-Friendlinessを両立
  - Additive Increase/Binary Search Increaseモード
  - パケットロス時のcwndをW_{max}に設定する



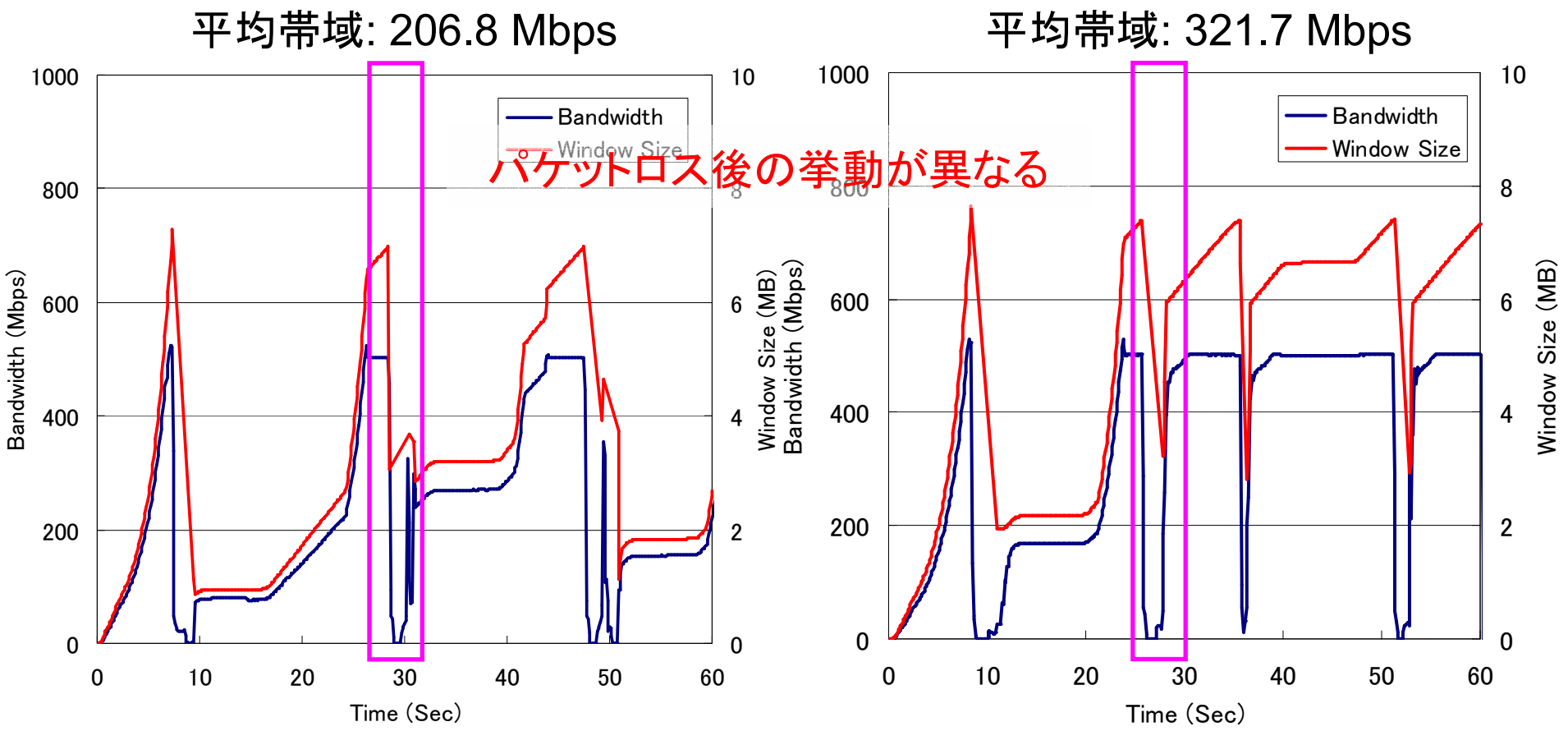
高遅延環境でのTCP通信における ペーシングの効果

（帯域は1分間の平均値）

	平均帯域 (Mbps)		パケットロス数
	Iperf (TCP)	GrtcNET-1 (Ethernet)	
noPSP	200	**206.8**	**1455**
PSPD	315	**321.7**	**285**
PSPS	433	**447.7**	**0**

- noPSP: ペーシングなし
- PSPD: 動的ペーシング
- PSPS: 静的ペーシング (BW_{max}=500Mbps)

平均帯域と輻輳ウィンドウサイズ（1）



(a) noPSP

(b) PSPD

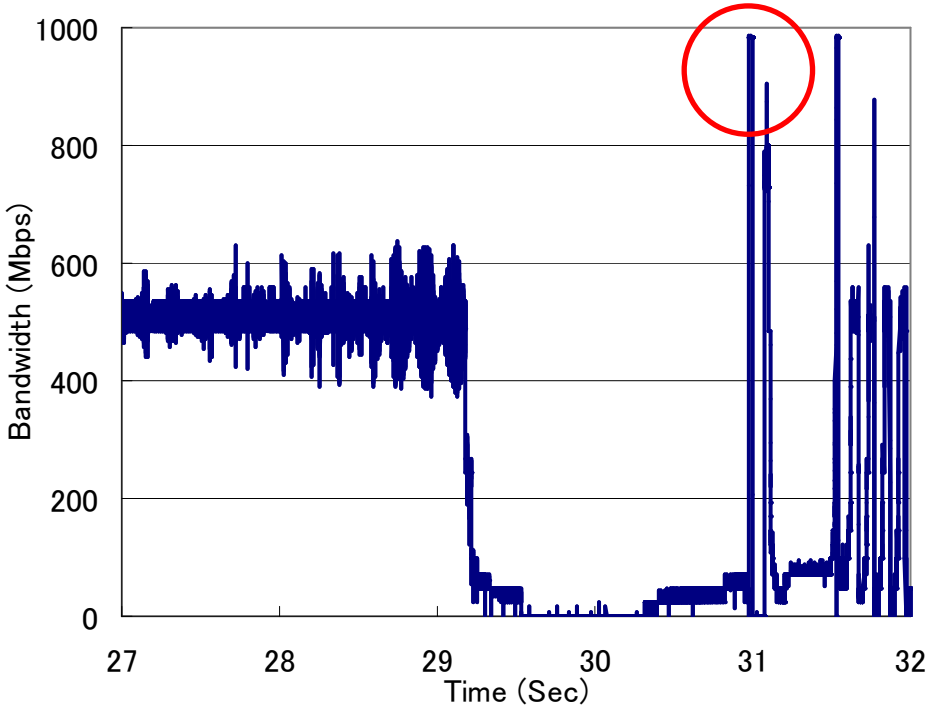
（帯域は100msの平均値，cwndは100ms間隔で取得）

Grid Technology Research Center AIST

AIST

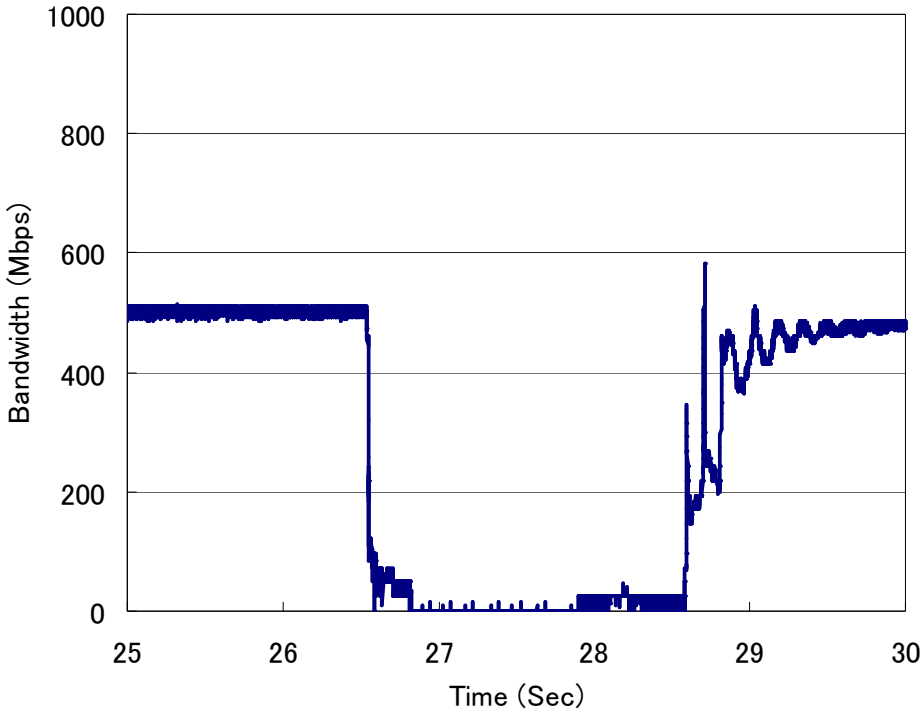
パケットロス直後の挙動

スロースタート後にパケットロスが発生し，W_{max}が縮退するため，cwndの回復に時間がかかる



(a) noPSP

スロースタート後にパケットロスが発生しない



(b) PSPD

（帯域は500us間隔で取得）

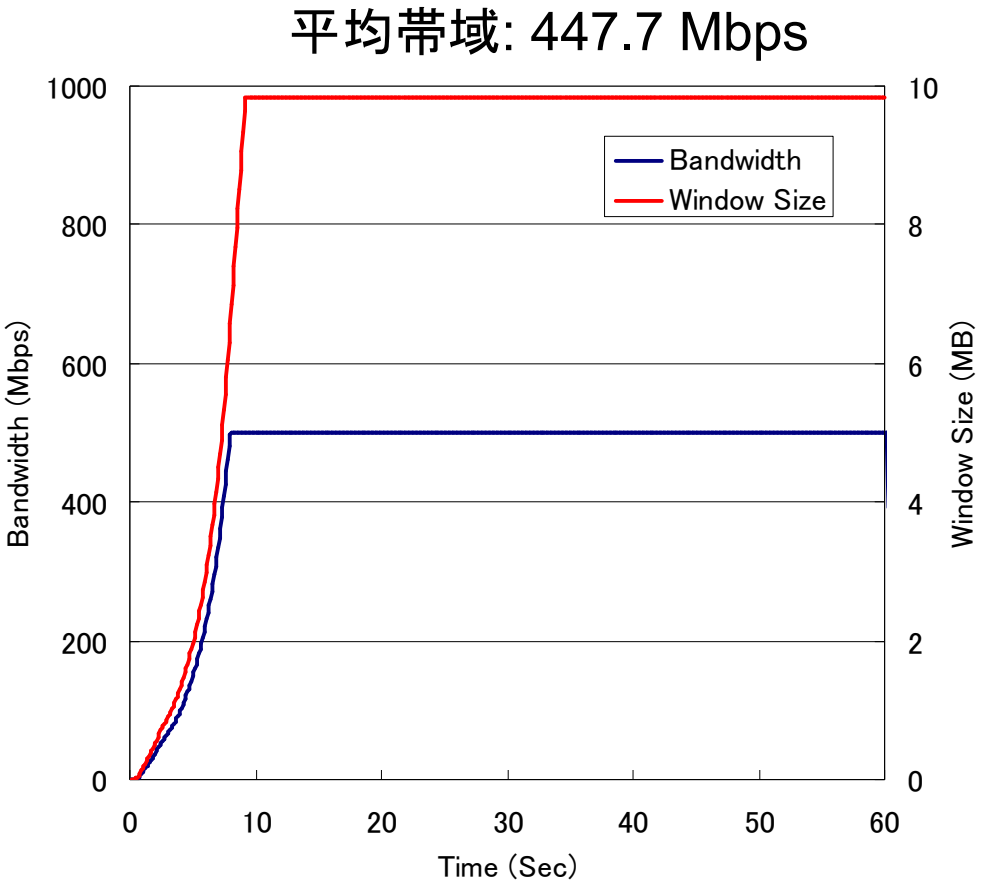
Grid Technology Research Center AIST

AIST

平均帯域と輻輳ウィンドウサイズ（2）

送信帯域がボトルネック帯域を超えないので，パケットロスが発生しない

cwndが約10MBまで上昇しているのは，送信PCのインタフェースキュー遅延が増加したため



(c) PSPS

（帯域は100msの平均値，cwndは100ms間隔で取得）

まとめ

- パケット送信間隔を精密にスケジューリングできるギャップパケットを提案した
  - イーサネット上での実現にPAUSEフレームを利用する
- ギャップパケットを用いてソフトウェアによる精密なペーシング方式を実現した
- ペーシングを適用した結果，高遅延環境において，TCP通信性能が向上することを示した
  - スロースタート時のパケットロスを削減できた

PSPacerはGNU GPLライセンスにて公開

GridMPI: http://www.gridmpi.org/

なお，本研究の一部は文部科学省「経済活性化のための重点技術開発プロジェクト」の一環として実施している超高速コンピュータ網形成プロジェクト（NAREGI: National Research Grid Initiative）による．

Grid Technology Research Center AIST

AIST

今後の課題

- BIC TCP以外の輻輳制御アルゴリズムを用いた評価
  - FAST TCP
- ペーシングによる同期ロス問題の検証
  - [Aggarwal00]