ネットワークモジュールの自動配布による通信の高速化

光来健一*　千葉滋**　益田隆司*

*東京大学大学院　理学系研究科　情報科学専攻
**筑波大学　電子・情報工学系

適応的なOS

- 適応的(Adaptable)とは？
  - 実行環境に応じた機能の拡張
  - 実行環境に合わせた性能のチューニング
- システム例
  - 拡張可能OS[Bershad et al.95]
  - loadable kernel module(LKM)
    - カーネルに新しい機能を追加できる
  - Plug&Play
    - 新しいハードウェア用のドライバが自動的に組み込まれる

実行環境

・ネットワーク構成
・アプリケーションの種類
など

適応的な分散OS

- 分散透明な適応
  - 各マシンで個別に動いているOSを統一的に適応させる
    - 分散システム全体にまたがった最適化が可能
  - ユーザには単一システムとして見せる
    - 管理コストの削減も可能
- 適応的なネットワークシステム
  - 実行環境に応じた専用プロトコルを使用する
  - 実行環境に合わせて実装パラメータを調節する
    - ネットワーク通信を高速化する

適応的なネットワークシステム

- ネットワークモジュールの自動配布
  - ネットワークモジュールとは？
    - ネットワークプロトコルを実装したOSのプログラム
    - ターゲットはネットワーク層、トランスポート層
  - 通信元から相手先のOSに自動的に配布し、組み込む
    - OSが最適なプロトコルを判断する
    - アプリケーションが専用プロトコルをOSに指示する
  - あらかじめモジュールを取ってきておく必要がない
    - ユーザに意識させない
    - 使用するプロトコルを実行時に柔軟に変えられる

適応的なネットワークシステムの課題

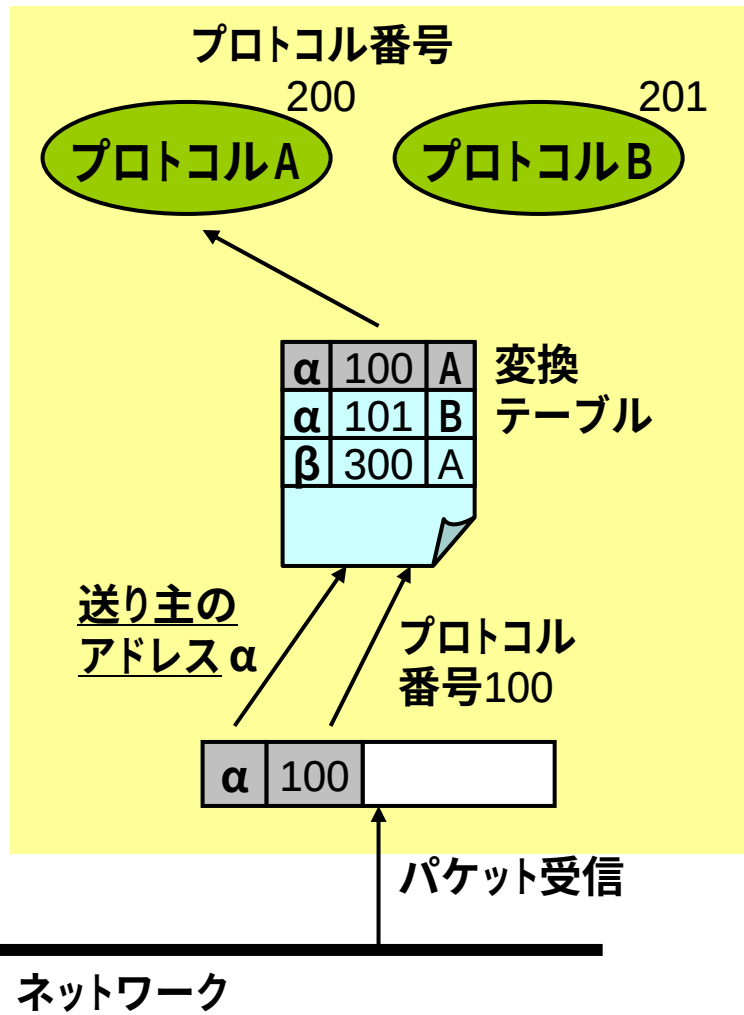
- プロトコル番号の解決
  - 新しいネットワークプロトコルを使う際に生じるプロトコル番号の割当て問題
- 障害対策
  - ネットワークモジュールを配布されたマシンがリブートした時にどうするか、など
- 安全性、異機種分散への対応
  - ネットワークモジュールの形式

プロトコル番号の動的解決

- プロトコル番号とは？
  - パケットの受信時にプロトコルを識別できるように一意に割り当てられる番号
  - パケットヘッダに格納される
- プロトコル番号は通信相手との間で動的に解決する
  - 標準でないプロトコルに固定の番号を割り当てるのは難しい
  - サーバが集中管理するのは妥当ではない

基本システム

- プロトコル番号はマシン毎に独自に割り当てる
  - マシンとプロトコル番号の組でプロトコルが同定できる
- 変換テーブルを使用してプロトコルを判別する
  - 変換テーブル
    - 入力：ネットワークアドレス
プロトコル番号
    - 出力：プロトコルの受信ルーチン



動的にプロトコル番号を解決するためのメタプロトコル

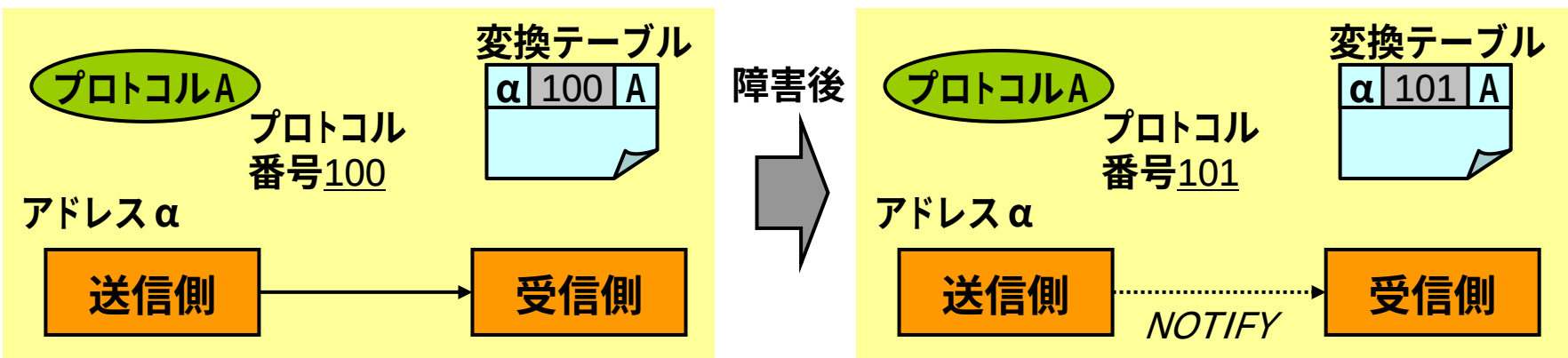
- どうやって変換テーブルを作成するか？
  - Dynamic Protocol Number Agreement Protocol (DPNAP) を使う
    - *NOTIFY*メッセージ
      - プロトコル情報を通知する
    - *REQUEST*メッセージ
      - プロトコル情報を要求する
    - *REPLY*メッセージ
      - *REQUEST*メッセージへの返事としてプロトコル情報を送り返す

DPNAPのアルゴリズム

- 送信側は通信に先立って相手に*NOTIFY*メッセージを送る
  - その通信相手にそのプロトコルを使ってパケットを送るのが最初の時だけ
  - 送信側のネットワークアドレス、プロトコル番号、プロトコルの名前･バージョンから成る

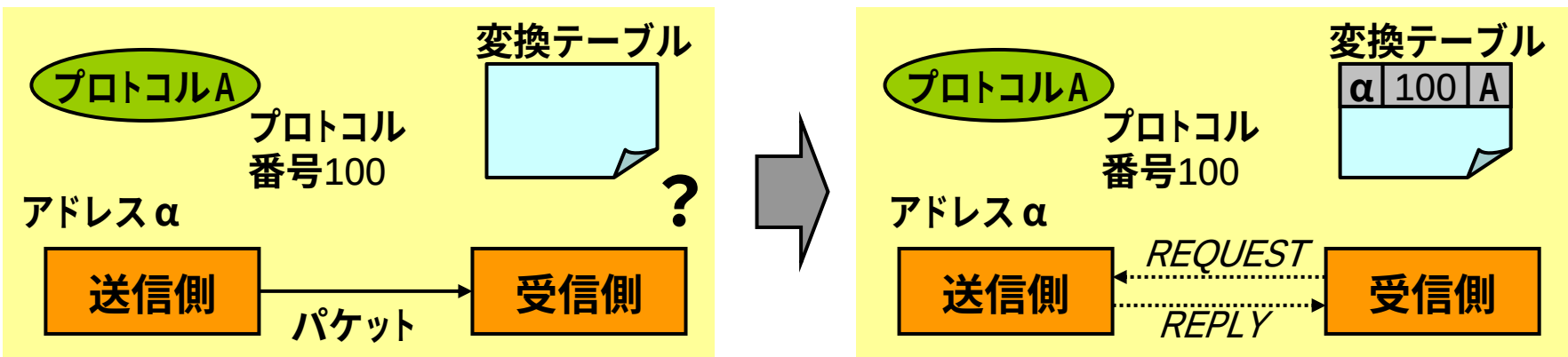
送信側の障害への対応

- 送信側でのプロトコル番号の割り当て方が変わるかもしれない
  - 通信相手の変換テーブルとの矛盾が生じる
- 最初にパケットを送る時は再び*NOTIFY*メッセージを通信相手に送る
  - 受信側の変換テーブルの古いエントリは上書きされる



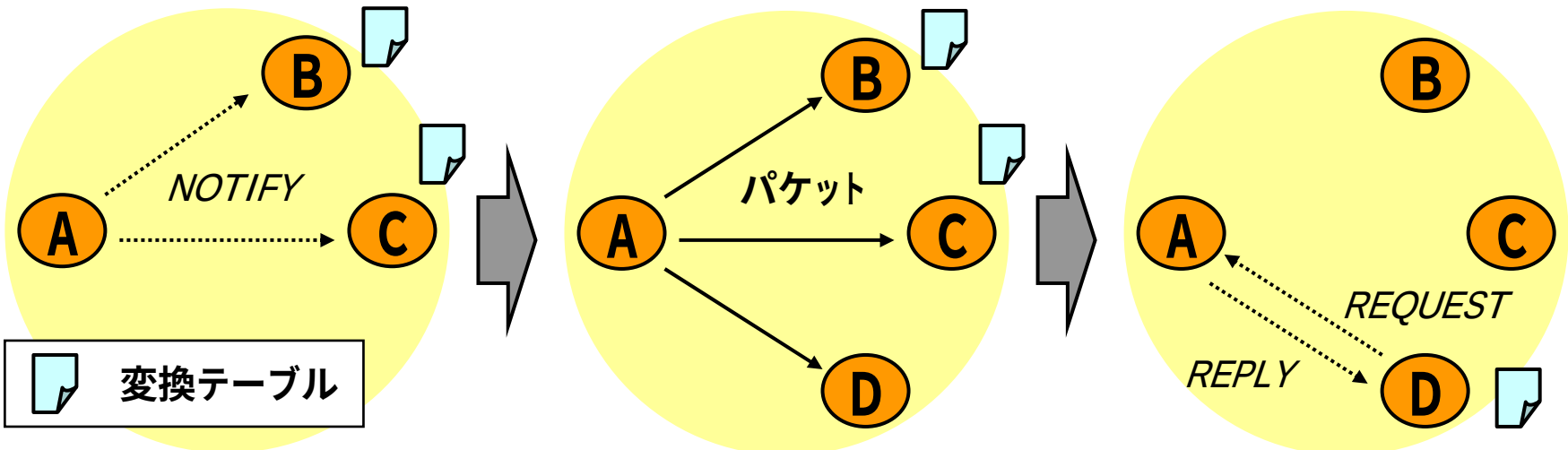
受信側の障害への対応

- 変換テーブルにエントリの存在しないパケットが送られてくるかもしれない
  - 変換テーブルの内容が失われるため
- 未知のパケットを受け取った時は*REQUEST*メッセージを送信側に送る
  - *REPLY*メッセージから変換テーブルのエントリを作成



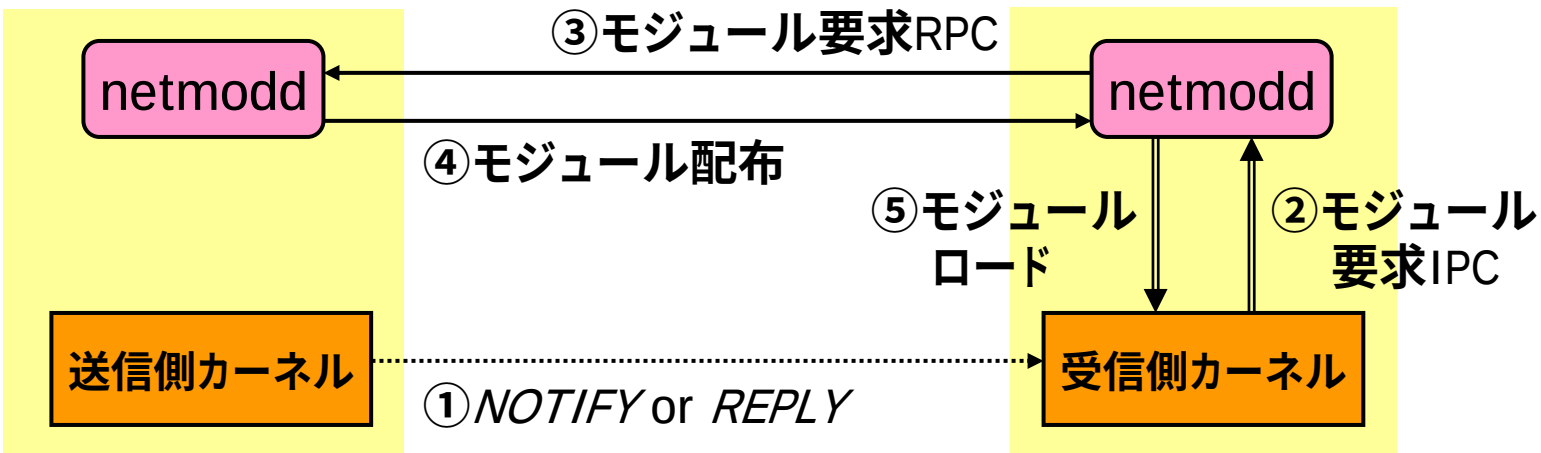
Broadcast/multicastへの対応

- *NOTIFY*メッセージを送る時に参加していたマシン
  - 正常に変換テーブルを作成できる
- その後で参加したマシン
  - パケットを受け取った時に*REQUEST*メッセージを送ることで変換テーブルを作成できる



ネットワークモジュールの自動配布機構の実装

- 必要なモジュールがシステムにロードされていなければデーモンnetmoddに要求を出す
  - DPNAPの*NOTIFY*または*REPLY*メッセージの受信時
- netmoddは送信側にモジュールを送ってもらい、システムにロードする

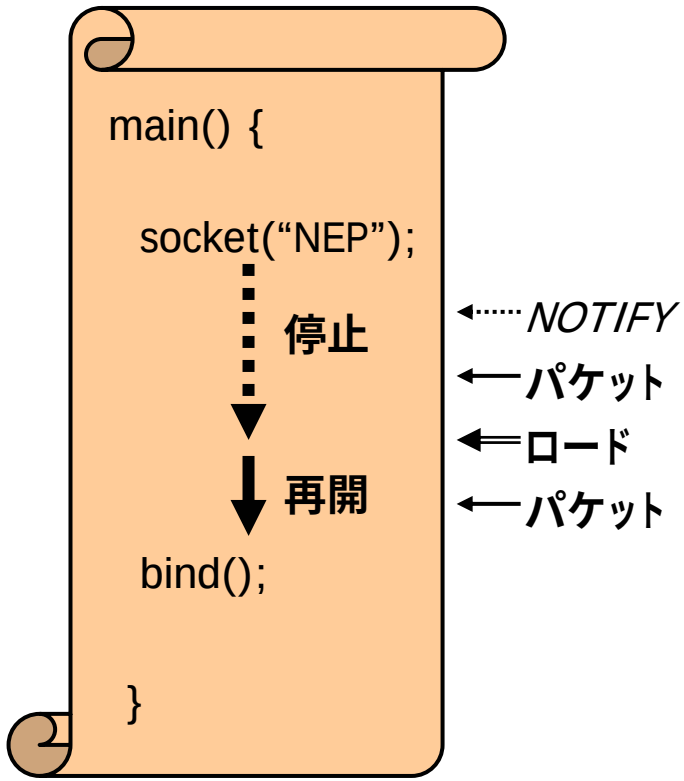


ネットワークモジュールのロード

- NetBSDのloadable kernel moduleの機構を使ってシステムにモジュールをロードする
  - 安全性、異機種分散は考慮されない
- ロードした時に…
  - そのマシン内だけで一意に識別できるプロトコル番号を割り当てる
  - プロトコルが使うネットワークアドレスを自動的に設定する
    - イーサネットアドレスかIPアドレス

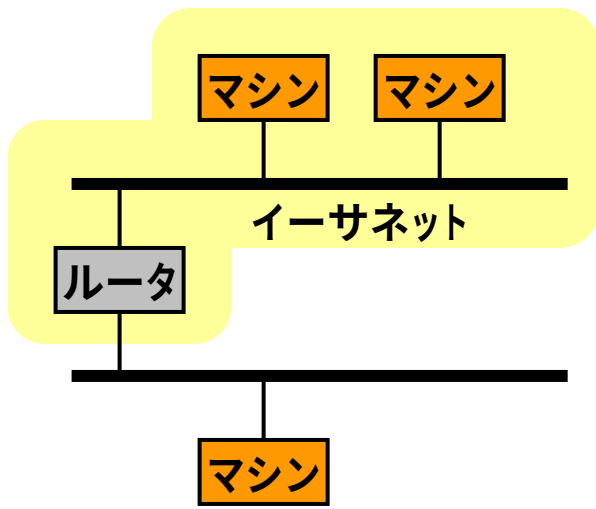
アプリケーションの視点

- socketシステムコール
  - モジュールがロードされていなければ実行を一時停止する
- bindシステムコール
  - ロードされたモジュールを使ってパケットの処理を開始する
    - パケットを受け取るソケットが決まるまで待つ
    - バインドが完了するまでに送られてきたパケットは保持されている



専用プロトコルの例：NEP

- イーサネットの同一セグメントでの通信に特化したプロトコル
  - 特徴
    - イーサネットドライバと直接パケットをやりとりする
    - ポートによりパケットを多重化する
    - チェックサムを計算する
  - 制限
    - パケットサイズは1500バイトまで
    - イーサネットセグメントを越えられない
    - 信頼性は保証しない

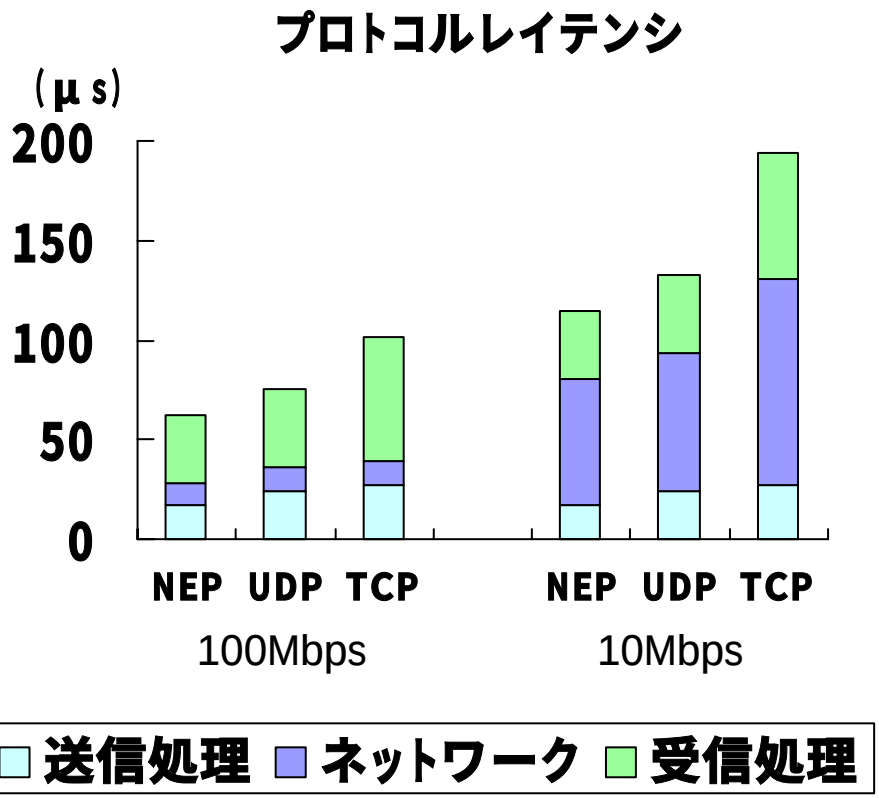


実験

- 実験の目的
  - 専用プロトコルNEPによる性能向上の割合を調べる
  - 動的にプロトコル番号を解決するオーバヘッドを調べる
- 測定内容
  - NEP, UDP/IP, TCP/IPについて1バイトのパケットを送り、end-to-endのレイテンシを測定した
- 実験環境
  - ＰＣ (PentiumII/400MHz、メモリ128MB) ２台
  - 3COM Fast EtherLinkネットワークカード
  - 100Mbps/10Mbpsのイーサネット

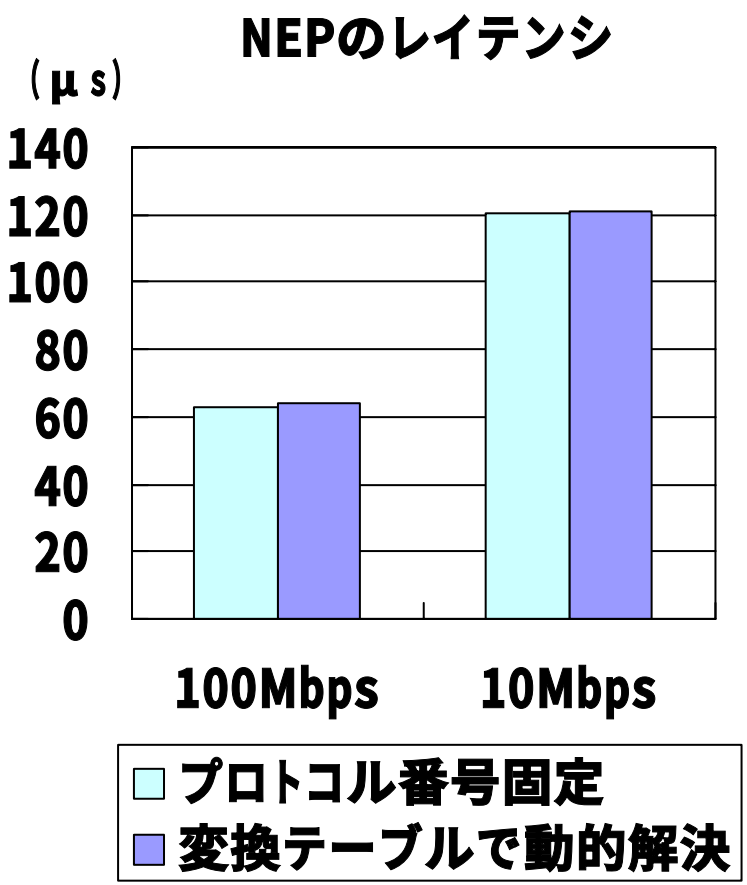
実験結果：専用プロトコルNEPによる性能向上の割合

- 性能向上の割合
  - UDP/IPに対して
    - 16% (100Mbps)
    - 9% (10Mbps)
  - TCP/IPに対して
    - 38% (100Mbps)
    - 38% (10Mbps)
- 実際にはさらに性能向上が可能
  - キャッシュミスの増大



実験結果：動的にプロトコル番号を解決するオーバヘッド

- 動的に解決するオーバヘッド
  - 1.8% (100Mbps)
  - 0.6% (10Mbps)
- 性能向上の割合から見ると十分小さい



関連研究

- **拡張可能**OS[Bershad et al.95]
  - **拡張モジュールを安全にシステムに組み込む**
  - **単一システムの拡張のみ**
- JavaOS
  - **アプレットをサーバからダウンロードして使う**
  - **アプレットを提供するサーバが必要である**
- Active Network[Tennenhouse et al.97]
  - **パケットにデータとそれを処理するコードを入れて送る**
  - **主なターゲットはルータ**

まとめ

- 適応的な分散OSの研究の一環として、適応的なネットワークシステムを構築している
  - ネットワークモジュールを自動配布し、通信を高速化する機構を提案した
    - プロトコル番号を動的に解決するためにDPNAPを開発した
- 専用プロトコルは性能を改善できる可能性が十分にあることを実験で示した
  - 100Mbpsで16%～38%の性能向上

今後の課題

- 異機種分散環境でネットワークモジュールを配布できるようにする
  - バイトコードを使ってモジュールを配布する、など
- ネットワークモジュールの安全性を考慮できるようにする
  - バイトコード、ネットワークシステムをどう設計するか
- 分散透明な適応ができるようにする
  - 分散システムにまたがって利用できる実行環境の情報を収集する