オペレーティングシステムと ネットワークプログラミング

担当: 吉藤 英明

E-Mail: yoshfuji+camp2008 AT wide.ad.jp

アウトライン

- 自己紹介
- 組織化と抽象化
  - 人、プログラム、行為
- 通信と標準化
  - オープンシステム
  - TCP/IP
- ソケットAPI(実習)

自己紹介

- 1974年東京生まれ
- 博士(情報理工学) (東京大学)
- 慶應義塾大学大学院政策・メディア研究科
- USAGIプロジェクト設立/コアメンバー
  - LinuxにおけるIPv6スタックの研究開発
- Linuxカーネルネットワーキング分野[IPv4/IPv6]共同保守担当(Co-Maintainer)

組織化

- 大きくなるとわかりづらくなる
  - みんなばらばら
- 近いものや一定の役割で固まる「組織」
  - “全体として”わかりやすくする
- 「組織化」優先で本末転倒にならないように注意

組織化（というか分解）[演習]

- 「人」を分解してみよう
- いろいろな方法がある

プログラムの組織化

- プログラムでも同様
  - 見通しがよくなる
    - 不具合を直しやすくなる
    - 期待している結果か検証しやすくなる
  - 複数人で手分けをして開発しやすくなる
  - 十分に「汎用」であれば別のプログラムに流用できる
  - 機能を増やしやすくする
- そうでないと…
  - みんなばらばら
  - 全体としてうまく動かなくなる

プログラムの分類

- BIOS (Basic Input/Output System)
  - 最も低レベルの入出力のための基本的なプログラム
- カーネル(Kernel; 核)
  - 特権操作
  - ハードウェア抽象化、資源管理と効率的な配分
- ユーザランド(Userland)
  - シェル(Shell)
  - 各種“アプリケーション”など

ユーザランド

- アプリケーション(Application)
  - Firefox, OpenOffice.Org, Emacs, ...
  - 全てのアプリケーションが最初から全て書かれているわけではない
- ライブラリ(Library)
  - よく使うものをまとめたもの
    - Linux: *.so, *.a
      - /lib をみてみよう
    - Windows: *.DLL, *.LIB

オペレーティングシステムとミドルウェア

- オペレーティングシステム(Operating System)
  - 目的とする機能を実現するために種々のソフトウェアをまとめたもの
  - 一般にカーネルからユーザスペースまでを含む
- ミドルウェア(Middleware)
  - アプリケーションとカーネルとの仲立ち
  - 比較的大規模なライブラリとその管理用アプリケーション群
  - 例: データベース、クラスタ制御、...

「行為」の分解[演習]

- (ルーズリーフでない)ノートに整理してメモを取る
- (遠くの)友達に連絡をとる

「行為」の分類

- 「ノートに書く」と「電話で話す」は同じか？
  - 行為者から見ると同じ側面も多い
  - 出す(`write()`)、入れ(`read()`)
- コンピュータ上の「ファイル」と「通信」
  - プログラムから見るとデータの入出力
    - ファイル
      - “名前”に紐付けられた記憶域との情報の出し入れ
    - 通信
      - “特定の”相手との情報のやりとり

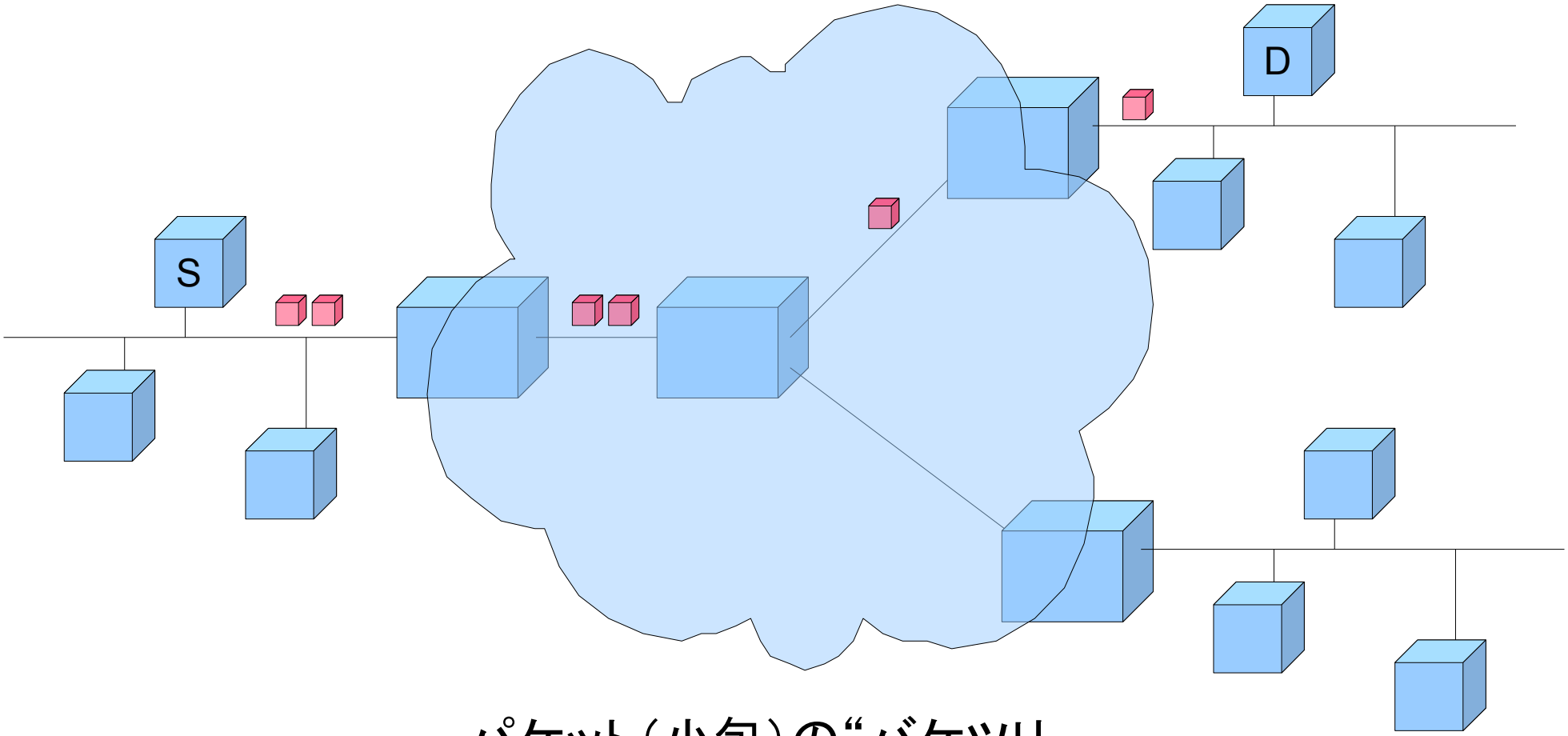
通信

- 情報の送受信(放送を含む(広義))
  - ユニキャスト, マルチキャスト, ブロードキャスト
- ソケット
  - 通信の端点（受話器）
- 通信相手の決定はユーザランドの責任
  - 一般にライブラリの助けを借りる

通信と標準化

- 多種多様なシステムへの接続/利用要求
  - 手順(プロトコル)の標準化
  - APIの標準化
  - 「オープンシステム」
- 通信プロトコルの代表: TCP/IP
  - IETF (Internet Engineering Task Force)
- API(Application Programming Interface(BSDソケットAPI)
  - POSIX (Portable Operating System Interface)ほか

インターネット

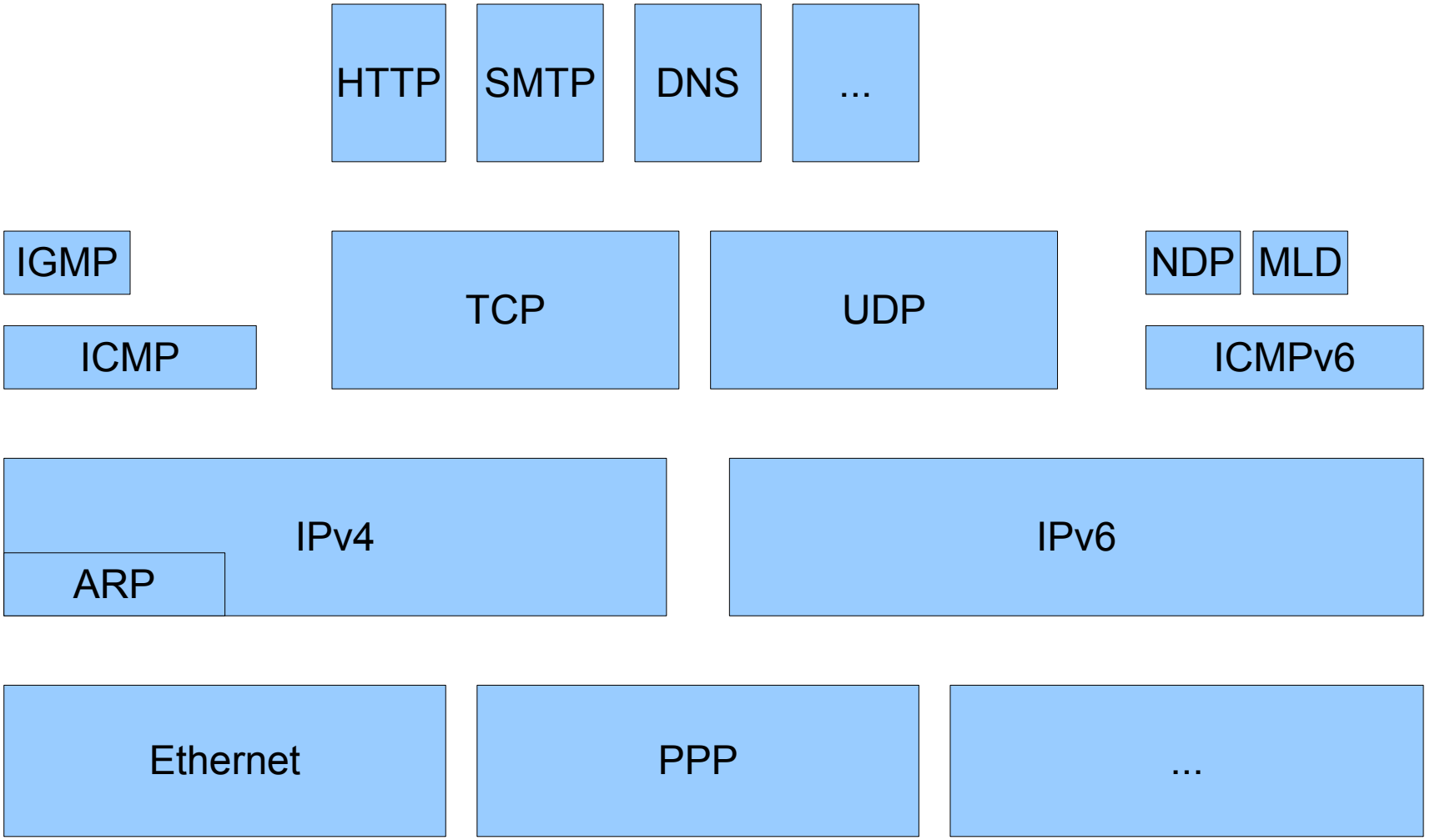


パケット（小包）の“バケツリレー”方式による転送

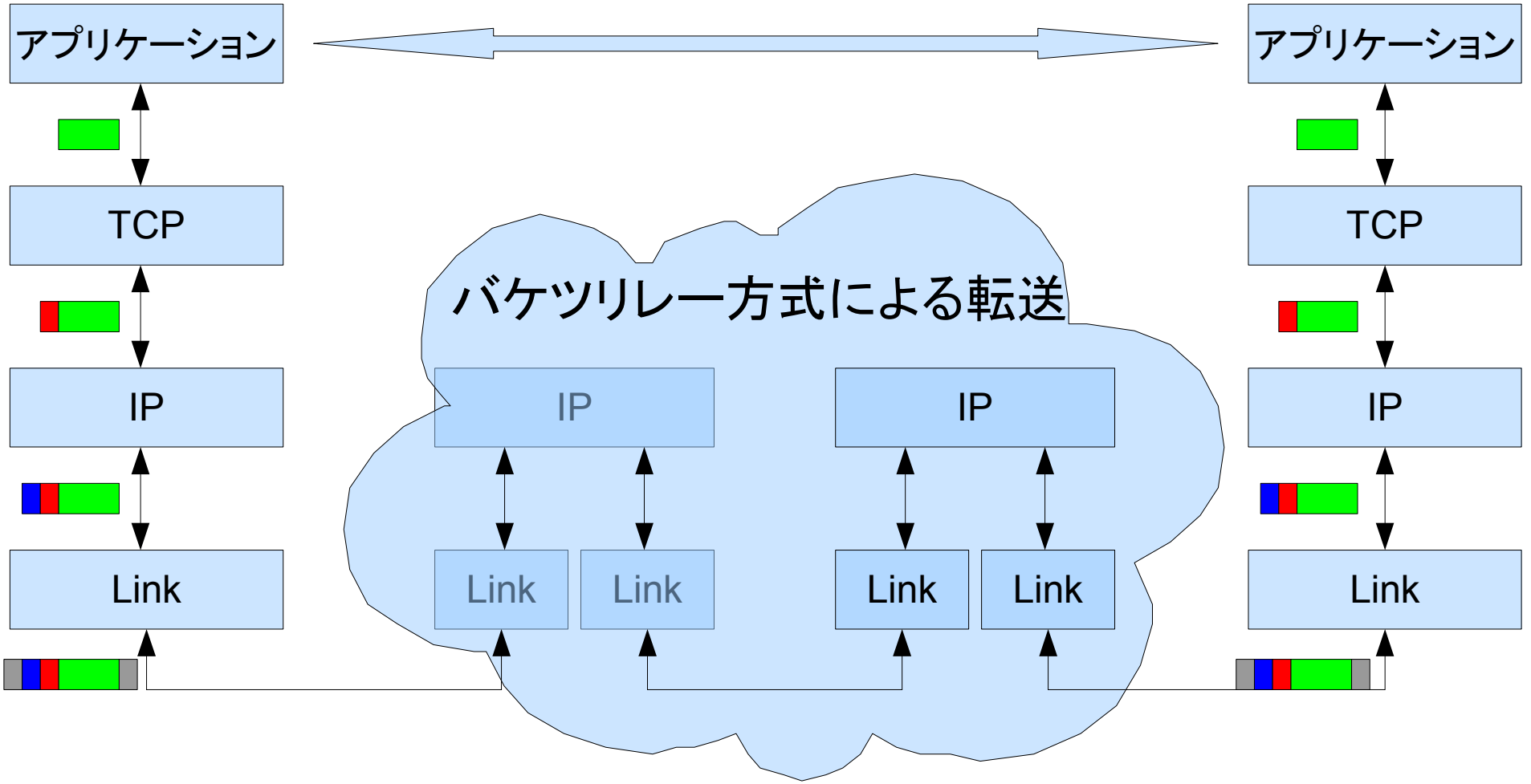
TCP/IP
（インターネットプロトコルスイート）

- アプリケーション(Application)層: HTTP, SMTP, ...
- トランスポート(Transport)層: TCP, UDP, ...
- インターネット(Internet)層: IPv4, IPv6
- データリンク(Data Link)層: イーサネット(Ethernet), ...
- 物理(Physical)層: イーサネット物理層

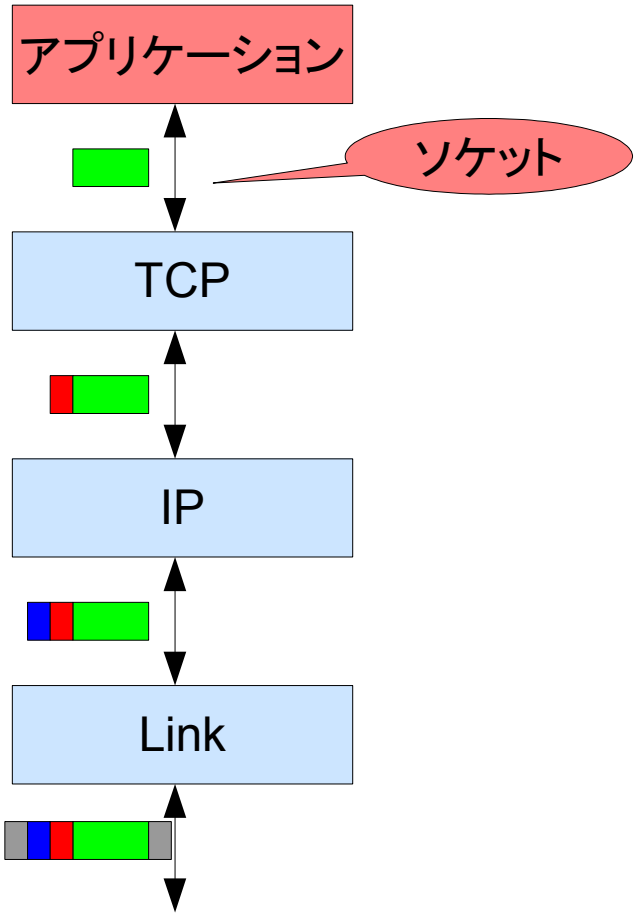
TCP/IP



TCP/IP
（インターネットプロトコルスイート）

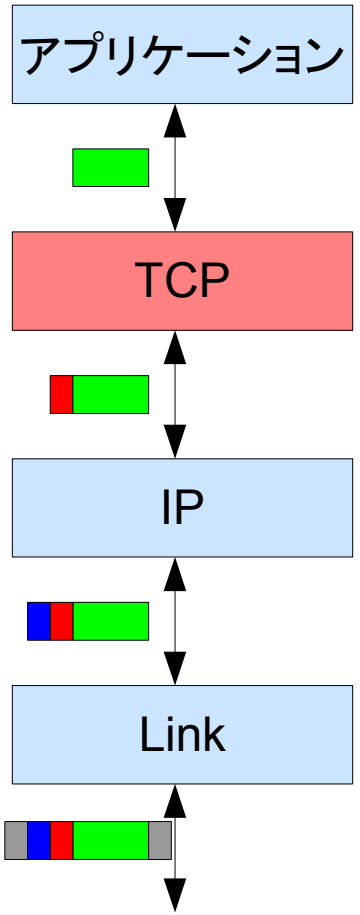


アプリケーション(Application)層



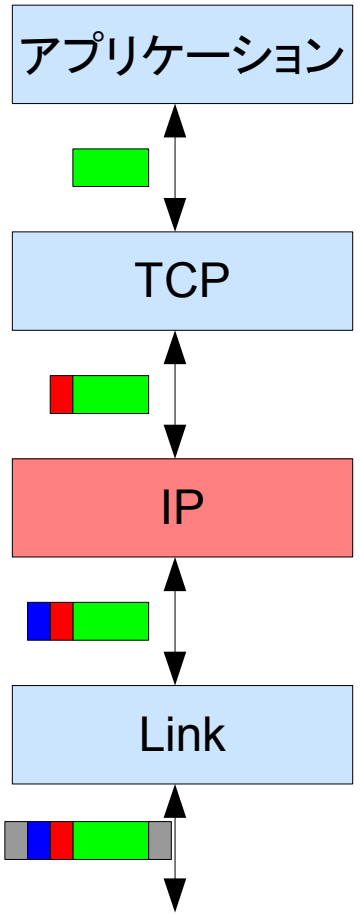
- HTTP(Webページ), SMTP(メール), XMPP(メッセンジャー), …
- 実際にTCP/IP通信を行おうとする層
- ソケットは下層とのAPI

トランスポート(Transport)層



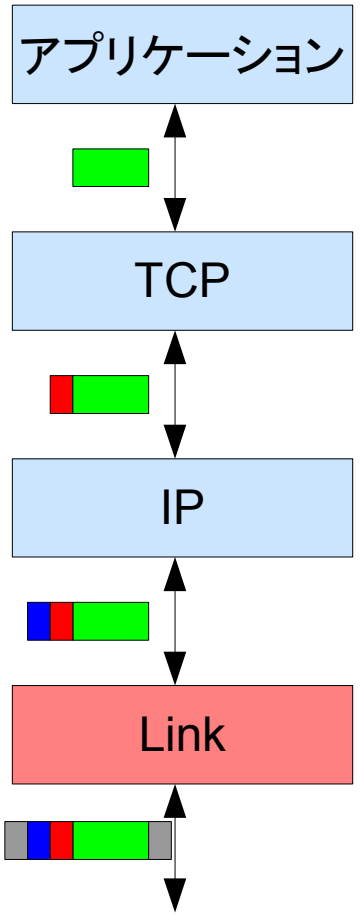
- TCP, UDPなど
- 適切なアプリケーションプロセスへデータを配送
  - ポート番号
- 仮想回線(コネクション)、誤り訂正、再送、フロー制御

インターネット(Internet)層



- IP(IPv4, IPv6)
- ネットワークを介したホスト間(エンド・ツー・エンド)の通信を実現
- IPアドレス

データリンク(Data Link)層



- Ethernetなど
- 近隣ノード間の通信を担当
- MAC(Media Access Control)アドレス
  - Ethernet: EUI-48

BSDソケットAPI

- 通信の端点
  - プロトコル、アドレス、ポート番号(5 tuples)
- 接続手順(TCP)
  - 受信: ソケット生成(socket)、ポート番号指定(bind)、聴取開始(listen)、受信(accept)、通信、終了(close)
  - 送信: ソケット生成(socket)、(ポート番号指定(bind)、)あて先指定(connect)、通信、終了(close)
- その他
  - ソケットオプション(setsockopt, getsockopt)

BSDソケットAPI: コア関数群

```
int socket(int domain, int type, int protocol);

int bind(int sockfd, struct sockaddr *myaddr,
         socklen_t addrlen);

int connect(int sockfd, const struct sockaddr
            *serv_addr, socklen_t addrlen);

int listen(int sockfd, int backlog);

int accept(int sockfd, struct sockaddr *addr,

           socklen_t *addrlen);
```

BSDソケットAPI(2)

- 名前解決(ホスト名/文字列表現⇔IPアドレス)
  - **getaddrinfo() / getnameinfo()**
  - **inet_pton() / inet_ntop()**
- 旧式
  - **gethostbyname() / gethostbyaddr()**
  - **inet_addr() / inet_aton()**

BSDソケットAPI(2): 名前変換関数群(1)

```
int getaddrinfo(const char *nodename,
                const char *servname,
                const struct addrinfo *hints,
                struct addrinfo **res);
struct addrinfo {
    int ai_flags;               /* フラグ */
    int ai_family;              /* プロトコルファミリAF_xxx */
    int ai_socktype;            /* ソケットタイプSOCK_xxx */
    int ai_protocol;            /* プロトコルタイプIPPROTO_xxx */
    socklen_t ai_addrlen;       /* ソケットアドレス構造体の長さ */
    char *ai_canonname;         /* ノードの正式名 */
    struct sockaddr *ai_addr;   /* ソケットアドレス構造体 */
    struct addrinfo *ai_next;   /* 次のアドレス情報 */
};
```

- プロトコルに依存しない名前変換関数
- **socket()**や**bind()**にそのまま利用できる形で情報を得られる

BSDソケットAPI(2): 名前変換関数群(2)

```
int getnameinfo(const struct sockaddr *sa,
                socklen_t salen,
                char *host, socklen_t hostlen,
                char *serv, socklen_t servlen,
                int flags);
```

- プロトコルに依存しないアドレス-名前変換関数
- **accept()**や**getpeerinfo()**など、カーネルから得られる構造をそのまま使うことができる

簡単な通信プログラムを作ってみよう

- TCP利用
- エコークライアント
  - 相手のサービスに接続
  - 標準入力に入力した文字列を相手に送信
    - fgets, NUL文字終端
  - 相手から返ってきた文字列を表示
- エコーサーバ
  - 特定のポートを開いて接続を待ち受ける
  - 接続されたら、接続元を画面に表示
  - 相手から入力された入力を相手に送り返す

通信プログラムの発展: チャットサーバ

- チャットサーバ
  - 複数人が同時にひとつのサーバに接続できるようする
    - **`select()`**
  - 接続されてきたらほかの参加者に知らせる
  - 入力は全ての参加者に送る
- さらなる発展
  - 文字コード変換: **`iconv()`**
  - 発言時刻の表示を加える
  - 「ないしょ」機能
  - …

通信プログラムの発展(2): Webサーバへのアクセス(1)

- Webサーバへのアクセス
- URL: http://www.example.com/path/to/file.html
  - http://
    - スキーム(プロトコル)
  - www.example.com
    - ホスト名
  - /path/to/file.html
    - パス

通信プログラムの発展(2): Webサーバへのアクセス(2)

- HTTPプロトコル
  - サーバの80番ポートに接続

    GET /path/to/file.html HTTP/1.0[CR][LF]

    Host: www.example.com[CR][LF]

    [CR][LF]

商標その他

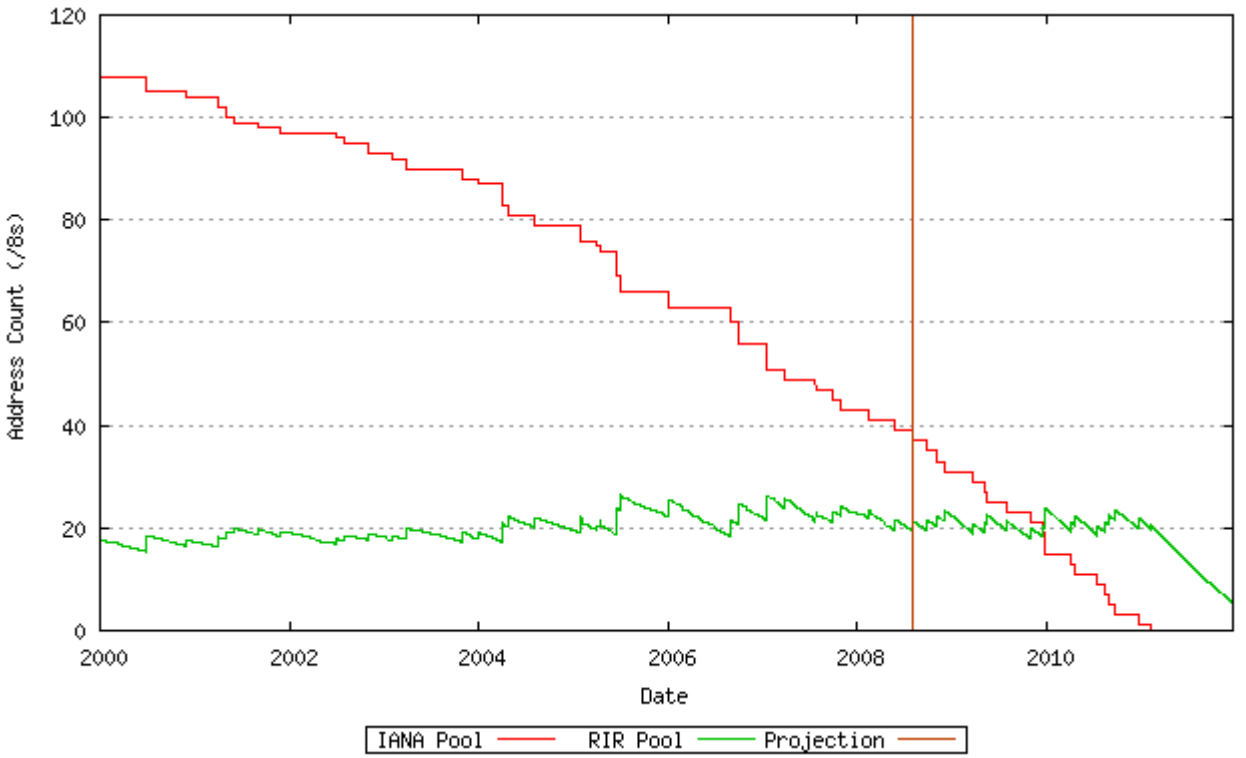
- この資料は「セキュリティ＆プログラミングキャンプ2008」のためにまとめられたものです。
- Linux, UNIX, Windows その他社名、製品名、サービス名などは、各社の商標または登録商標です。

付録

IPアドレスの枯渇

- IPv4アドレス
  - $2^{32}=3.4\times10^{9}$(43億)
  - 2012年前後には枯渇すると予測

IPv4アドレスプールの減少



Source: http://www.potaroo.net/tools/ipv4/

IANAアドレスプール枯渇予測: 2011/2/8
RIRアドレスプール枯渇予測: 2011/12/21
(2008年8月5日現在予測)

IPバージョン6

- 特徴
  - アドレス空間の拡張
    - $2^{128}=3.4\times10^{38}$(340澗(かん))
  - セキュリティやモビリティ(移動透過性)のサポート
- 現行のほとんどのオペレーティングシステムやルータは対応
  - Vista, Linux, *BSD, ...

IPv6ヘッダ