インターネット
アーキテクチャ概論

慶應義塾大学
村井　純
jun@wide.ad.jp

チュートリアルの目的

- インターネットアーキテクチャを構成する基幹技術を理解する
- インターネットアーキテクチャの持つ特徴を理解する

キーワード

- Best Effort
- Scalability
- Security
- Operation
- Autonomy
- Distributed
- Exponential Growth
- End System

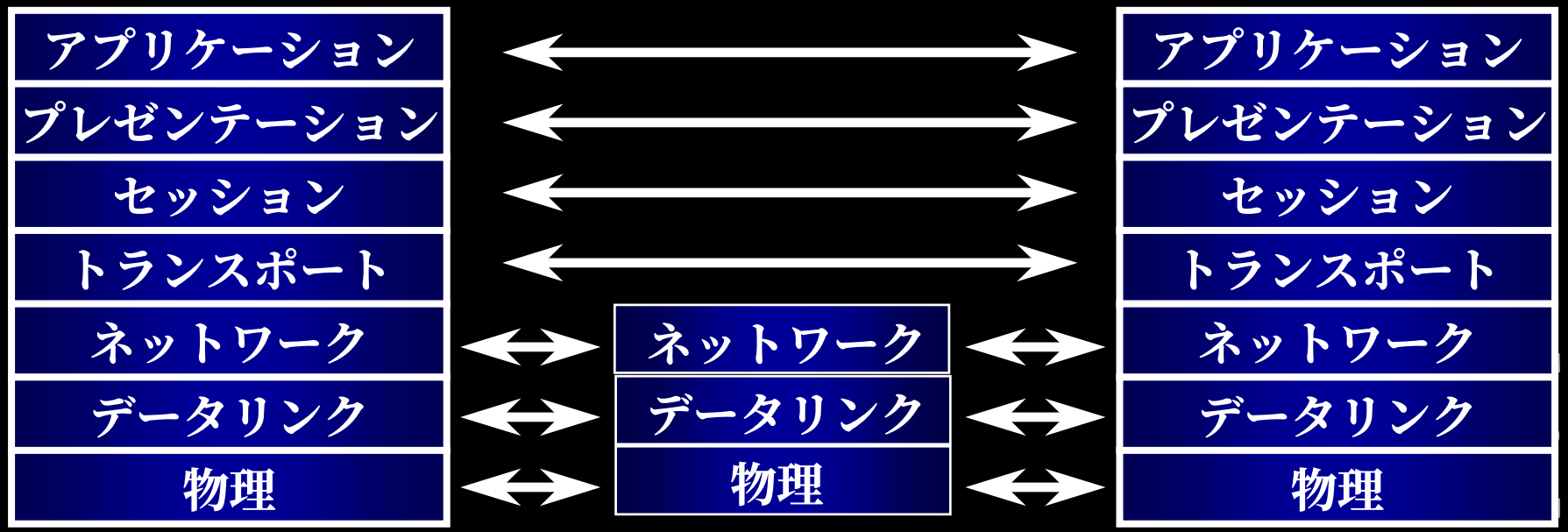
AGENDA

- はじめに
- インターネットのコア技術
- インターネットの運用技術
- 通信媒体とインターネットアーキテクチャ
- 環境適応のための技術
- アプリケーションアーキテクチャ

はじめに

OSIの7階層モデル

- プロトコル設計のものさし
- わかりやすい機能の分離と独立性
- エンドシステムの n層同士が通信を行っているつもり
- 中間システムは1～3層でリレーするだけ



OSI各層の役割(1)

- 物理層(第1層)
  - 物理メディアへの直接接続の提供
- データリンク層(第2層)
  - 宛先のシステムまでのパスのうちの隣接するシステムとの間の情報の流れを制御
- ネットワーク層(第3層)
  - エンドシステム間のコネクションの確立

OSI各層の役割(2)

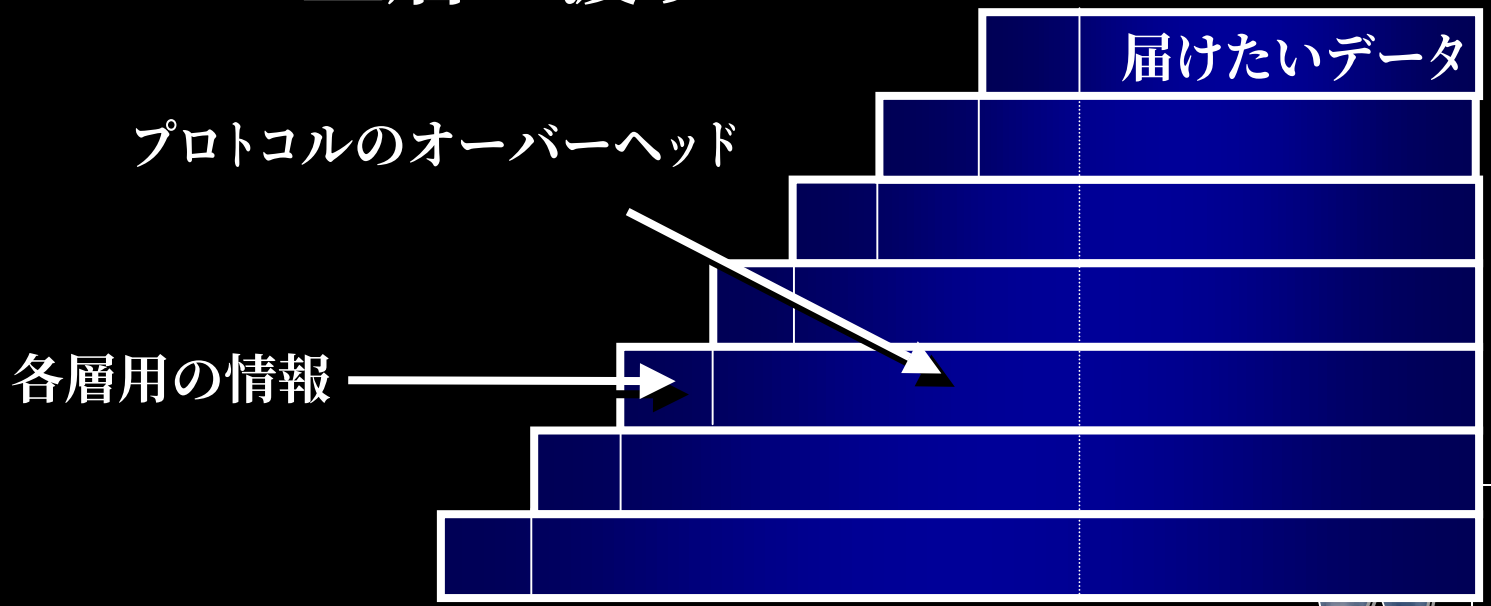
- トランスポート層(第4層)
  - スループットと信頼性について、上位層にサービス品質を保証
- セッション層 (第5層)
  - システム間の対話に使われる枠組みの確立に対する責任

OSI各層の役割(3)

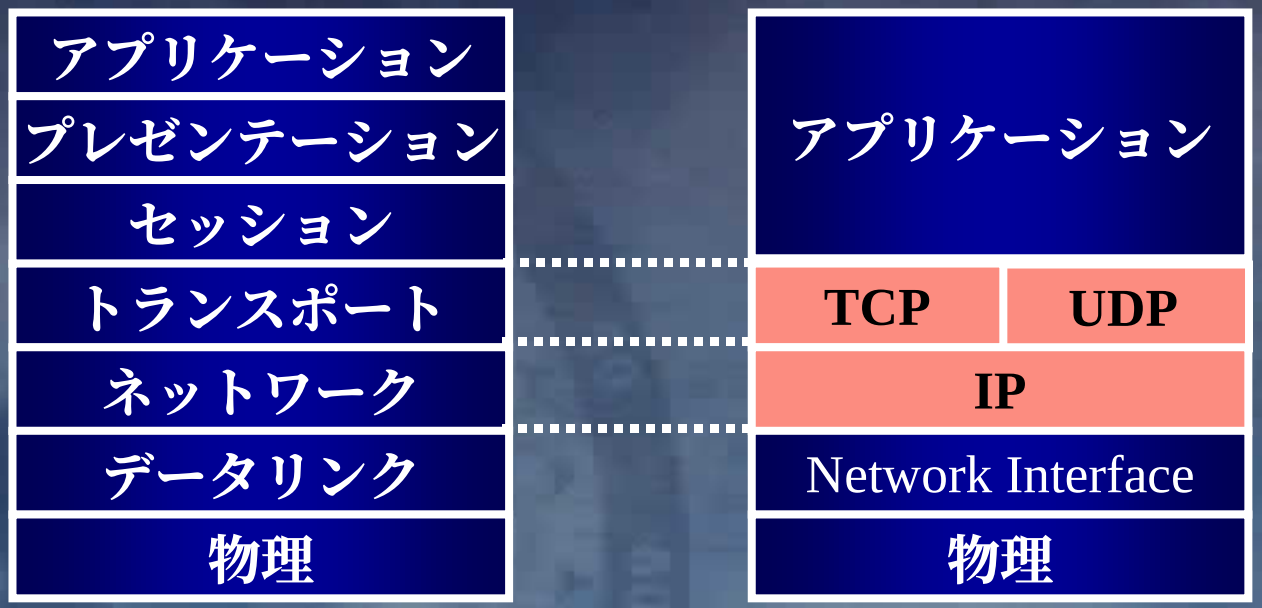
- プレゼンテーション層（第6層）
  - 転送される情報の表現の共通化
- アプリケーション層（第7層）
  - ユーザプロセスに通信環境へのアクセスを提供

階層型プロトコル

- 送信側：各層がそれぞれの情報を付加して下層へ渡す
- 受信側：各層がそれぞれの情報を取り除いて上層へ渡す



OSIモデルと
インターネットアーキテクチャ



通信形態

- バーチャルサーキット型
  - 仮想的なパイプを通した転送
  - コネクション型
- データグラム型
  - データを小分けにし、バラバラに転送
  - コネクションレス型

VC型 vs DG型

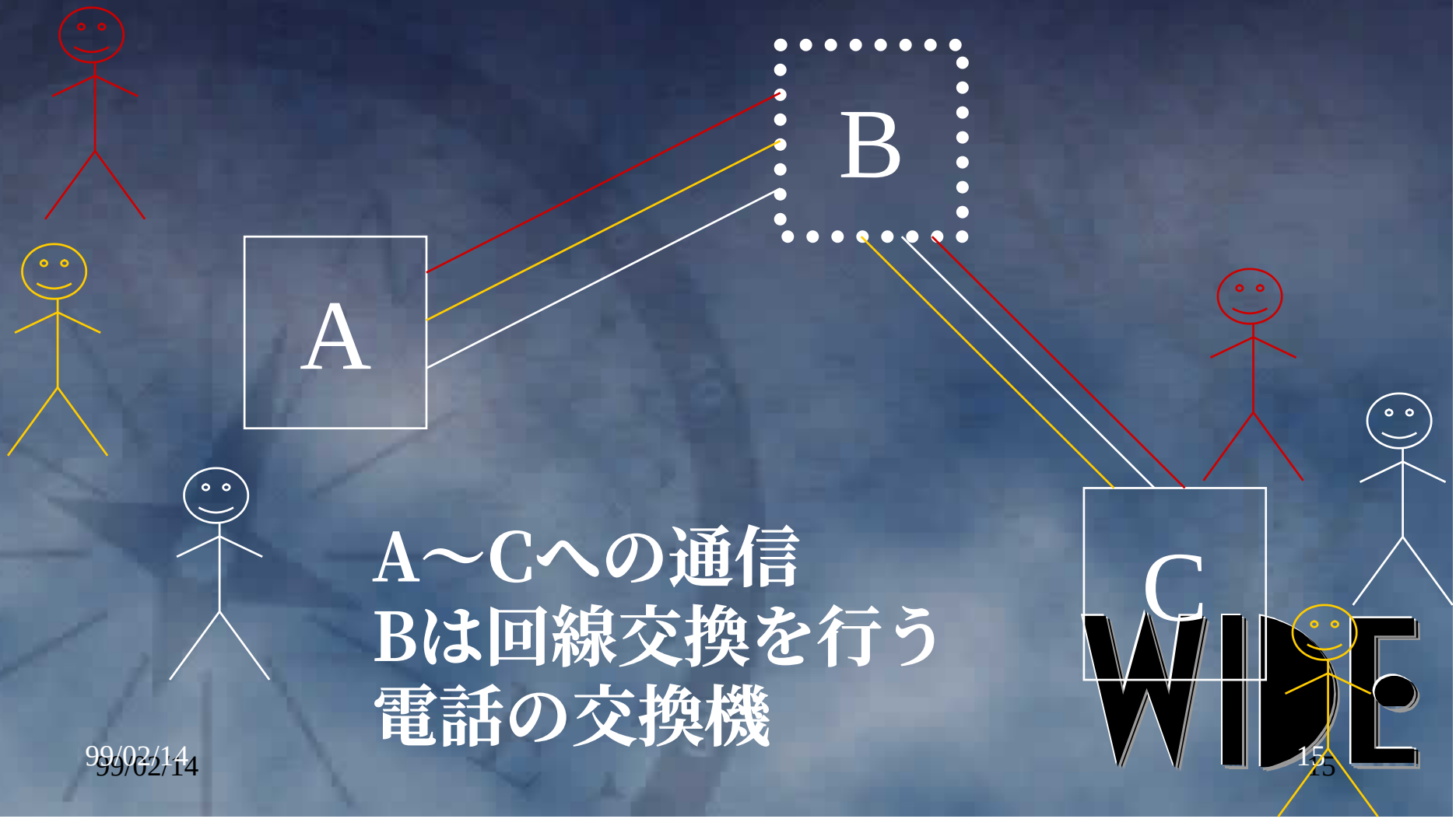
- バーチャルサーキット
  - 信頼性
  - 順序保証
  - フロー制御
  - 輻輳制御
  - 再送
  - 重量課金
- データグラム
  - 信頼性保証せず
  - 順序保証せず
  - フロー制御なし
  - 輻輳制御なし
  - 再送なし
  - 固定課金

VC型 vs DG型

- VC の手順 (例:FAX)
  - あて先を指定し、まず接続
  - 接続を確認してからデータを転送
  - 転送が終わったら接続を切断
- 送信順に相手に届く
- 信頼性があるが、オーバヘッドが大きく処理能力が必要

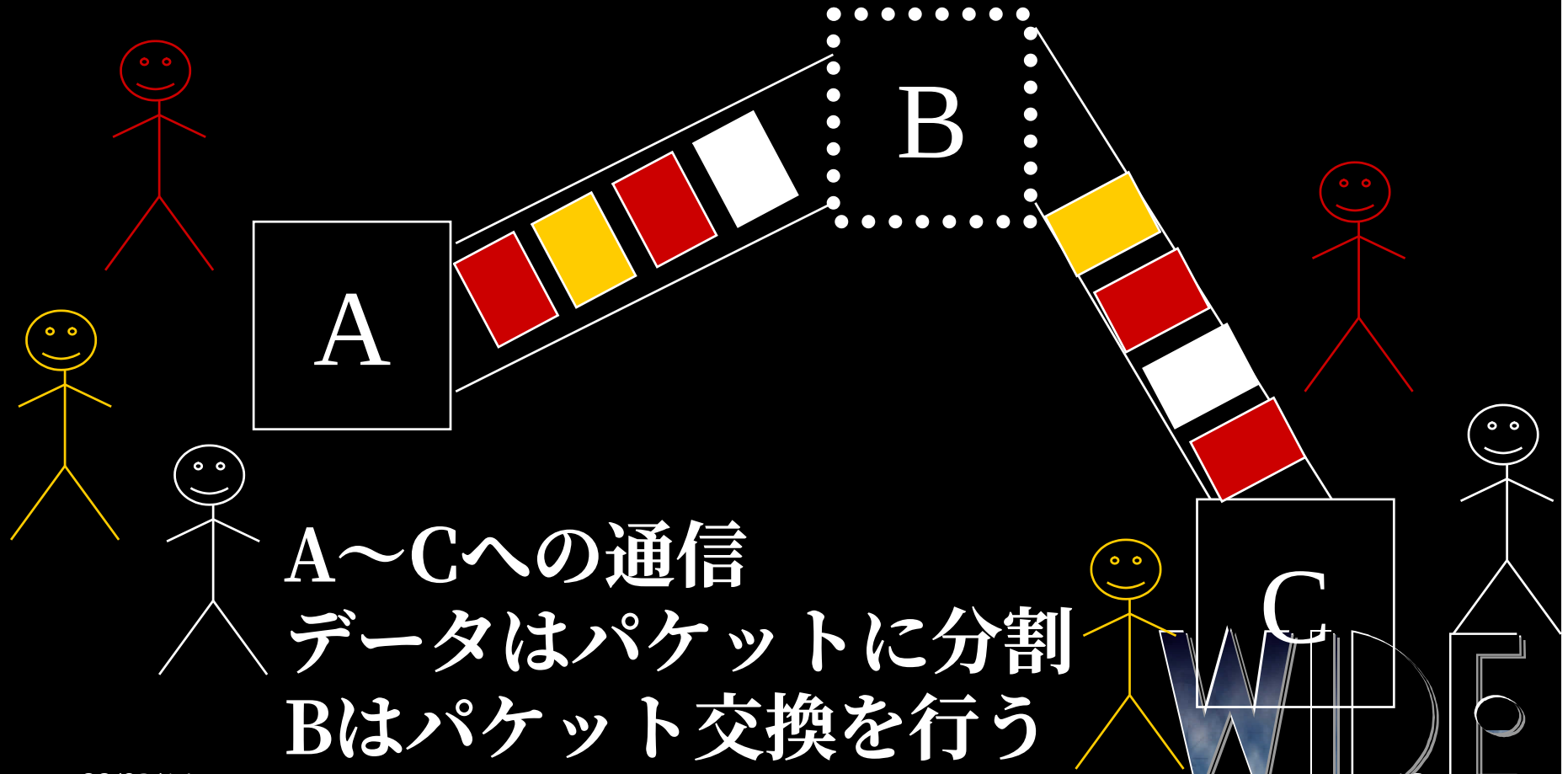
- DGの手順 (例:はがき)
  - あて先を指定してポストに投函
- 投函した順には届かない
- 信頼性はないが、オーバヘッドが少なく、処理能力を要求しない

回線交換方式



A～Cへの通信
Bは回線交換を行う
電話の交換機

パケット交換方式

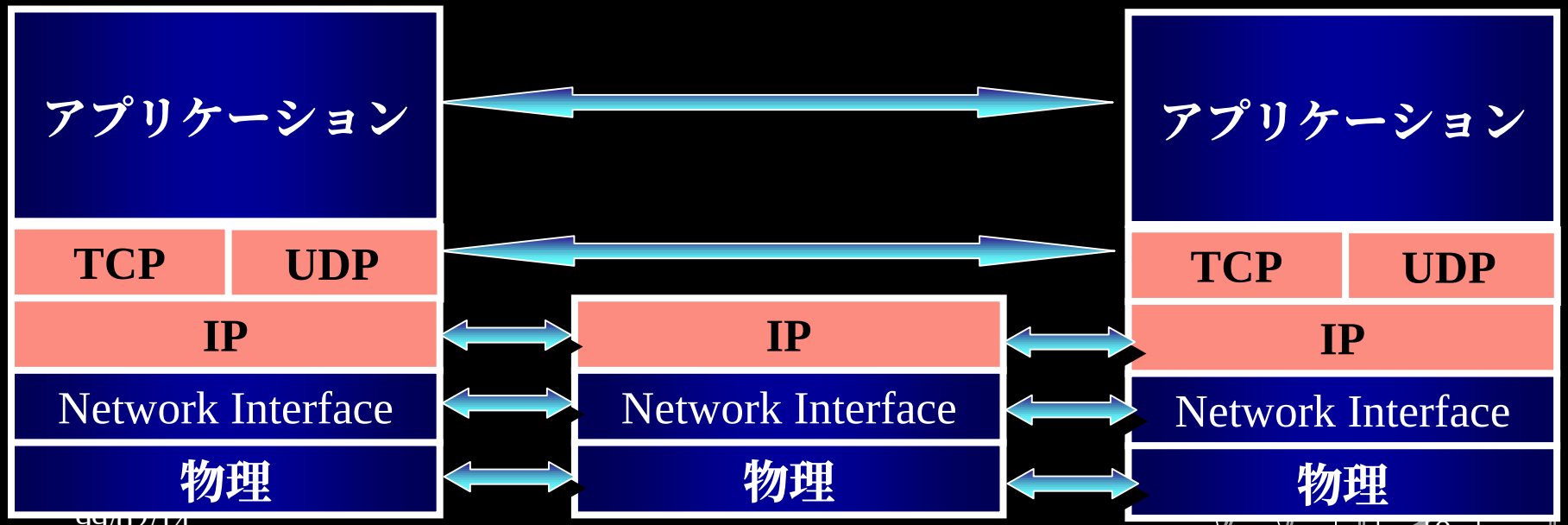


A～Cへの通信
データはパケットに分割
Bはパケット交換を行う

インターネットのコア技術

ネットワークプロトコル

- ネットワーク層
  - IP - Internet Protocol
- トランスポート層
  - TCP - Transport Control Protocol



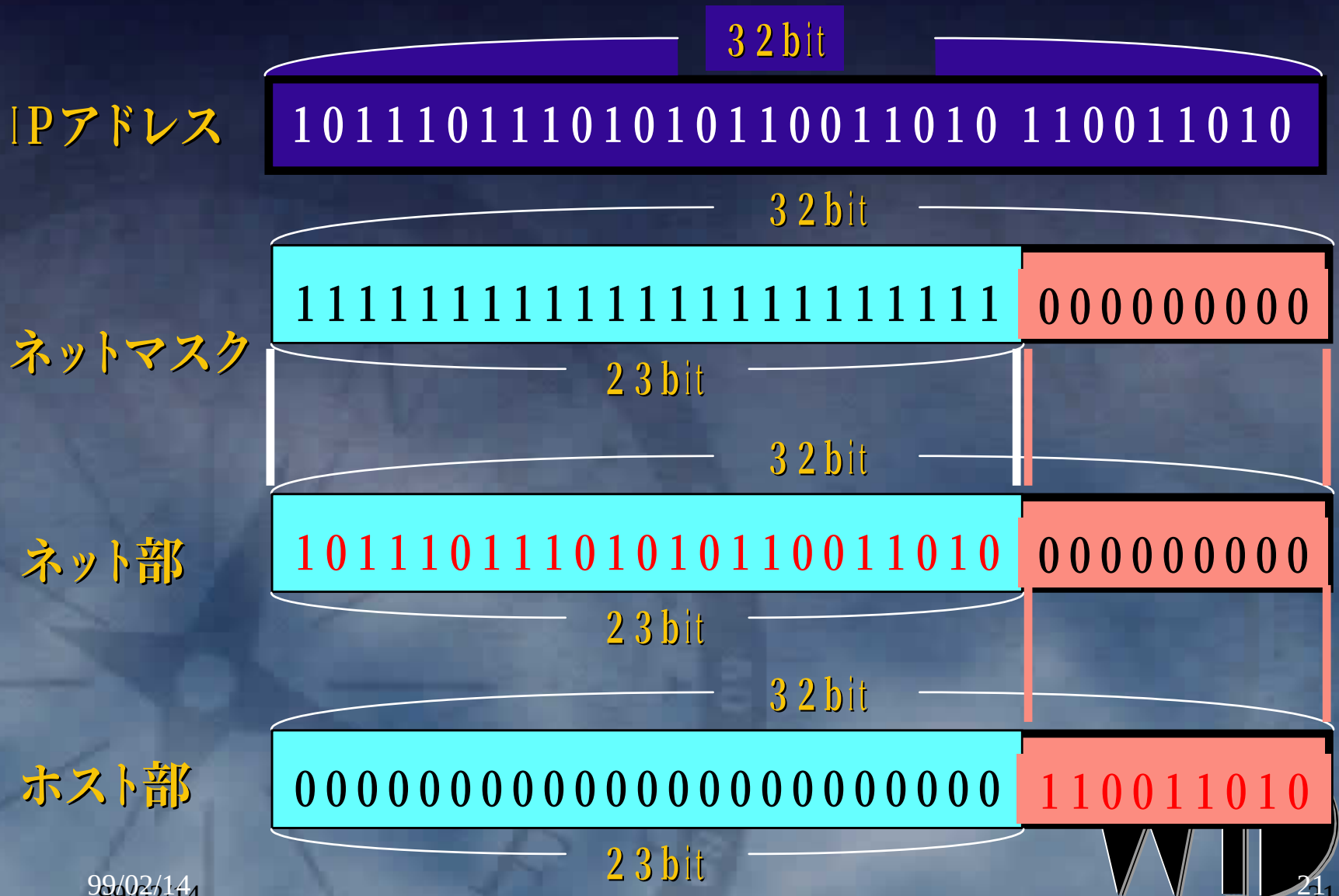
IPの機能

- ホストの識別
  - 32bit の識別子　ーIPアドレス
- 経路の決定
  - IPアドレスの一部から経路を決定
- データの中継
  - バケツリレー方式
- データの分割
  - 一度に送れないデータを分割、再構成

IPアドレス

- 32bitのアドレス空間(IPv4の場合)
- TCP/IPネットワークに接続するためのID
  - 世界中で唯一自分を証明するためのもの
  - 重複してはいけない
  - 自分と相手を認識する
- 構造化されたアドレス
  - ネットワークを表すネットワークアドレスとホストを表すホストアドレスから成り立つ
  - ネットマスクによる柔軟な構造
  - ネットワーク部から経路を決定

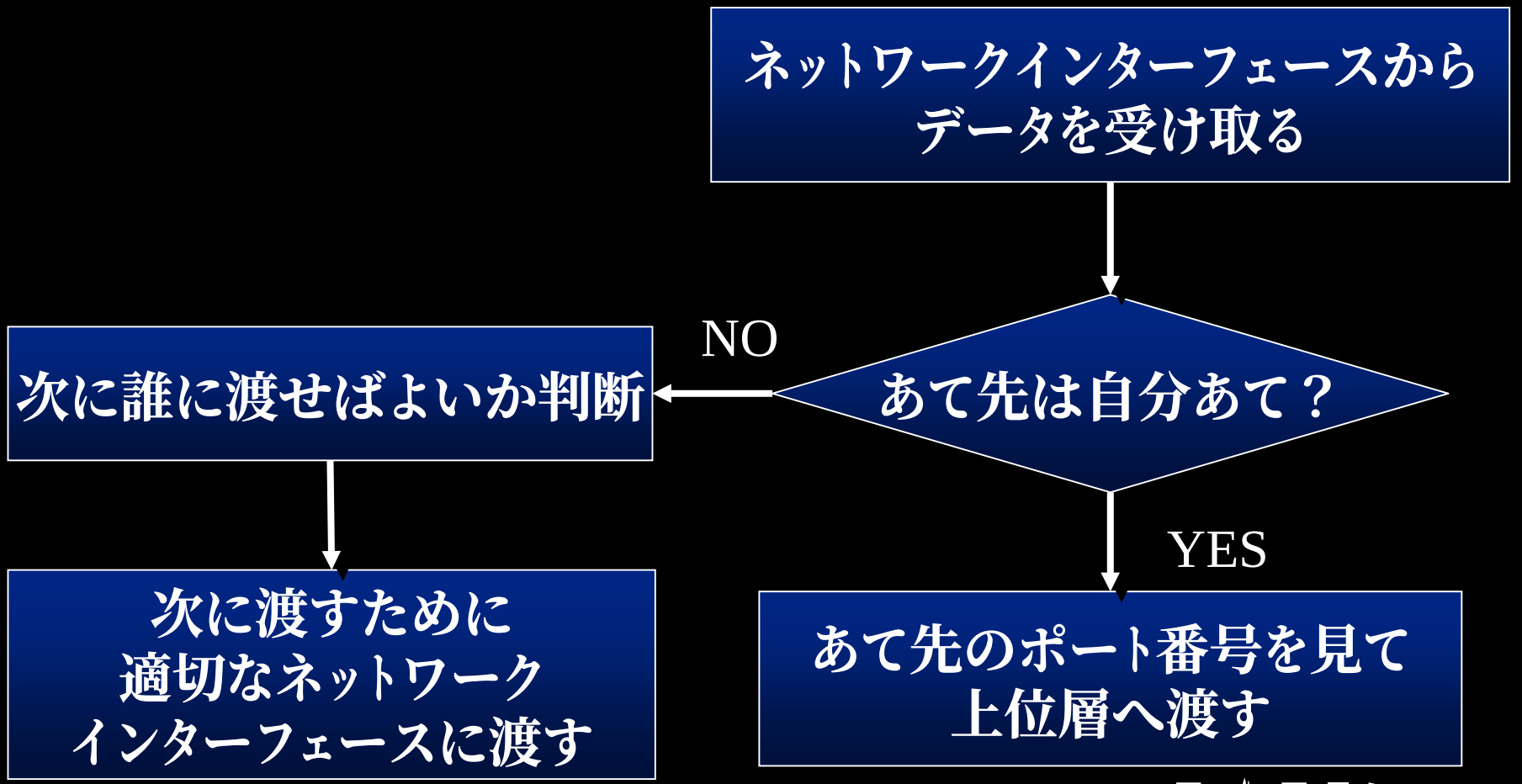
IPアドレスとネットマスク



IPの特徴

- データグラム型の通信
  - 信頼性のない通信を提供
  - 仕組みが単純
- Best Effort（最善努力型）
  - 中間ノードに一度に処理できない数のパケットが集まると待ち行列(queue)が発生(混雑の発生)し、限界を超えると破棄
  - エラーの訂正は行わないが、エラーの報告は行う
  - ICMP - Internet Control Message Protocol

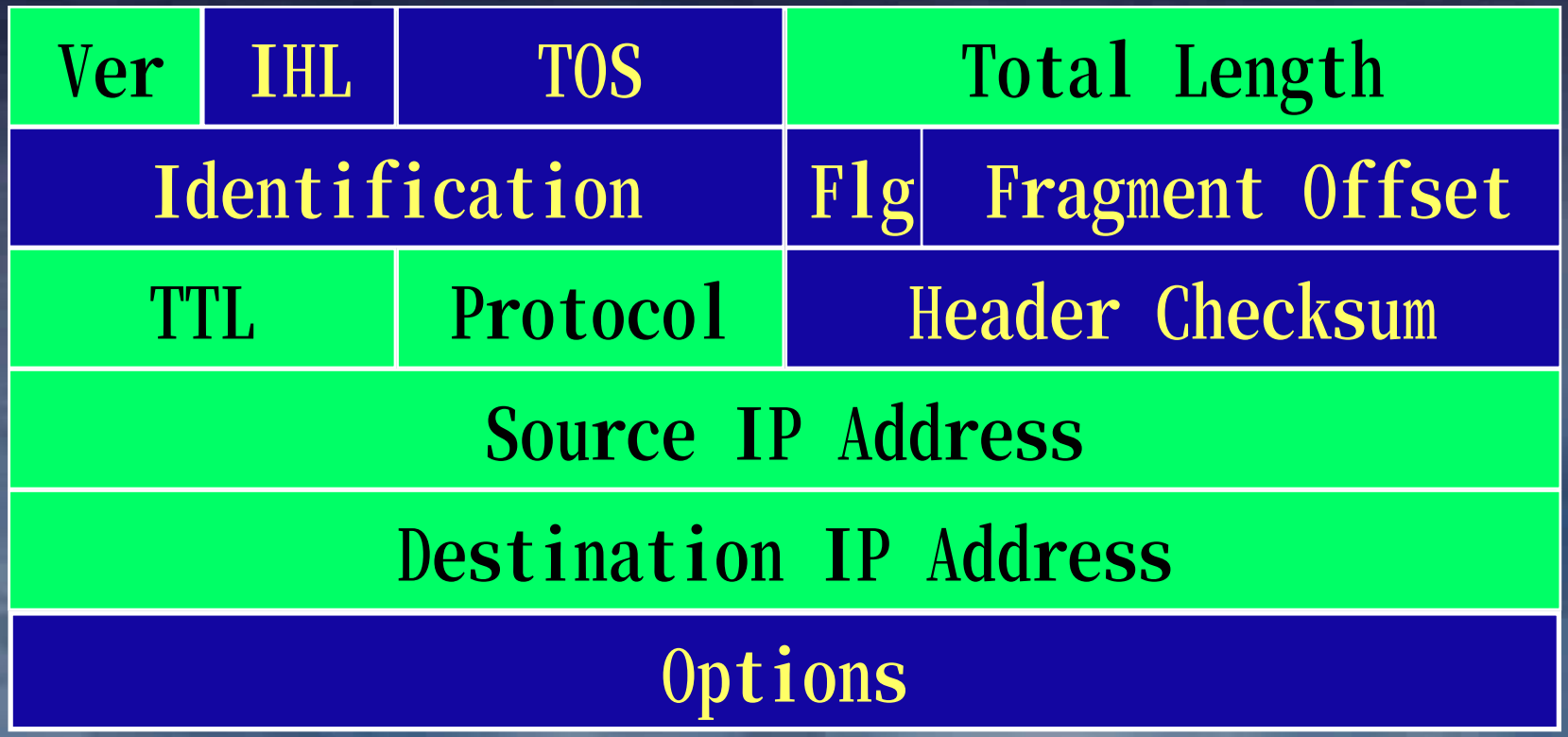
IPの仕事の流れ


ネットワークインターフェースから
データを受け取る
あて先は自分あて？
NO
次に誰に渡せばよいか判断
次に渡すために
適切なネットワーク
インターフェースに渡す
YES
あて先のポート番号を見て
上位層へ渡す

IPヘッダ

- データグラムの配送に必要な情報を含む
  - 送信者アドレス
  - 受信者アドレス
  - データグラムの長さ　etc. . . .
- 一つ一つのデータグラムに付加される
  - Virtual Circuitではない

IPヘッダ (IPv4)


Ver
IHL
TOS
Total Length
Identification
Flg
Fragment Offset
TTL
Protocol
Header Checksum
Source IP Address
Destination IP Address
Options
4 octets

IPヘッダ

- Ver
  - バージョンフィールド - 現在は 4
- IHL
  - ヘッダの長さ 4octet を 1 としてカウント
- TOS
  - Type of Service 配送のタイプ
- Total Length
  - IPデータグラム全体の大きさ（1octetでカウント）
- Identification
  - もともとのデータを識別する
  - Fragmentation の際に必要
- flg （Flag）
  - Fragmentされたか、Fragmentしてはならないか

IPヘッダ

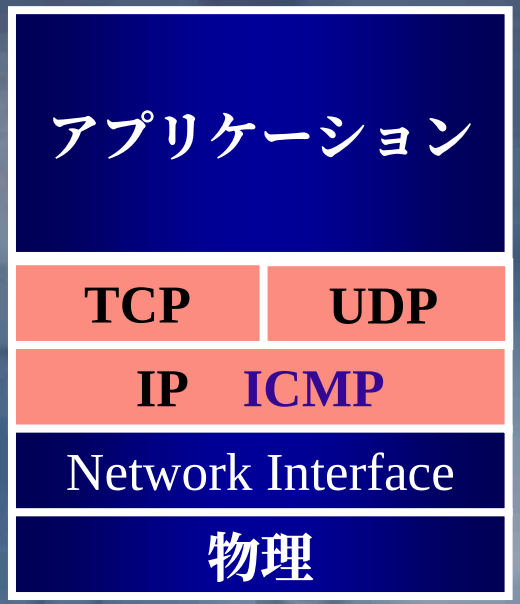
- Fragment Offset
  - 元のデータのどこでFragmentか
  - 8octets 単位で計られる
- TTL（Time To Live）
  - このデータグラムの寿命
  - 基本的には1つのホストを経由すると1減る
  - 初期値は64
- Protocol
  - 上位層が何か
- Header Checksum
  - ヘッダ部分の検証
  - データが壊れていないか
- Source Address
  - 送信元ホストのアドレス
- Destination Address
  - 送信先ホストのアドレス
- Option
  - 特別な機能を提供

IPとエラー

- エラー
  - データグラムの紛失
  - checksum の不一致
  - Fragment したデータの一部欠落
- IPとエラー訂正
  - IPはエラーの訂正は行わない
  - 送信者にエラーの報告を行う
  - エラー報告するのがICMP

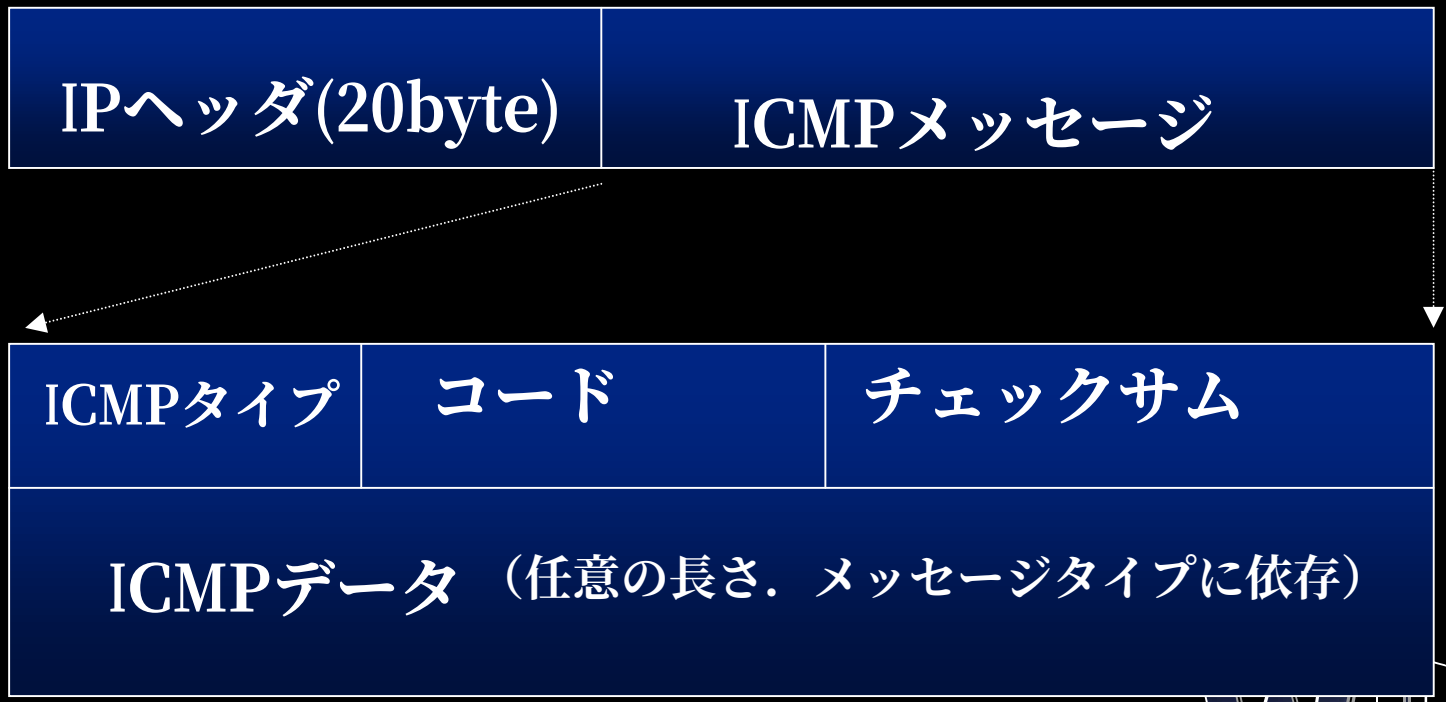
ICMP

- Internet Control Message Protocol
- 機能的にはIPを補うもの
  - 送信先への到達不能，フロー制御など
  - IPにかわってエラー報告する．



ICMPパケット

IPデータグラム



ICMPメッセージタイプ

- エコー応答 (Echo replay)
- 宛先到達不能 (Destination Unreachable)
- 発信抑制(Source Quench)
- ルート変更(Redirect)
- エコー要求(Echo request)
- Datagramの時間超過(Time exceed)
- Datagramのパラメータ異常(Parameter Problem)

ネットワーク層のまとめ

- ネットワーク層だけでは，信頼性のある通信を行うことはできない。
- 送信側と受信側とが応答確認する必要性がある。.

インターネットと鉄道網

- 客(データ)は必ず目的駅(コンピュータ)に到達する
- 駅には乗換え駅(ゲートウェイ)と普通の駅がある
- 各路線(ネットワーク)は自律運用、かつ、相互協調
- 選択可能な経路、経路の選択は知的な作業
- 鉄道網の「オーナー」はいない

乗り換え駅

- 鉄道の路線と路線が交わるところ
  - 例：藤沢駅
    - ＪＲ東海道線
    - 小田急江ノ島線
    - 江ノ島電鉄
- 客の行き先で路線を選ばなくてはならない
  - 例：湘南台から品川
    - 湘南台：藤沢に行く
    - 藤沢：東海道線に乗り換え

経路制御とは

- だれが？　—　乗換駅（ルータ）
- 何を？- 最適な路線（経路）
  - 行き先に応じて路線を判断
  - 最適な路線を選ぶ
    - 事故の迂回・近い経路・速い経路
- どうやって？
  - 経路表を作成
  - 経路に関する情報を交換し経路表を更新
    - RIP
    - OSPF

経路制御

- どの路線（経路）に乗せればいいか？
  - 目的のホストがどの経路に接続してるか？
- 経路制御表
  - 経路の選択、制御に用いる
  - どの路線にはどのホストが接続しているという情報（経路情報）をまとめた表
  - すべてのホストはまとめきれない
    - IPアドレスのネットワーク部
    - default という経路

経路表の例

```
nr60: {2} % netstat -rn
Routing tables
Internet:
Destination          Gateway          Flags    Refs     Use  Interface
default              203.178.140.1     UG       7    35972  ef1
127                  127.0.0.1         UR       0        0  lo0
127.0.0.1            127.0.0.1         UH       0        0  lo0
133.27.12.129        203.178.140.1     UGHc     1      120  ef1
133.27.171/24        203.178.140.1     UG       0        0  ef1
202.0.73             203.178.140.1     UG       0        0  ef1
202.0.73.96/27       203.178.140.1     UG       0        0  ef1
202.0.73.128/27      203.178.140.1     UG       0        0  ef1
202.0.73.236/30      203.178.140.1     UG       0        0  ef1
203.178.138.18/30    203.178.141.9     UG       0        0  ef0
203.178.139.64/27     203.178.140.1    UG       0        0  ef1
203.178.139.96/27     203.178.140.1    UG       0        0  ef1
203.178.139.128/27   203.178.140.1     UG       0        0  ef1
```

RIP
Routing Information Protocol

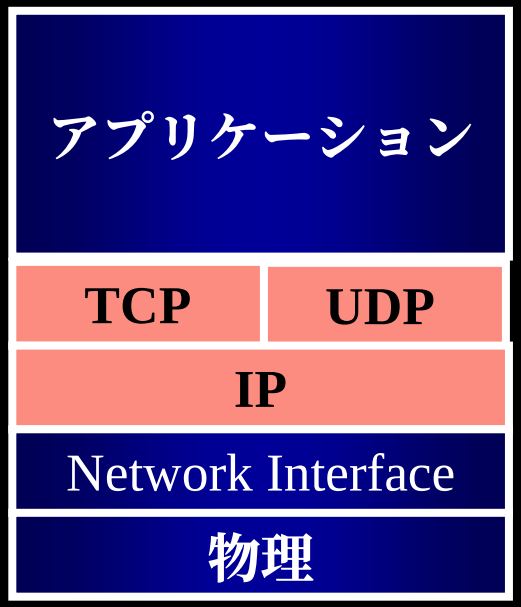
- 送信元と宛先との間で最適なルートを探す
- 送信元と宛先間の通過しなければならないネットワークの数を単純にカウントする.
- ホップ数の一番短いルートが最適ルート
  - 遅延, 不可, 信頼などは考慮されていない.
- 最適経路の計算には簡単なコスト計算 (三角不等式) をもちいる.

OSPF
Open Shortest Path Find

- サイト内のルータは同じデータベースを共有する.
- この情報を元に最短パスツリー(Shortest Path Tree)を形成する.
- ネットワークの変化に柔軟に対応できる

トランスポート層

- マシンまで到達したデータをアプリケーションと結び付ける
- TCP
- UDP



ポート番号

- ネットワーク層によるホストーホスト間のデータ配送
- 同一ホスト内でのアプリケーションの識別
  - 一つのホスト内でアプリケーションごとに割り振りが必要
- アプリケーションの識別子
  - プロトコル(TCP･UDP)
  - ポート番号(プロトコルに固有)

99/02/14 41

ポート番号の例

www.sfc.wide.ad.jpのport 番号80番に接続

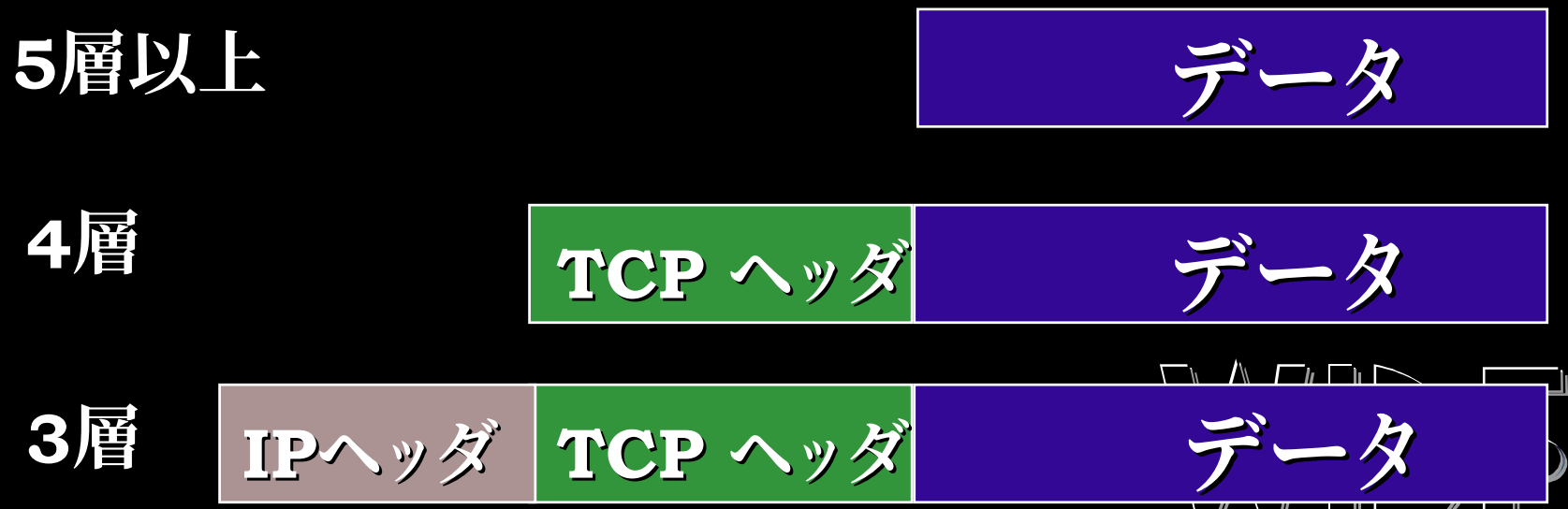
```
% telnet  www.sfc.wide.ad.jp  80
Trying 203.178.140.3 ...
Connected to enterprise.sfc.wide.ad.jp.
Escape character is '^]'.
GET /index.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Tokuda, Murai, Kusumoto &amp;amp; Nakamura Laboratory</TITLE>
<LINK REV=MADE HREF="mailto:www-admin@sfc.wide.ad.jp">
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-
2022-JP">
</HEAD>
<BODY BGCOLOR="#D6AF85">
```

/etc/services

```
#
# @(#)services 1.16 90/01/03 SMI
tcpmux          1/tcp       # rfc-1078
echo            7/tcp
echo            7/udp
discard         9/tcp       sink null
discard         9/udp       sink null
systat          11/tcp      users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp      ttytst source
chargen         19/udp      ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp      mail
time            37/tcp      timserver
time            37/udp      timserver
```

ネットワーク層とトランスポート層

- トランスポート層でもヘッダを付ける
- ネットワーク層ではトランスポート層　ヘッダがついたデータにネットワーク層のヘッダを付加する



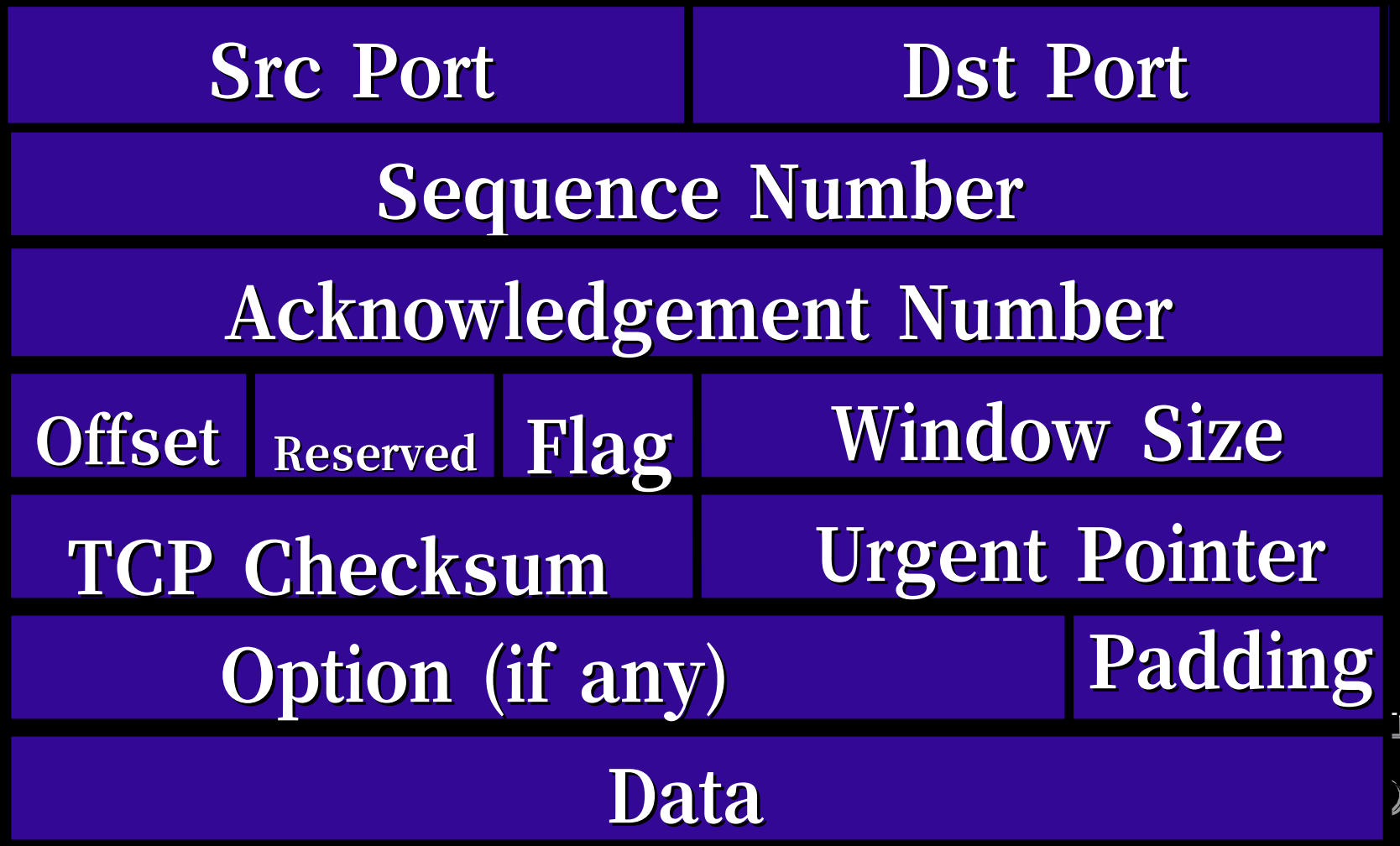
UDP
User Datagram Protocol

- Datagram型通信の提供
  - IPはDatagram型の通信
    - Datagram型通信のための新たな機能は必要無い
  - トランスポート層はアプリケーションを識別
    - 識別子があればOK

TCP
Transmission Control Protocol

- データ転送に関する制御を行う
- 具体的には.....
  - IPに信頼性を加える
    - Virtual Circuit型通信
    - エラー検出とエラー訂正
    - フロー制御
    - 順序の再構成
  - データ転送に関するインターフェイスを提供

TCPヘッダ

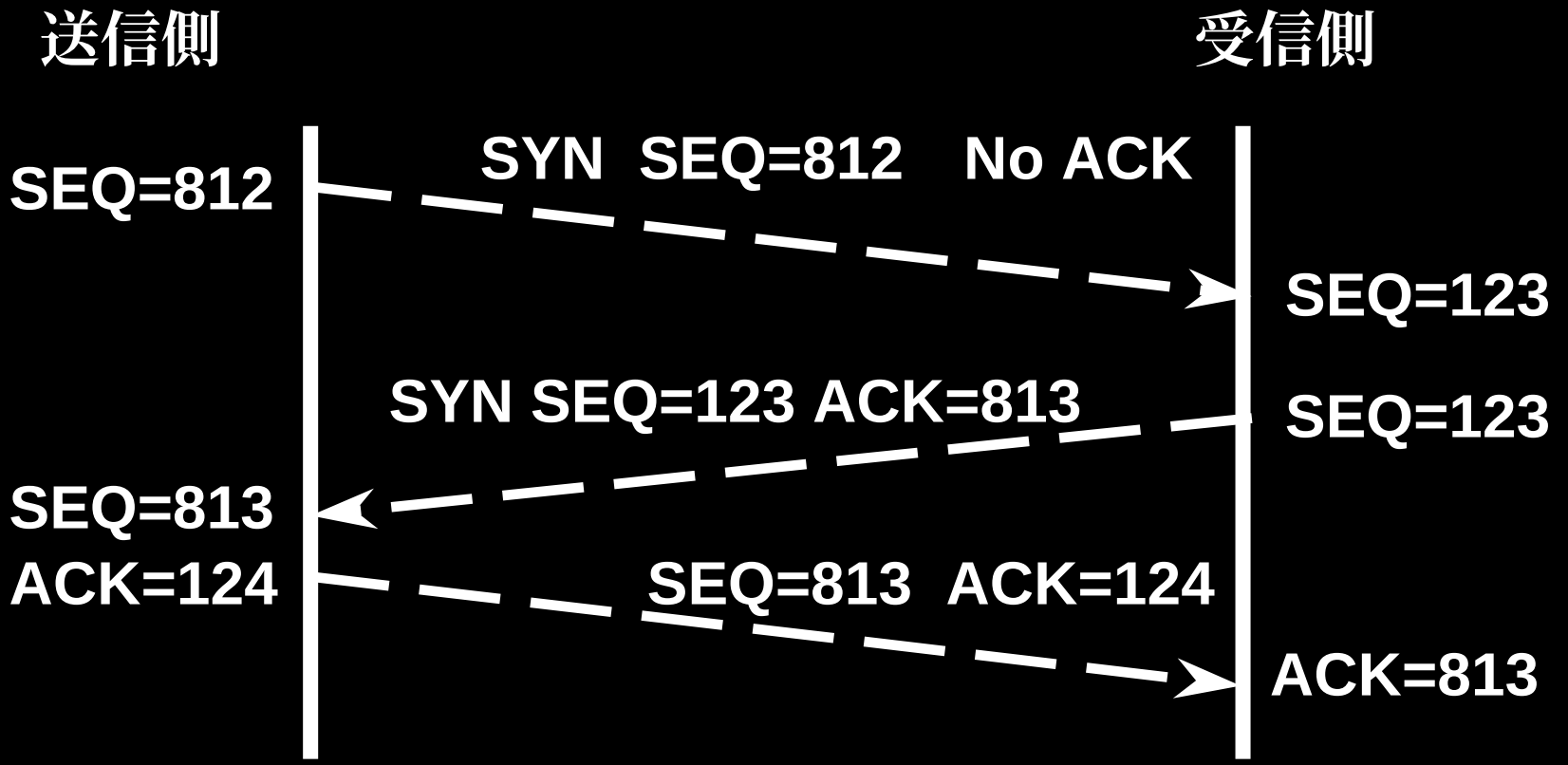

Src Port
Dst Port
Sequence Number
Acknowledgement Number
Offset
Reserved
Flag
Window Size
TCP Checksum
Urgent Pointer
Option (if any)
Padding
Data

Flags CODE

- FlagsはTCPがコネクション制御に利用
- 6つのflagsがある
  - URG： Urgent Pointerが有効
  - ACK： Acknowledge Numberが有効
  - PSH： セグメントをすぐに上位層へ渡す
  - RST： エラーによる強制的なクローズ
  - SYN： コネクションセットアップの同期をとる
  - FIN： コネクションを終了する

TCPの状態遷移図

コネクション開始


送信側
受信側
SEQ=812
SYN SEQ=812 No ACK
SEQ=123
SYN SEQ=123 ACK=813
SEQ=123
SEQ=813
ACK=124
SEQ=813 ACK=124
ACK=813

3way Handshake

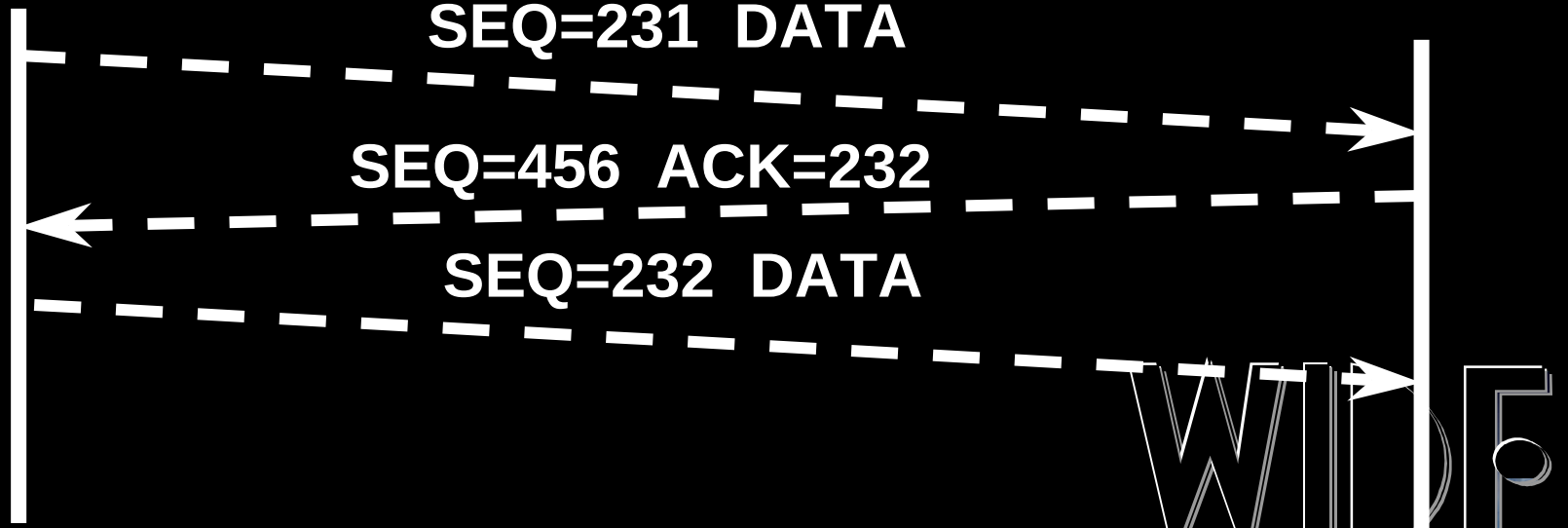
- コネクションのセットアップ
  - ホスト1、ホスト2間で
    - ホスト1はSYN Flagのついたパケットを送る
    - SYNを受けたホスト2は相手との同期を取るためにSYN Flagのついたパケットを送る。この時いっしょに受け取ったSYNに対するACK Flagもつける
    - ホスト2からSYNを受け取ったホスト1はACKを返す
  - SYN と ACKが相乗り
    - Piggy Back

コネクションの終了

- コネクションの終了はFIN Flagによって行う
- コネクションの終了は一方ごとにできる

データの転送

- Sequence NumberとAcknowledge Numberによって、データの順番を保証
- ACKが帰ってきたら次のデータを送る
- ACKが帰ってこなかったら前のデータを再転送



データ転送（続き）

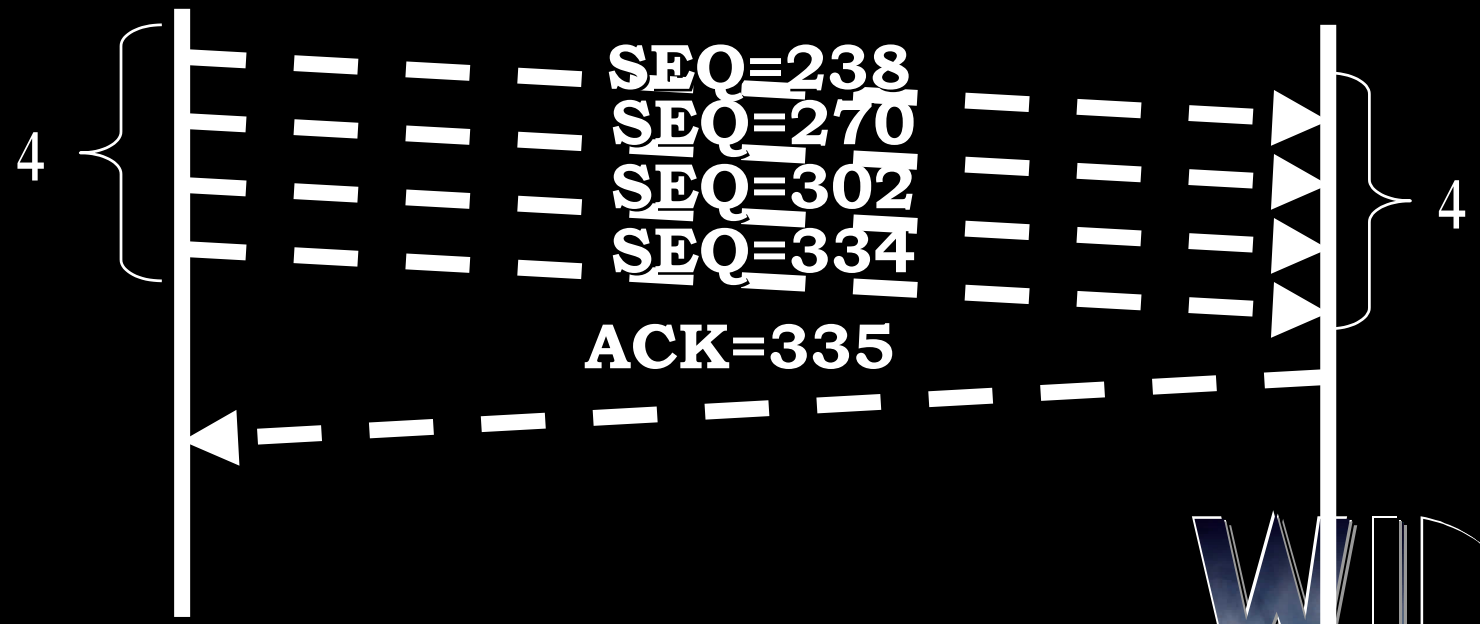
- いちいちACKを待つと効率が悪い
- 推測を元にある程度先送りした方が良い
- だが、先送りしすぎるとACKがなかなか帰ってこない
- 幅を決めなければならない

フロー制御

- フロー制御とは？
  - 相手の受信能力にあわせた送信
  - ネットワークの状態にあわせた送信
- フロー制御のための機構
  - ウィンドウ方式
- ウィンドウサイズの決定
  - 受信側が自分の能力にあわせて決定
  - 送信側がネットワークの状態を判断して決定
  - 最適値 = 通信経路の帯域 * RTT

ウィンドウコントロール

- 先送りする幅を決める仕組み
- 送るパケットを制限する



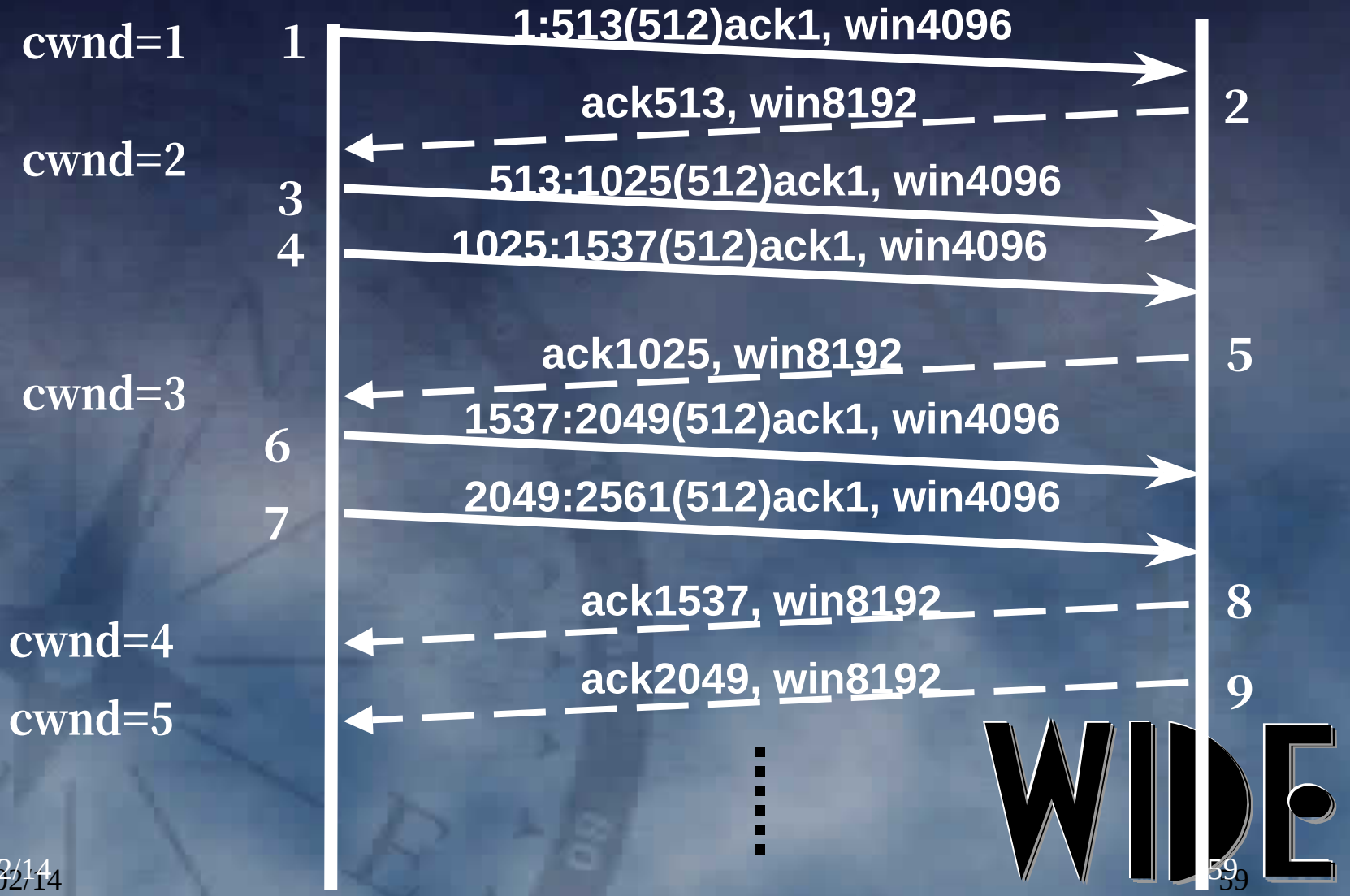
ウィンドウサイズ制御

- 通信開始時のウィンドウサイズ制御
  - スロースタート
  - ACK毎にサイズを +1
- 輻輳回避時のウィンドウサイズ制御
  - ACK毎ににサイズを “1/輻輳ウィンドウ“づつ増加

スロー・スタート

- ネットワークに送り出されるパケットの通信速度を他方のエンドから返される確認応答の通信速度を同期させるためのもの
- 送り手のTCPに別のウィンドウを追加する。
  - 輻輳ウィンドウ(cwnd)
- 新しいコネクションが確立
- 輻輳ウィンドウをセグメント1つに初期化
- ACKが受け取られるたびに、輻輳ウィンドウは1セグメントずつ増やされる

スロー・スタート animation



TCP-輻輳制御

- 輻輳とは？
  - 中間ノードにキューができ通信に遅延が生じた状態
  - 判断基準
    - パケット喪失が発生
- 輻輳回避のストラテジ
  - 輻輳を検出したらウィンドウサイズを小さく
  - ネットワークに負荷をあたえずすみやかに最適なウィンドウサイズへ回復

輻輳への対応策

- 喪失パケットの再送ストラテジ
  - Fast Retransmit
  - 1パケット喪失／1ウインドウの場合に適用
- 輻輳後の転送レートの回復ストラテジ
  - Fast Recovery
  - 輻輳から速やかに回復する

99/02/14 61

Fast Retransmit

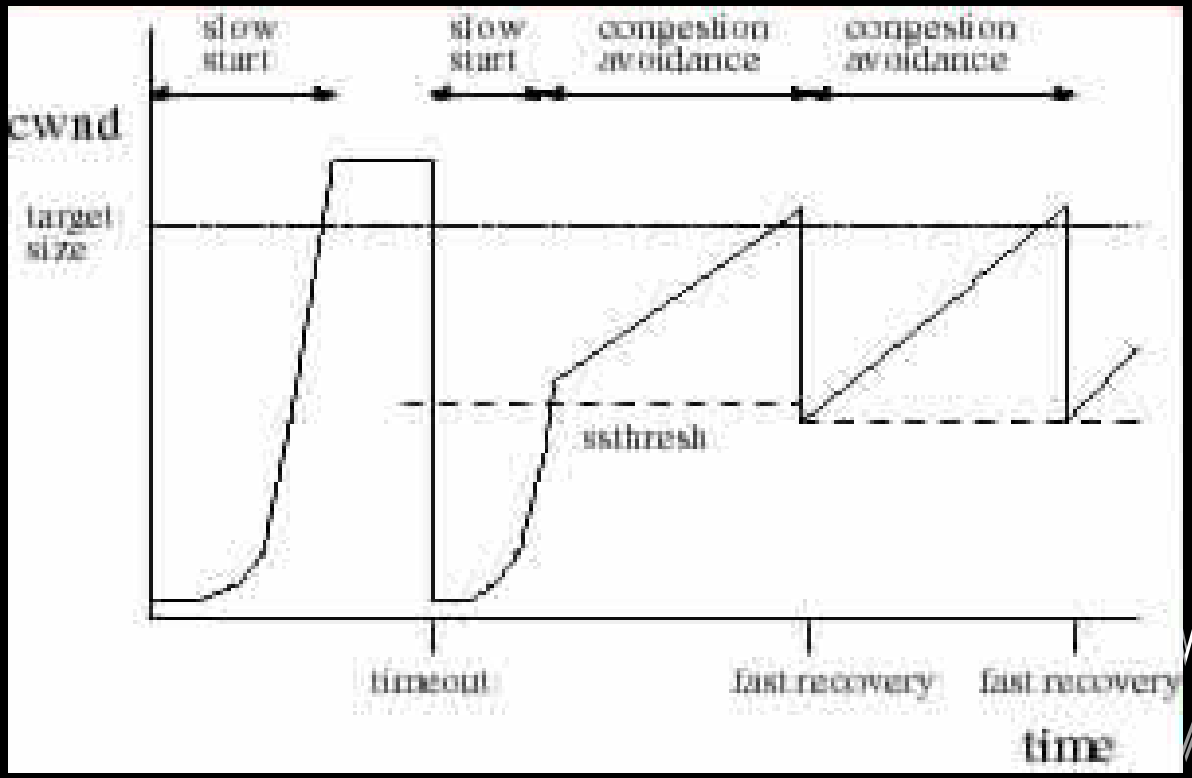
- DUP ACKが3つ来た場合に適用
  - すぐにパケットを再送する
- さらに DUP ACKがくる場合
  - 1パケット喪失した後に、さらにパケットが到着していると推測
    - 輻輳ウインドウを一旦増加
    - 新しいセグメントを送出する
- 新しいACKがきたら
  - 輻輳の終了とみなす
    - Fast Retransmitの成功

Fast Recovery

- Fast Retransmitが成功した場合
  - 輻輳は深刻なものではなかったと推測
  - 速やかにウインドウを回復
  - スロースタート段階に入らず、いきなり輻輳回避段階へ

TCP通信モデル

- ウインドウサイズの変化
- スロースタートを経て平衡状態へ
- 平衡状態ー一定間隔でパケットロス



インターネットの運用技術

運用技術

- DNS
- DHCP
- 品質保証
- セキュリティ技術

名前とIPアドレス

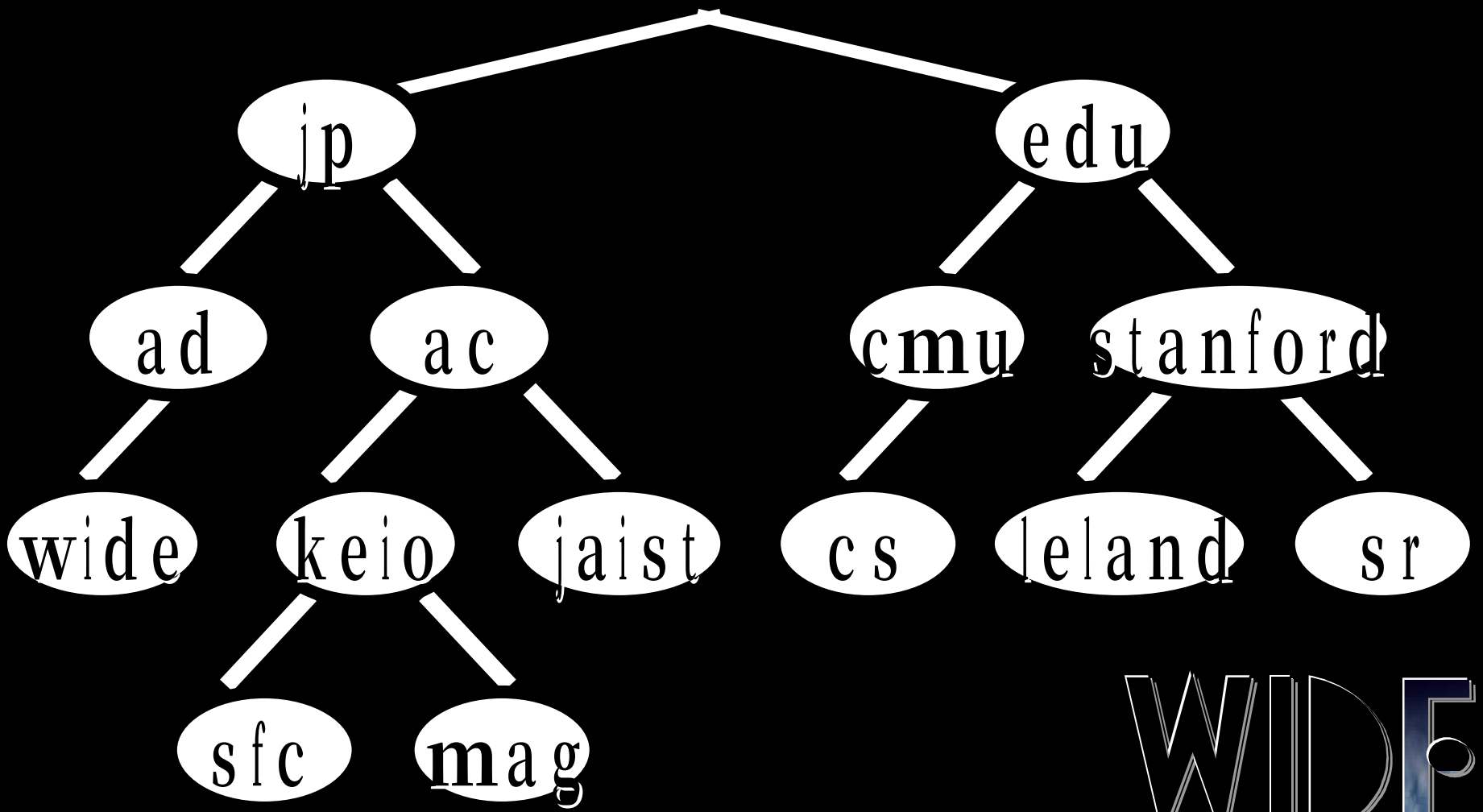
- コンピュータは数字を扱う方が得意
  - コンピュータはアドレスが分ればOK
- 人間は数字を扱うのは得意ではない
  - 数字より名前の方が扱いやすい
- 名前
  - ホストの名前（例：mail0 ）
  - 組織の名前　（例：sfc.keio.ac.jp)

WIDE

DNS
Domain Name System

- ホスト名 (例: mail0.sfc.keio.ac.jp) とIPアドレス(例: 133.200.113.5) のマッピングを管理
  - ホスト名とIPアドレスの分散データベース
  - 世界中の全ホストに関する情報を持つ
  - 世界住のどこからでも検索可能
  - 階層化されたマスター情報の管理
  - サーバ:データの形式と答え方
  - クライアント:問い合わせ先の情報と問い合わせ方

DNSの構造



WIDE

DHCP

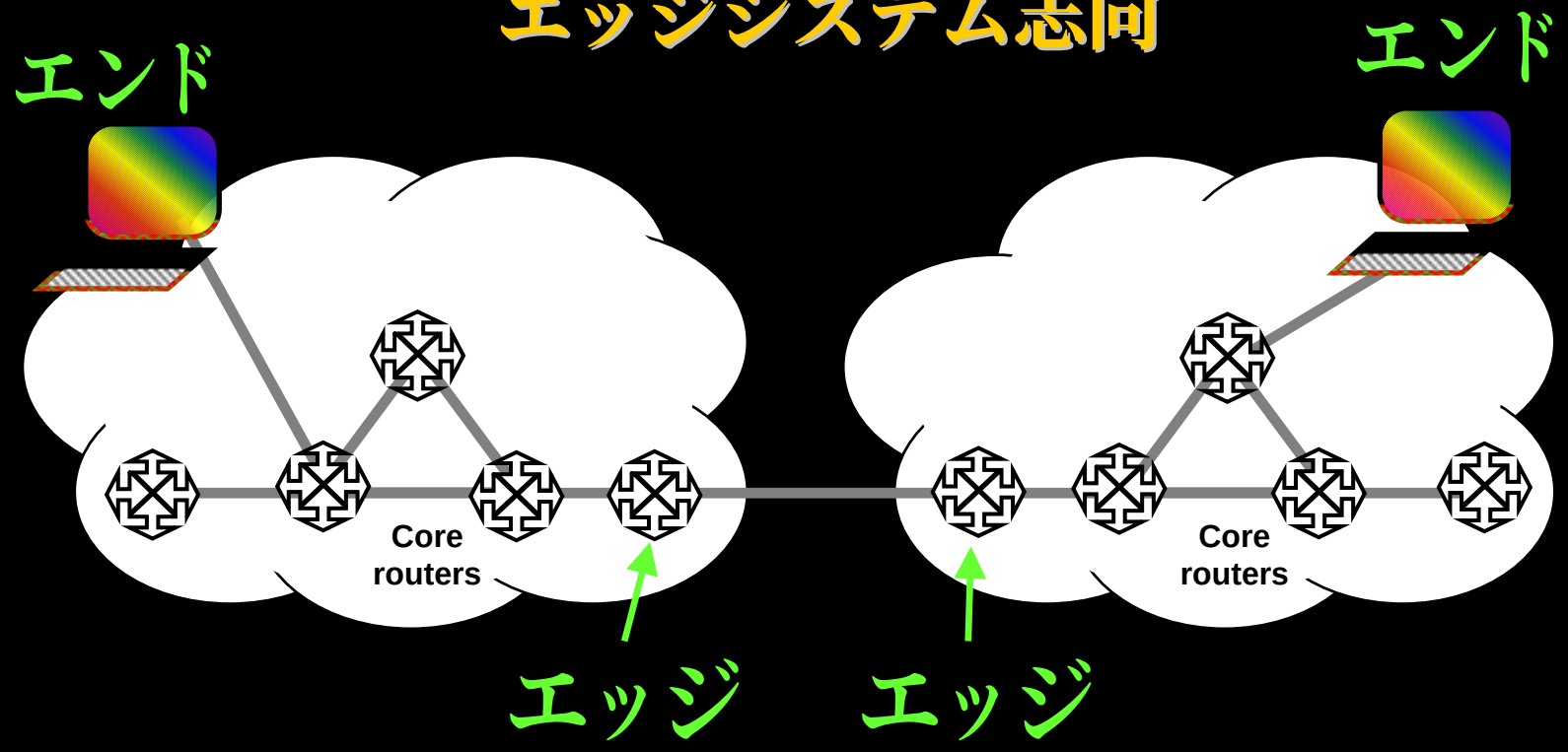
- Dynamic Host Configuration Protocol
- RFC2131
- 主に動的に接続するホストを自動構成するためのプロトコル

QoS
Quality of Service

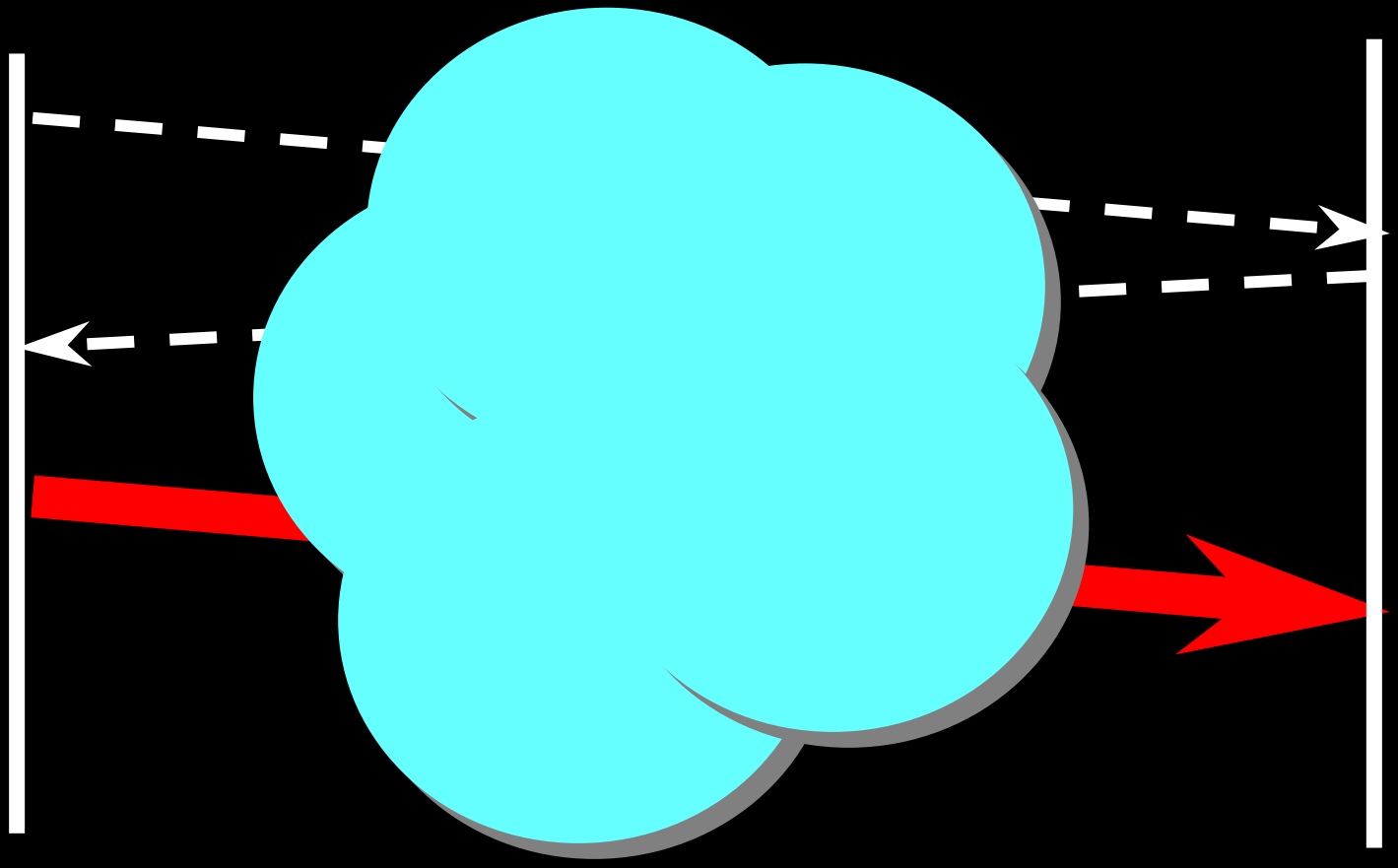
- Quality
  - 到達性の保証
  - 帯域の保証
  - 遅延の保証
- 優先度に応じた制御

知性は外側へ

エンドシステム志向
エッジシステム志向

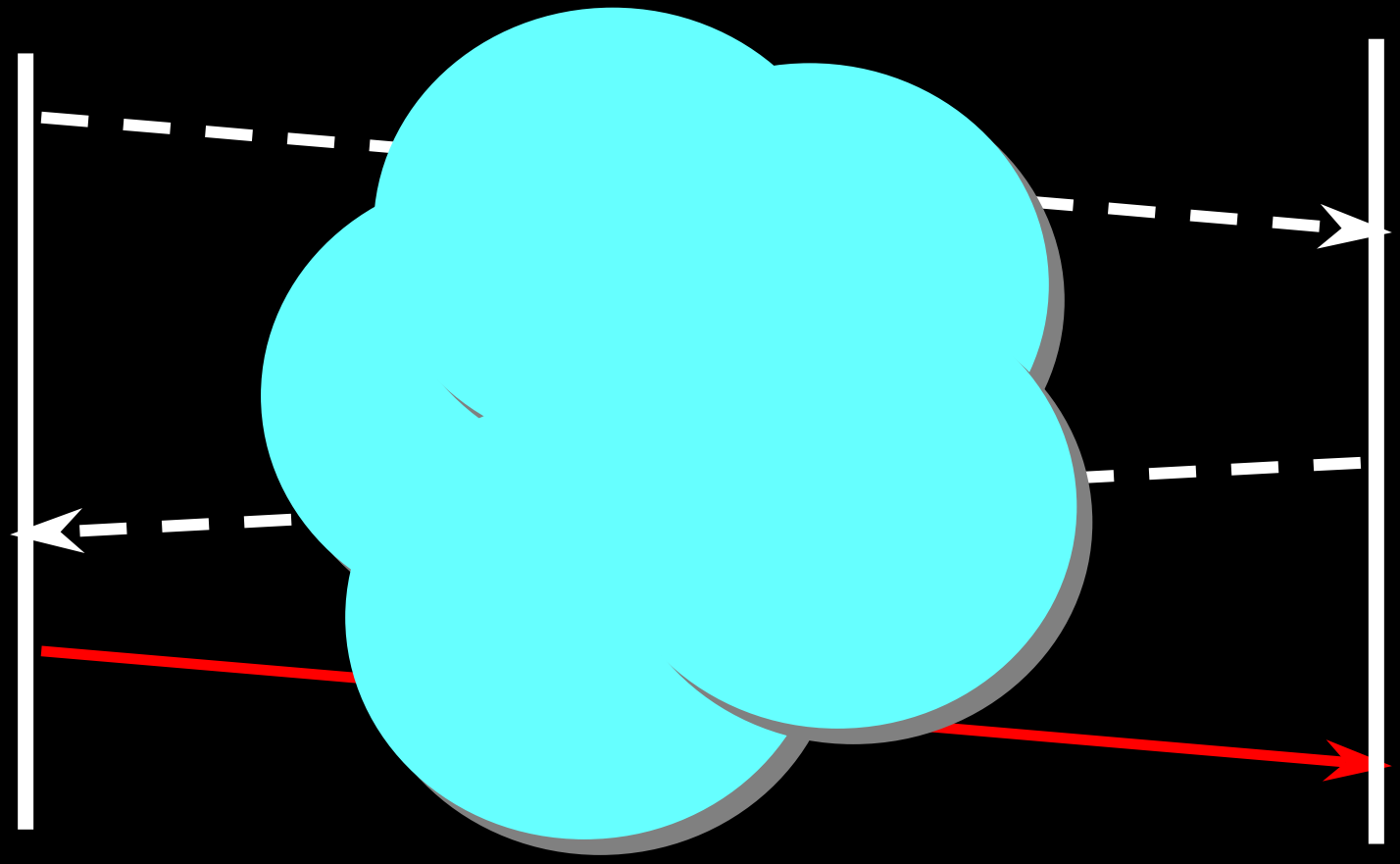


輻輳制御ルール１：返事で判断



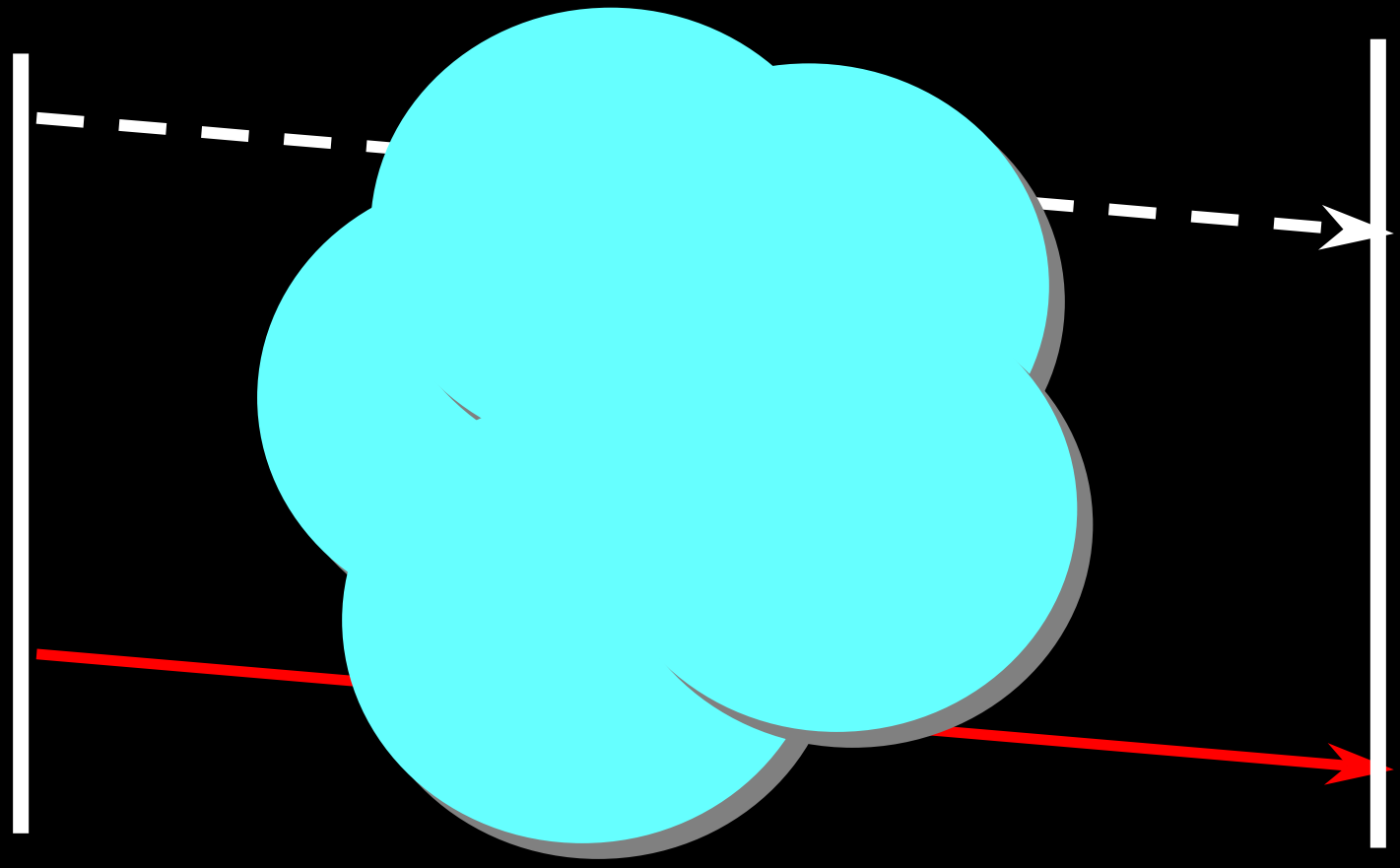
返事が早い→空いてる→沢山送る

輻輳制御ルール2:混んだら小さく



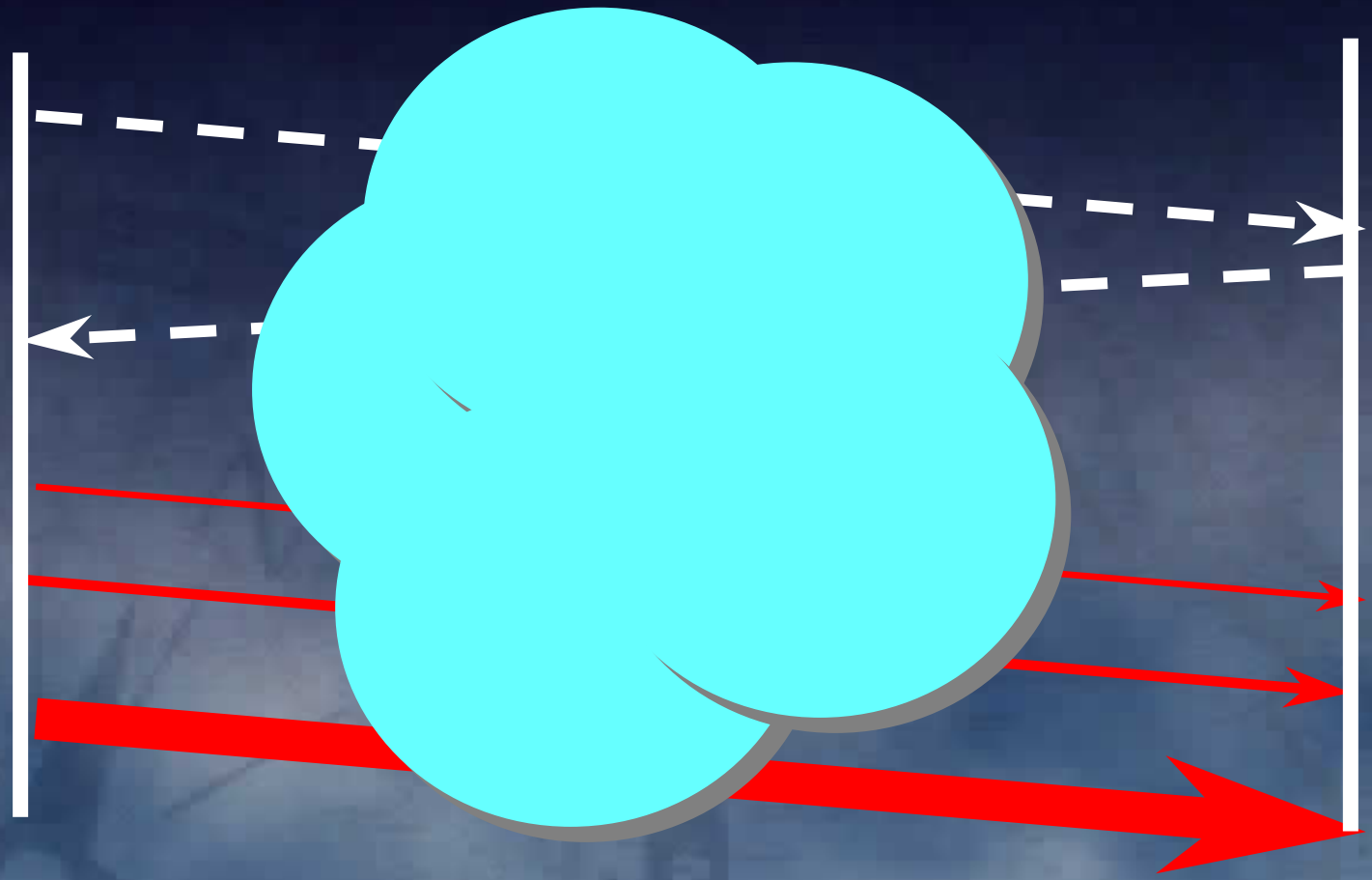
返事が遅い→混んでる→少し送る

輻輳制御ルール2:混んだら小さく



返事が無い→混んでる→少し送る

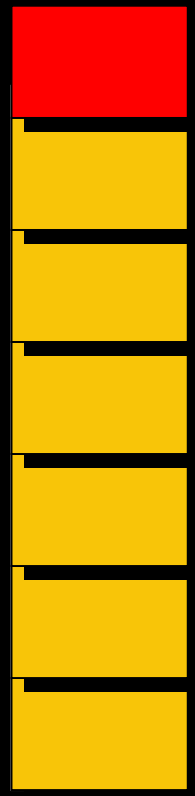
輻輳制御ルール3:空いても急ぐな



返事が早い→空いてる→ゆっくり送る

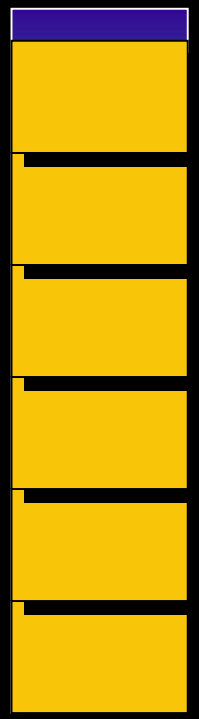
インターネットが遅い原因

- 返事が遅い
  - 相手がのろい
  - 途中がのろい
- 相手がのろい場合
  - 相手が過負荷
    - アクセスが多すぎる
    - 力がなさ過ぎる
- 途中がのろい場合
  - どこかの待ち行列であふれている



インターネットが遅い原因

- 返事が遅い
  - 相手がのろい
  - 途中がのろい
- 相手がのろい場合
  - 相手が過負荷
    - アクセスが多すぎる
    - 力がなさ過ぎる
- 途中がのろい場合
  - どこかの待ち行列であふれている

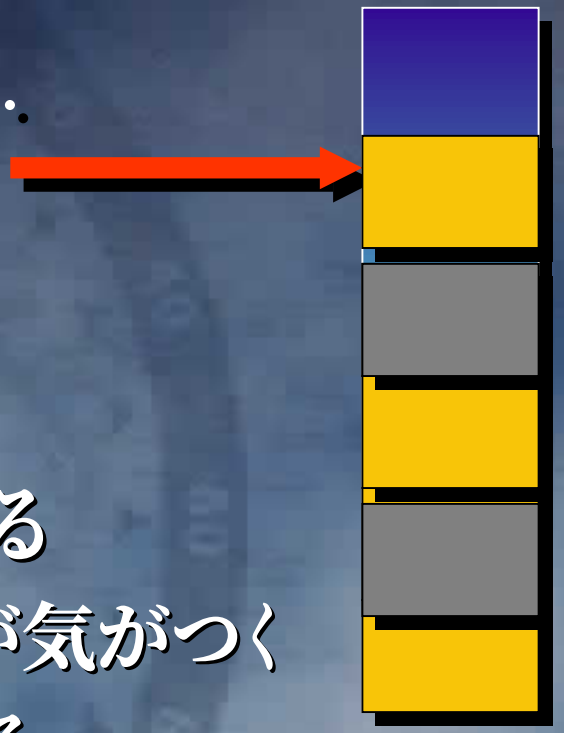


Differentiated Service

- 今までのインターネット
  - ベストエフォート
    - 届くように努力する
    - コアシステムに負荷が無い
- サービス保証システム
  - 契約したサービスを保証する
  - コアシステムの負荷が大きい
- これからのインターネット
  - ベストエフォート＋DiffServe
  - コアシステムに負荷をかけずに優先度のあるサービスを提供する

Random Early Detection (RED)
Explicit Congestion Notification (ECN)

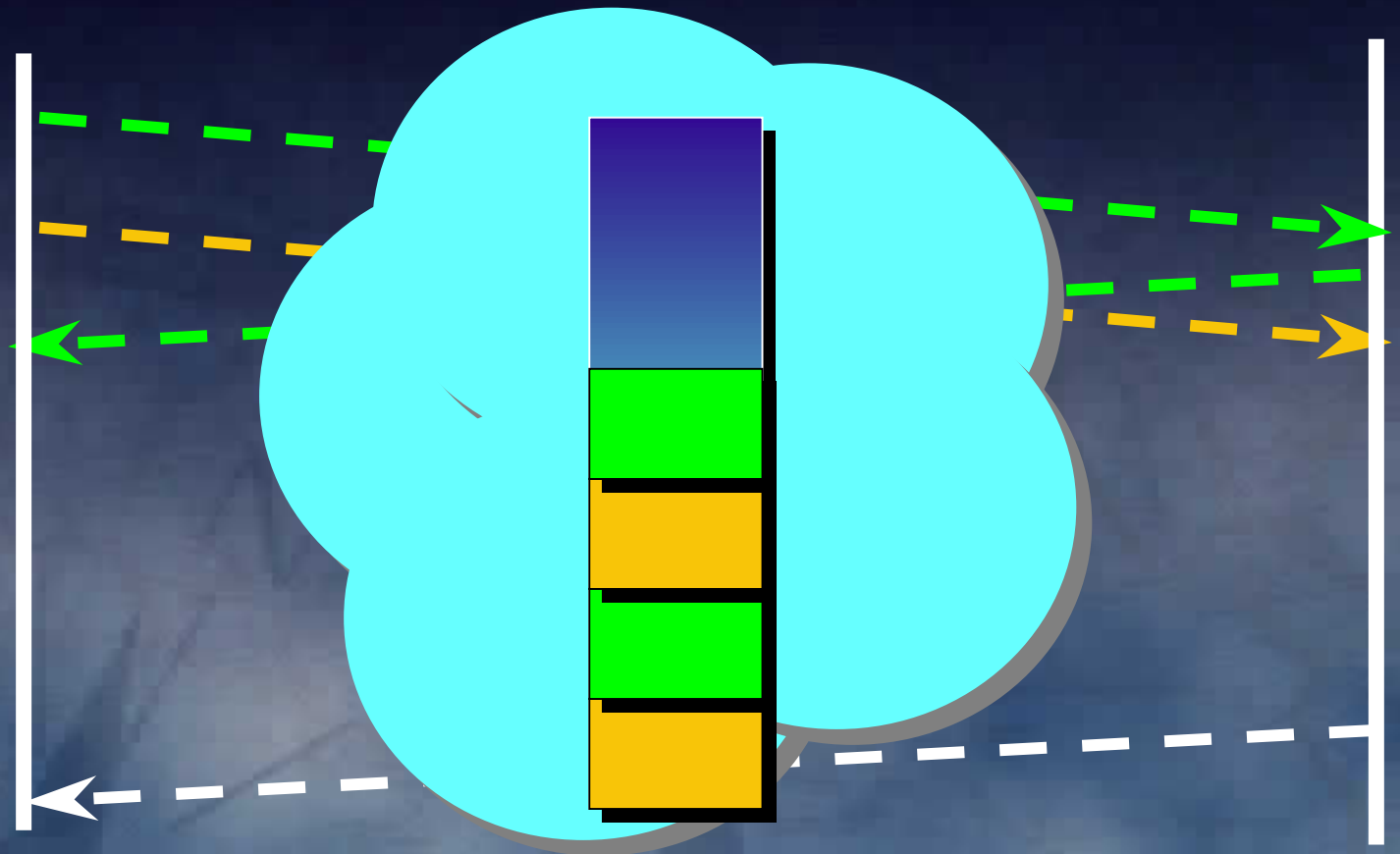
ここまでつまったら‥



- いくつか捨てる
- いくつかマークする
  - エンドシステムが気がつく
  - ペースダウンする

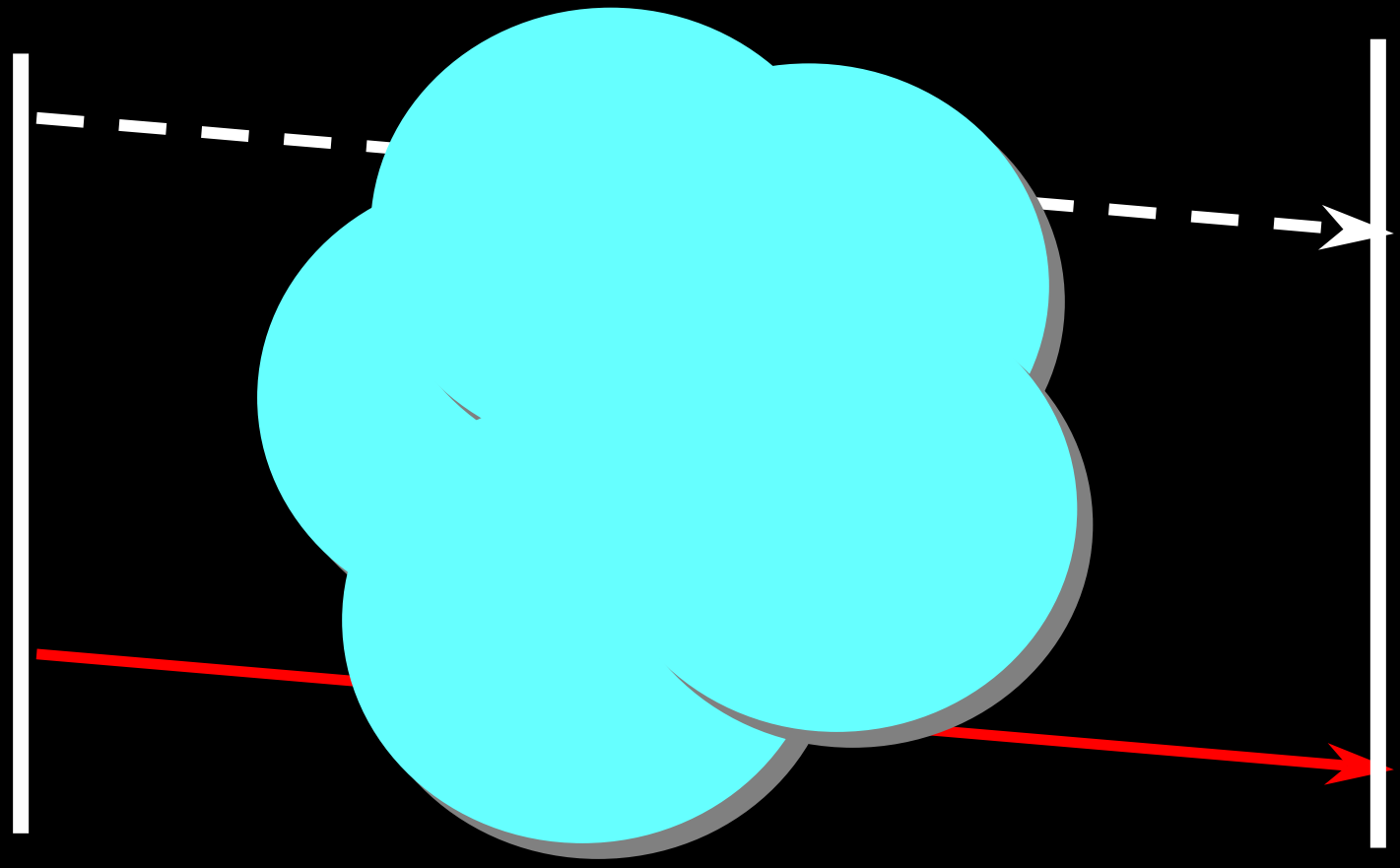
99/02/14

プライオリティサービス



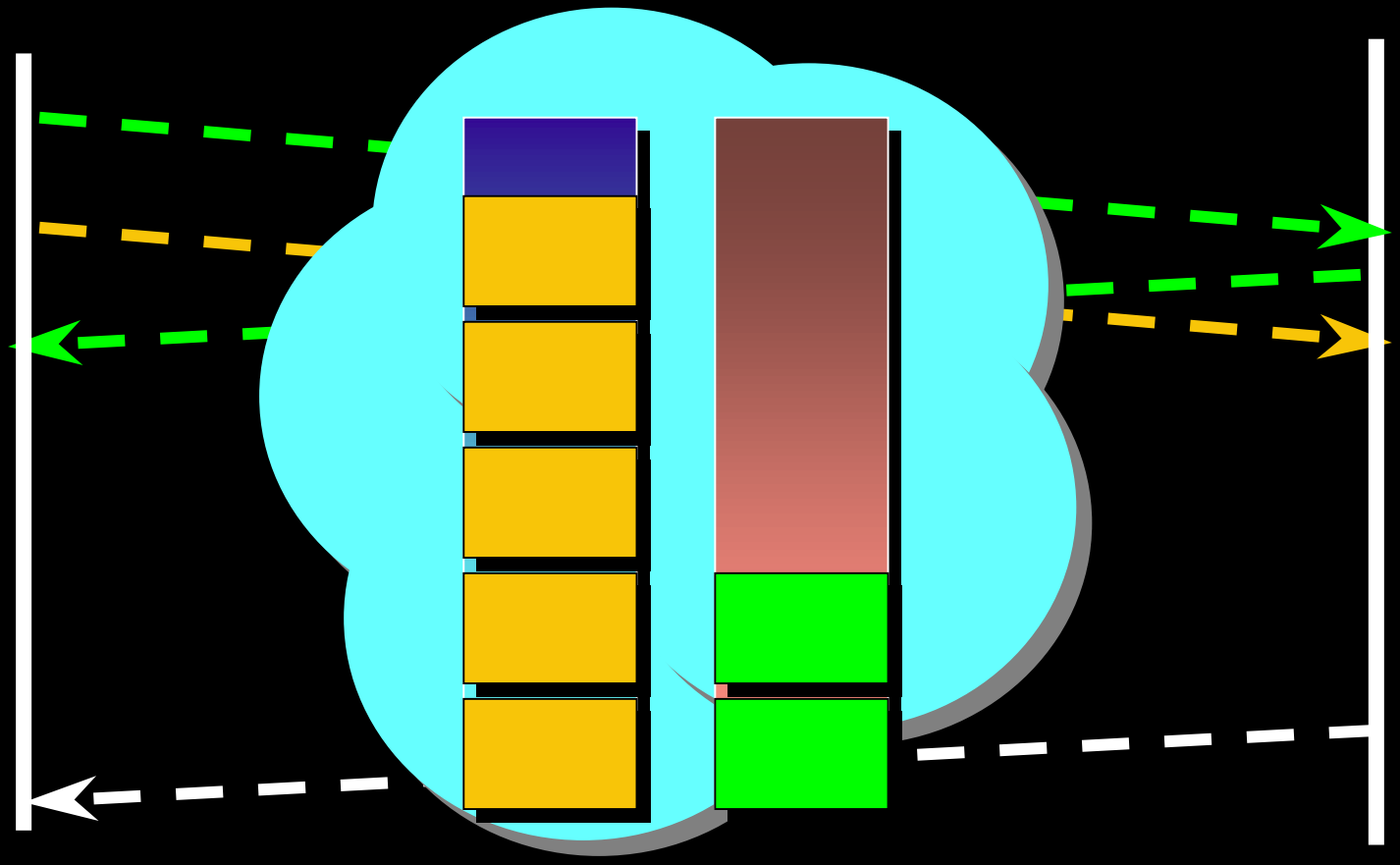
優先度に応じて捨てちゃえ!

輻輳制御ルール2:混んだら小さく



返事が無い→混んでる→少し送る

プレミアムサービス



優先度に応じた別の待ち行列

RED
Random Early Detection

- ルータの負荷が増大したときに，優先度の低いトラフィックを意図的に遅らせることで輻輳の発生を軽減し、優先度の高いトラフィックの優先的到達を保証する技術
- 待ち行列の中にあるパケットを優先順位に基づきにマークもしくは落とす

セキュリティ

- 公開鍵暗号系を用いた認証に基づくセキュリティの確保

99/02/14 85

通信媒体とインターネットアーキテクチャ

通信媒体技術

- 広域系
  - 衛星
  - Cable TV
  - xDSL
  - SONET
  - WDM
  - ISDN
- LAN系
  - 10baseT
  - 100baseT
  - Gigabit
  - Wireless
  - HUB
  - Switch
  - FDDI

帯域が異なる媒体の混在

- フラグメントをさける技術
  - PathMTU
- ボトルネック
  - Packat Pair

遅延の大きく高速な通信媒体の混在

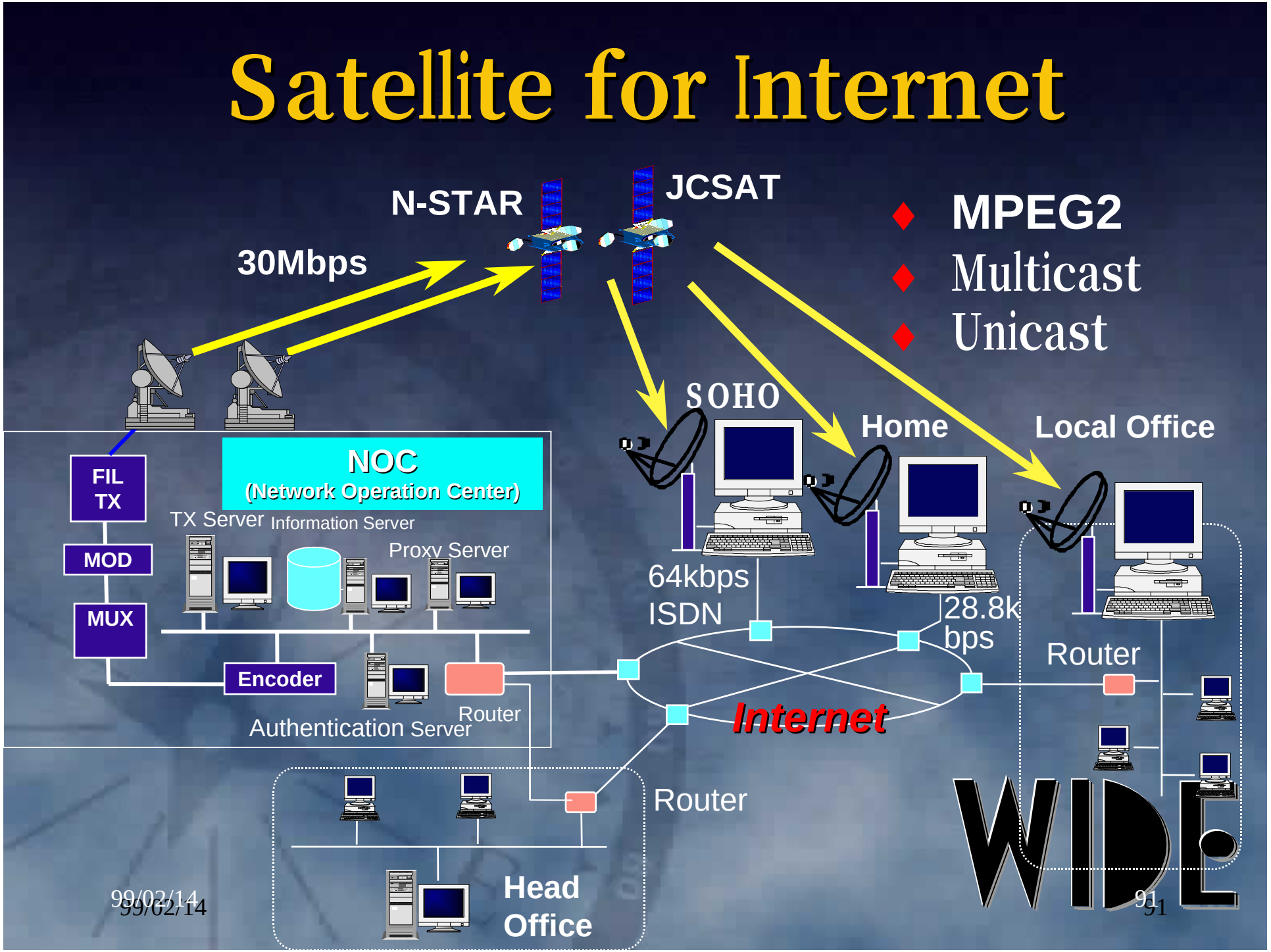
- 輻輳制御の副作用で利用効率が悪化

99/02/14 89

片方向の通信媒体の混在

- UDLRなどを用いた有効利用が必要

Satellite for Internet



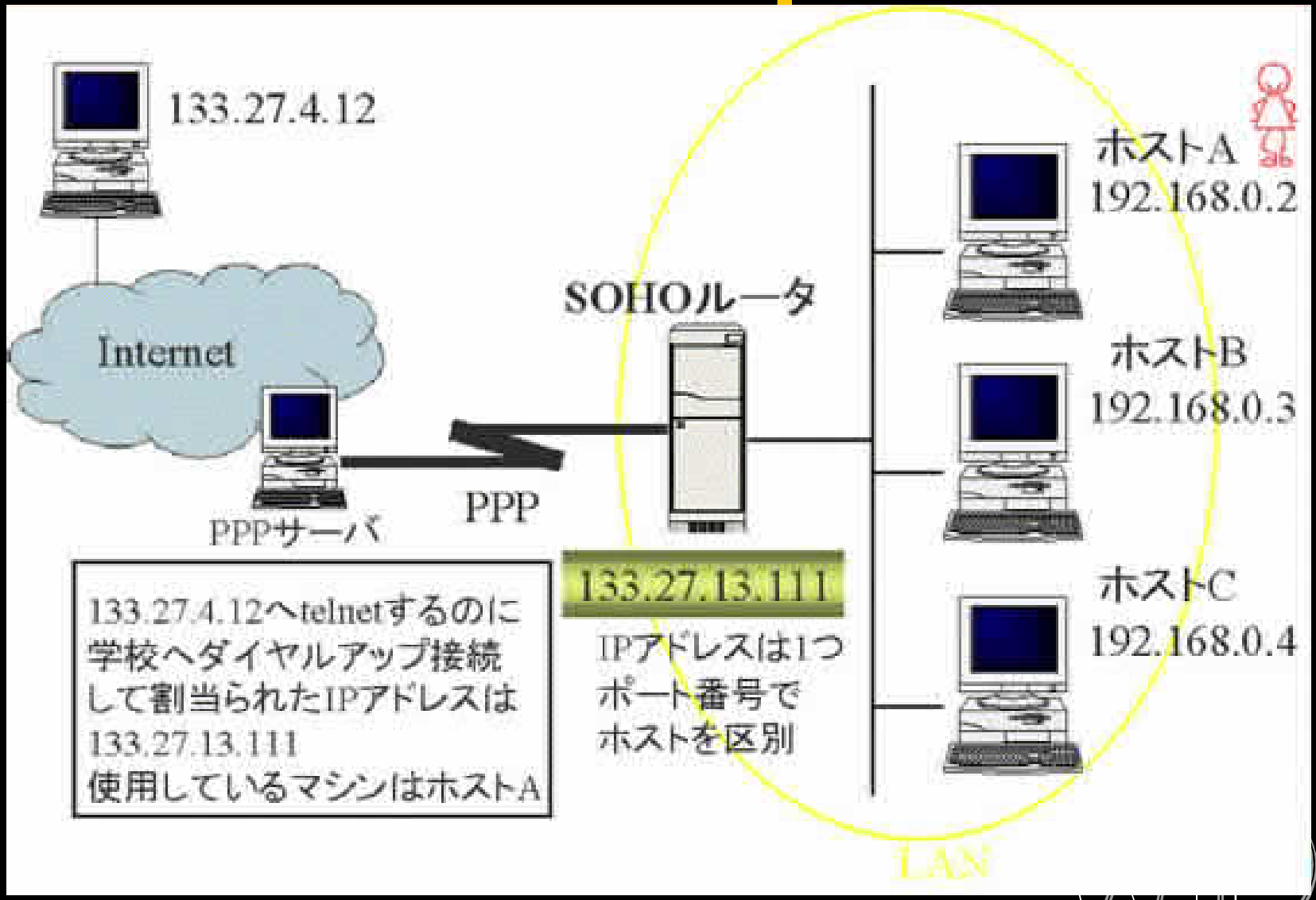
環境適応のための技術

適応のための技術

- SOHO
  - NAT/Masquerade
- モバイル
  - Mobile-IP
  - IP トンネル
- 非対称通信
  - UDLR
- Security
  - Firewall
  - Application Gateway
  - VPN
- アドレス枯渇
  - CIDR
- 経路情報の氾濫
  - IGP+BGP

本来アーキテクチャはどう対応すべきか？

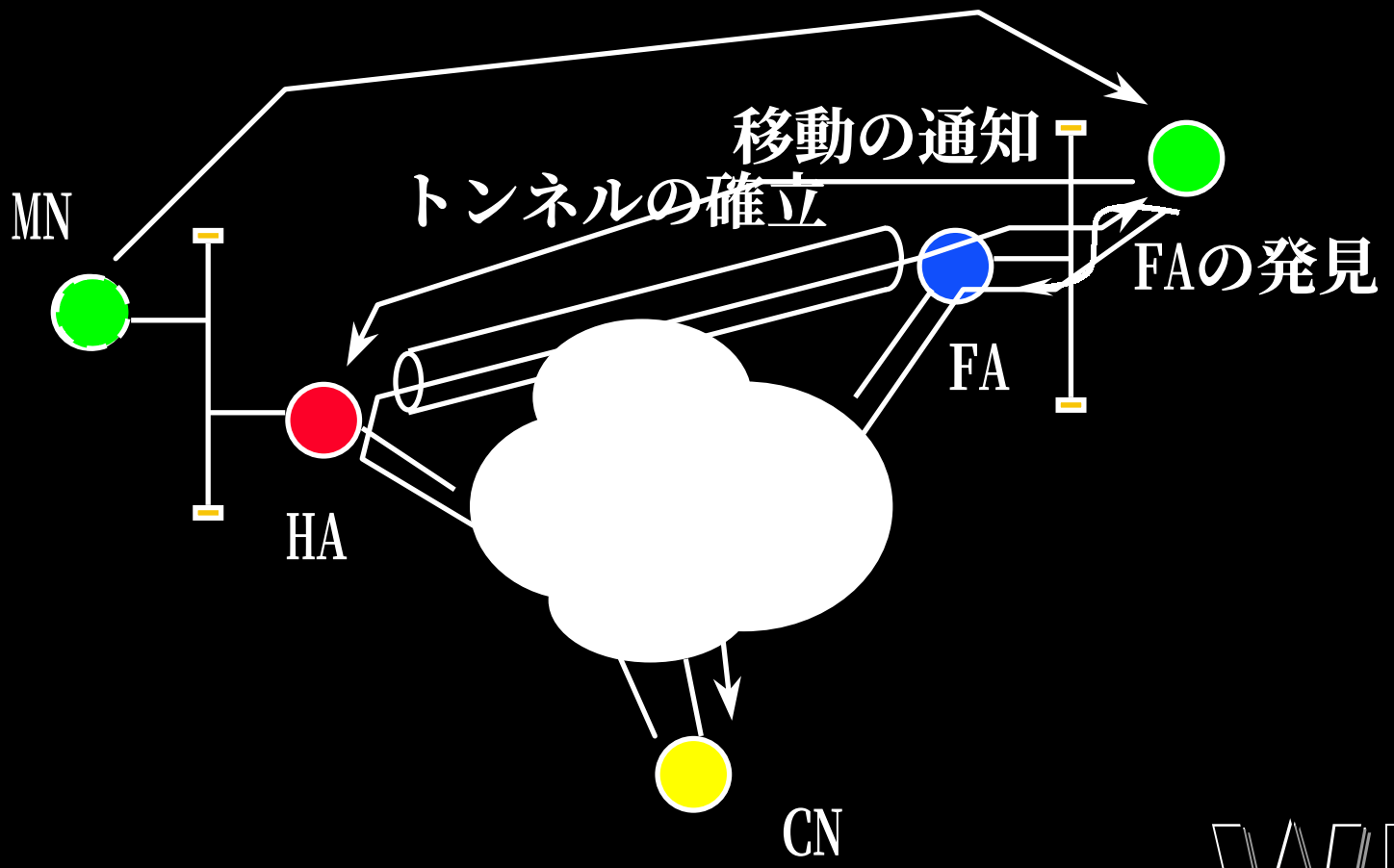
NAT・Masquerade



Mobile IP

- RFC2002
- Mobile Node:
  - 移動ノード
- Home Agent:
  - 移動ノードの位置を管理
- Foreign Agent:
  - 移動先での移動ノードの管理
- Care-of Address:
  - IP tunnelingの移動先での端点

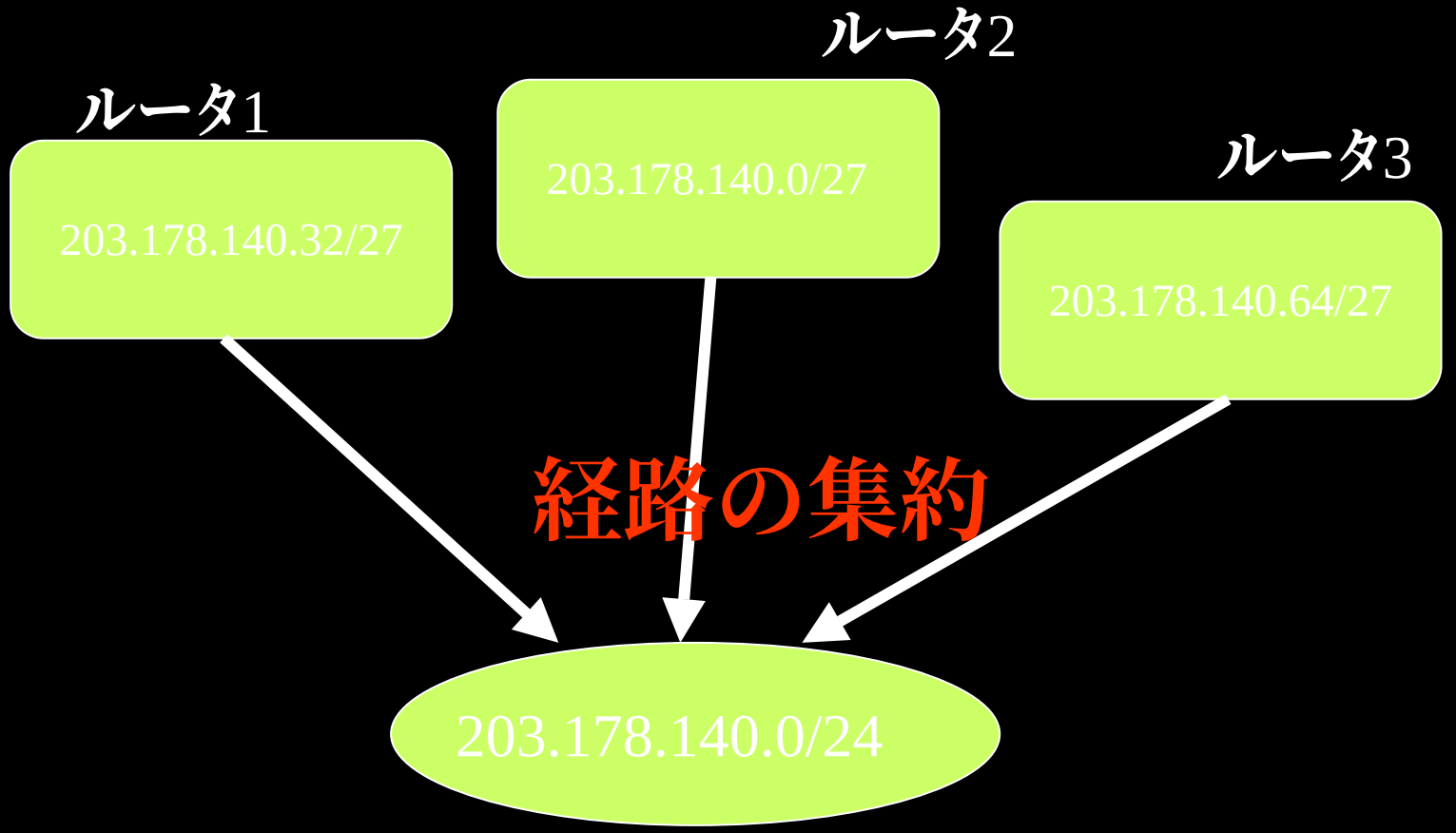
Mobile IP



経路の集約(aggregation)

- 各ノードが持つ経路情報はホストの数だけ存在する.
- ホストが増えるにつれて，経路情報も増大.
- このままでは破綻してしまうので，経路情報の階層化，経路情報の圧縮をおこなう.

経路の集約その２



集約した経路を流す

CIDR
Classless Internet Domain Routing

- 複数のIPアドレスをより少ない数のルーティングテーブルに集約する
- 例
  - インターネット上で異なる8個所に割り当てられているネットワークのアドレスを集約化して1つのルーティング・テーブル・エントリとして扱う

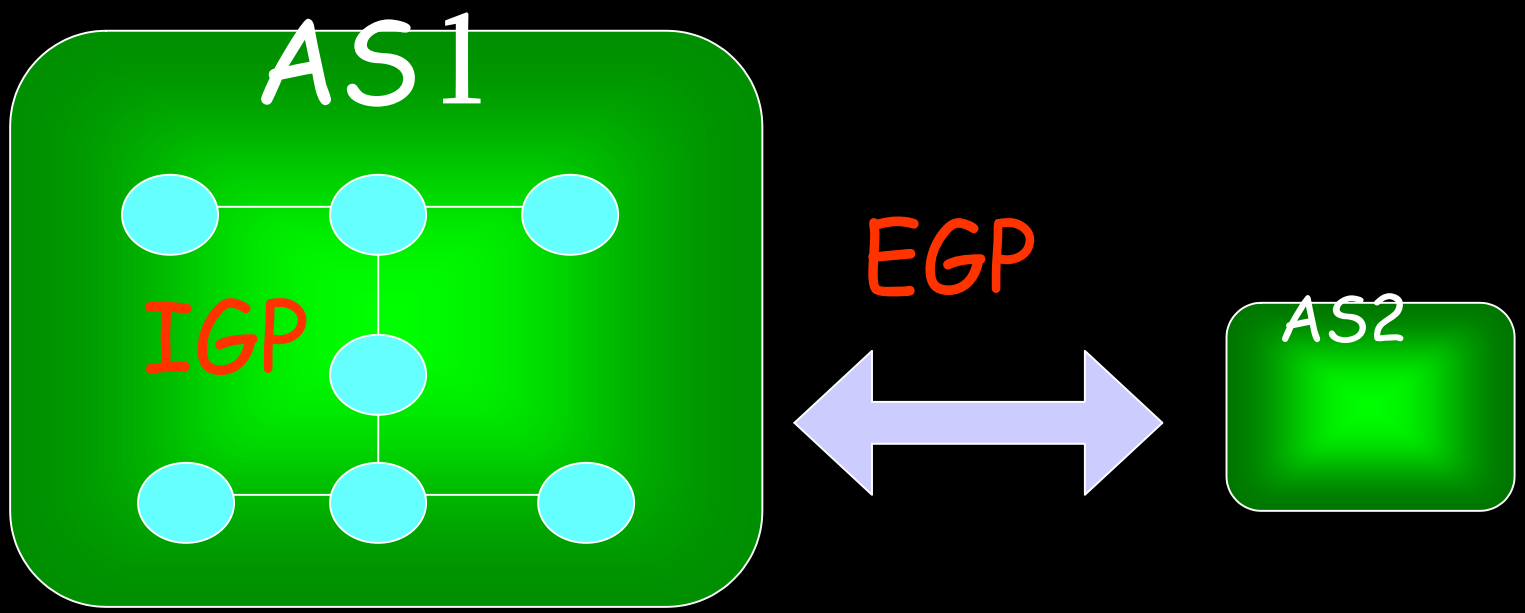
AS - Autonomous System

- さまざまなネットワークが相互接続している状況では通信を行う経路は複数存在
- どの経路を選ぶかは，ネットワークを運営する組織のポリシーによる.
- 経路情報を一定のグループにまとめて扱うことができるようにASという概念を導入.

IGPとEGP

- 経路制御プロトコルには，スケールする限界がある．
- おもにAS内でもちいるプロトコルをIGP(Interior Gateway Protocol)という．
- ASとASとの間の経路情報交換などを想定してるつくられている経路制御プロトコルをEGP(Exterior Gateway Protocol)

ASとルーティング



代表的なAS番号

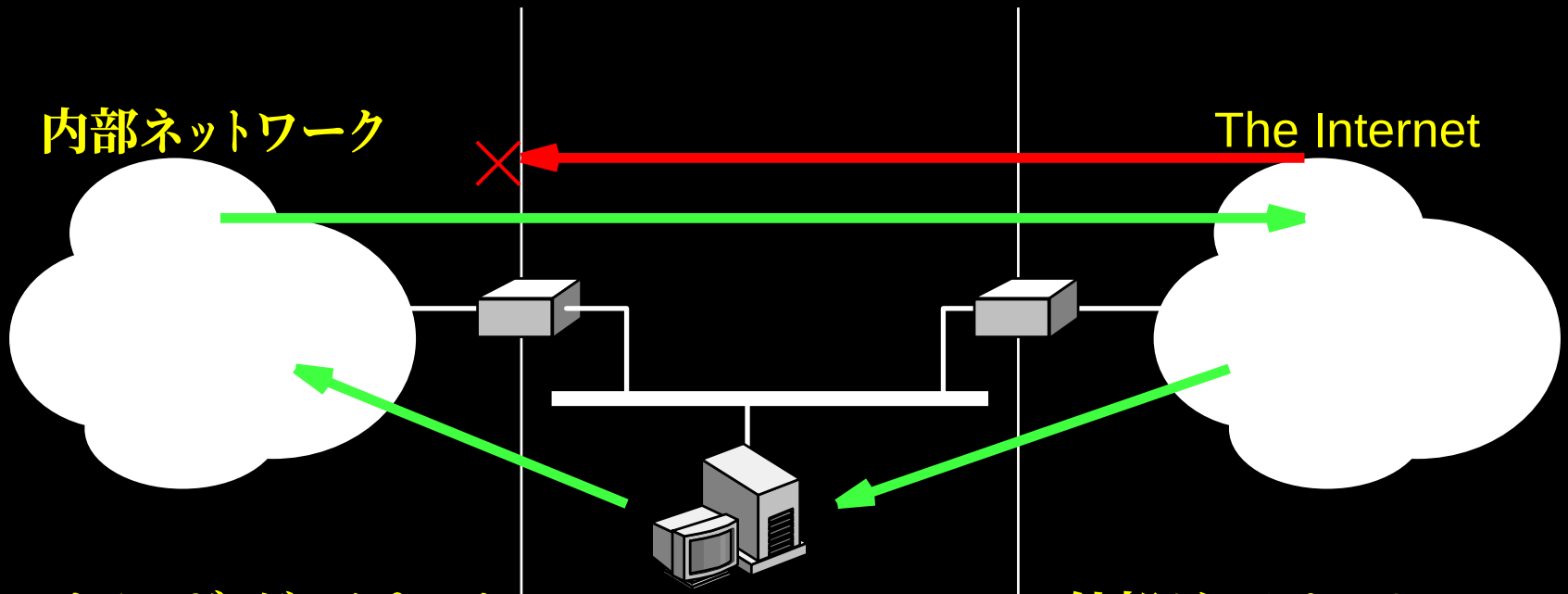
ASN-WIDE 2500

OCN 4717

EGP
Exterior Gateway Protocol

- ASとASとの間でのネットワーク到達情報を伝達するプロトコルの総称.
- EGP,BGP(Border Gateway Protocol) BGP2,BGP3とへて現在はBGP4がもちいられている.

Packet Filtering Firewall



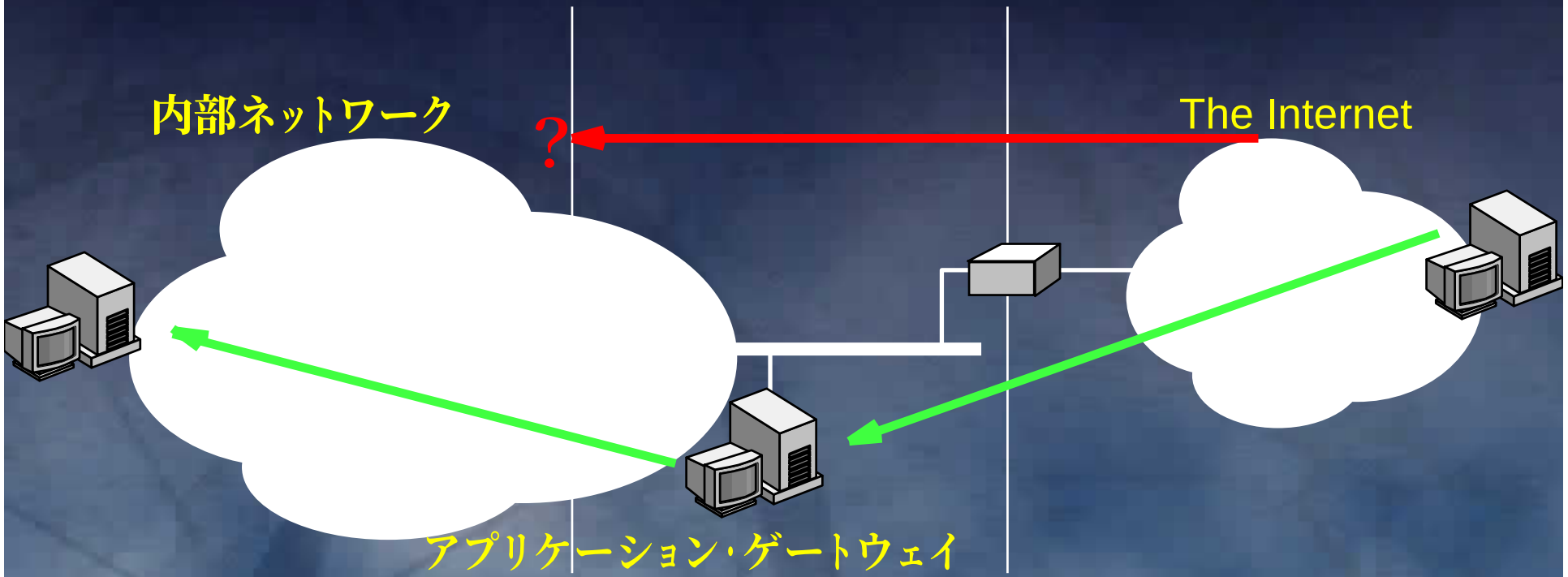
フィルタリング・ゲートウェイ
(外部から直接内部にアクセスする通信をブロックする)

外部ゲートウェイ
(全てを通過させる)

出島
(アクセスコントロールをこのゲートウェイに集約。攻撃の発見のためにモニタリングも実施)

WIDE

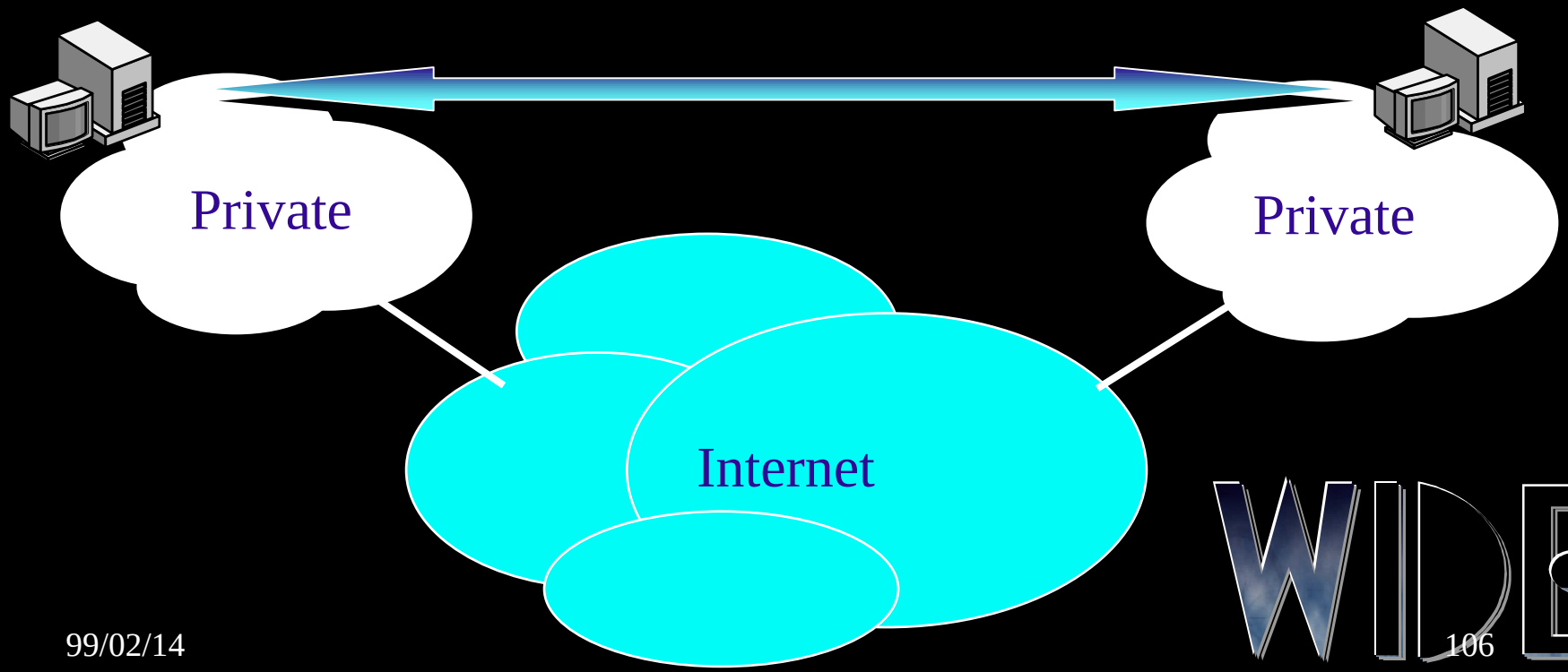
Application Gateway



アプリケーション･ゲートウェイ
外部からはこのマシンのみが見える
外部アプリケーションはこのマシンに接続。
プロキシ的な役割

VPN

- Virtual Private Network
- プライベートネットワーク間でパケットをカプセル化してインターネット上を配送



アプリケーション
アーキテクチャ

アプリケーション アーキテクチャ

- Client - Server モデル
- WWW
  - キャッシュ
  - Proxy
  - Java
  - CGI vs Java
- SLP

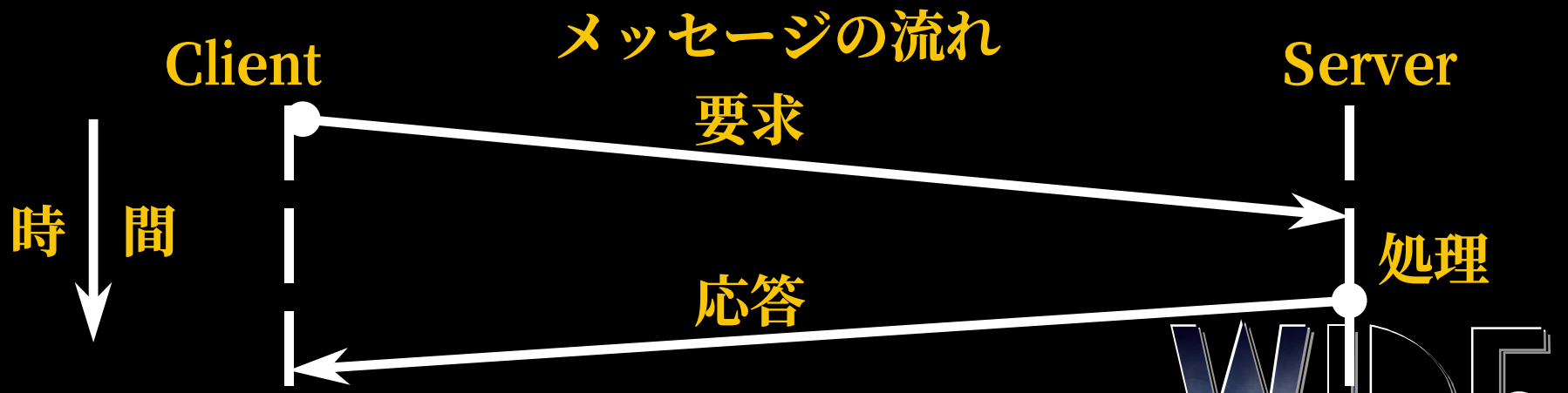
クライアント・サーバモデル

- 通信相手とのタイミング
  - 2つのプログラムのコミュニケーション
    - 2つが非同期だとうまく通信できない
    - 一方が待ち受けし続け、一方がリクエストを送る
      → クライアント・サーバモデル
- クライアント
  - 能動的にサービス提供を促す側
- サーバ
  - 受動的にサービス提供する側

WIDE

クライアント・サーバ

- クライアントとサーバの役割分担
  - クライアント
    - 通信を開始するアプリケーション
  - サーバ
    - クライアントからの要求を待ち受けるアプリケーション

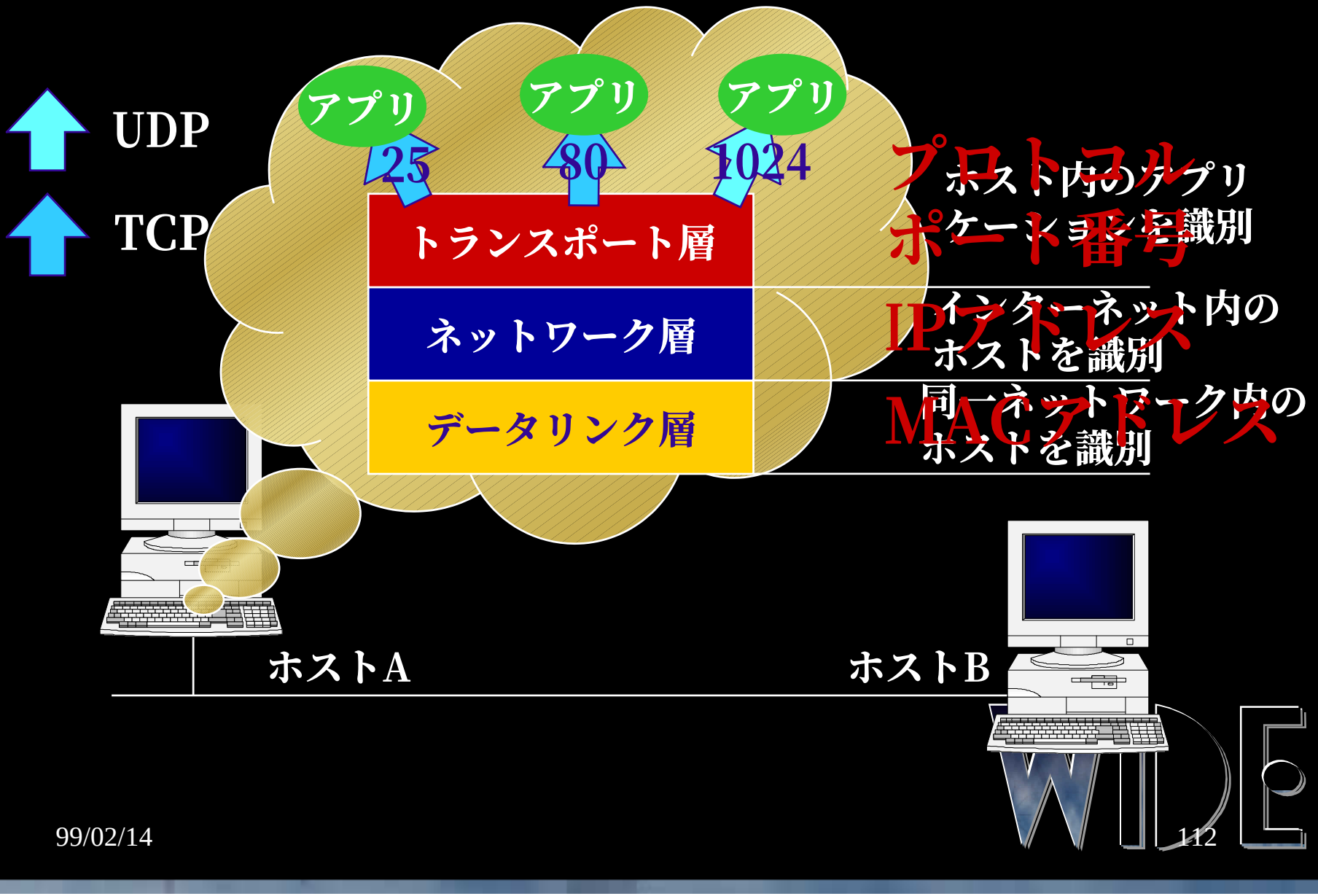


クライアント・サーバの例

- サーバ（サービスを提供する側）
  - finger サーバ　（fingerd）
  - WWWサーバ　（httpd）
  - 体育予約サーバ
- クライアント（サービスを受ける側）
  - finger クライアント　（finger）
  - WWWクライアント　（Netscape）
  - 体育予約クライアント

WIDE

アプリケーションの識別



WWWの仕組み

- クライアントサーバモデル
  - クライアント
    - Webブラウザ
    - 情報の獲得･表示を行う
  - サーバ
    - Webサーバ(httpd)
    - リクエストされた文書(ドキュメント)を送る
- URL
  - Uniformed Resource Locator
  - リソース(情報)の取り方と場所を一意に表記

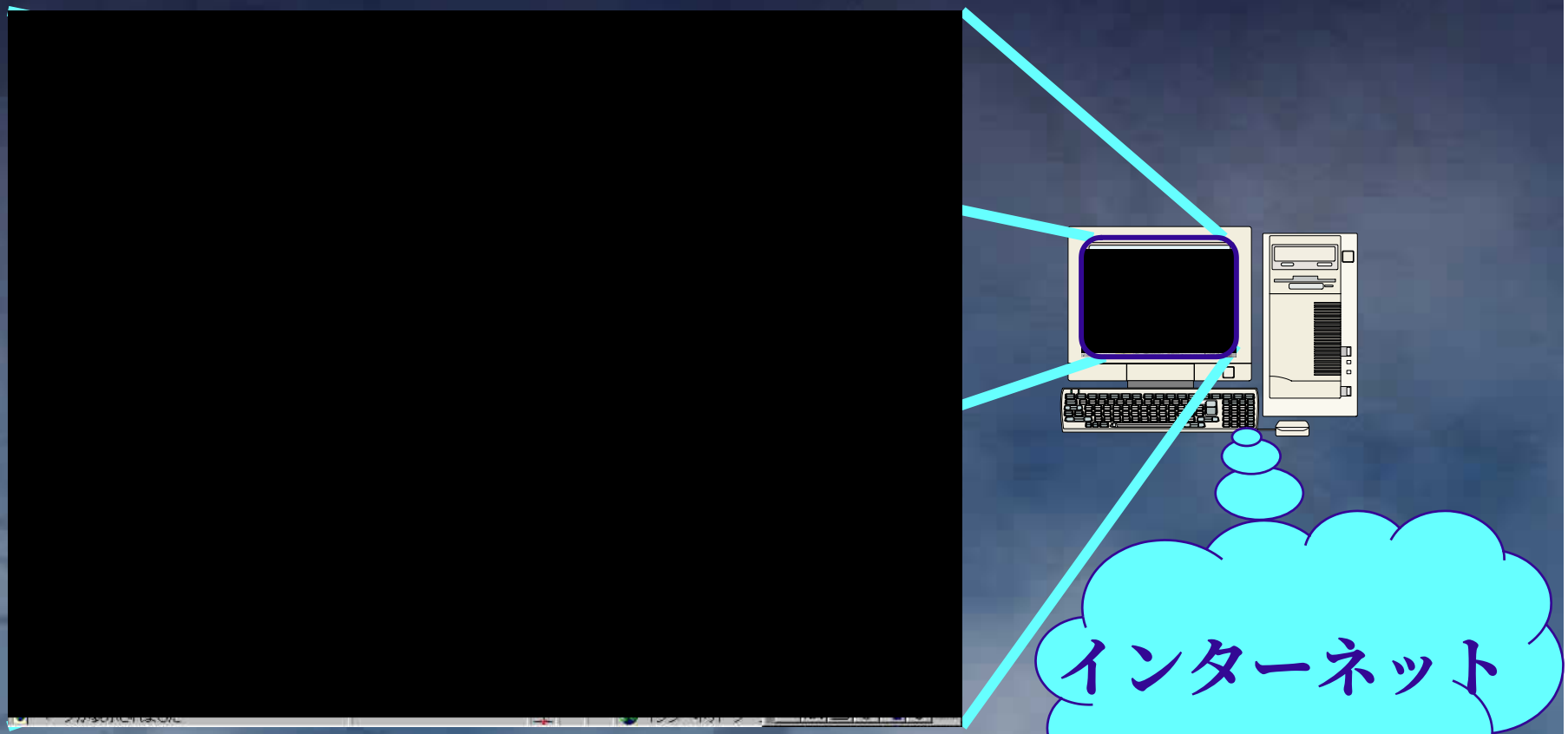
http://www.nic.ad.jp/iw97/index.html

入手方法(プロトコル)　サービスしているサーバ　サーバ内の場所(フォルダ・ファイル名)

WWWの仕組み

- Proxy サーバ
  - 中間で両方の代理として動作
    - クライアントの代わりにサーバにアクセス
    - サーバの代わりのクライアントに情報を提供
  - 様々な利用方法
    - キャッシュ
    - セキュリティ
- HTTP
  - Hyper Text Transfer Protocol
  - TCPを用いた通信
  - 様々な種類のリソースを転送
    - HTML文書
    - plain Text
    - GIF画像 etc…
  - 単純なプロトコル
    - 数種類のRequest Method

WWW

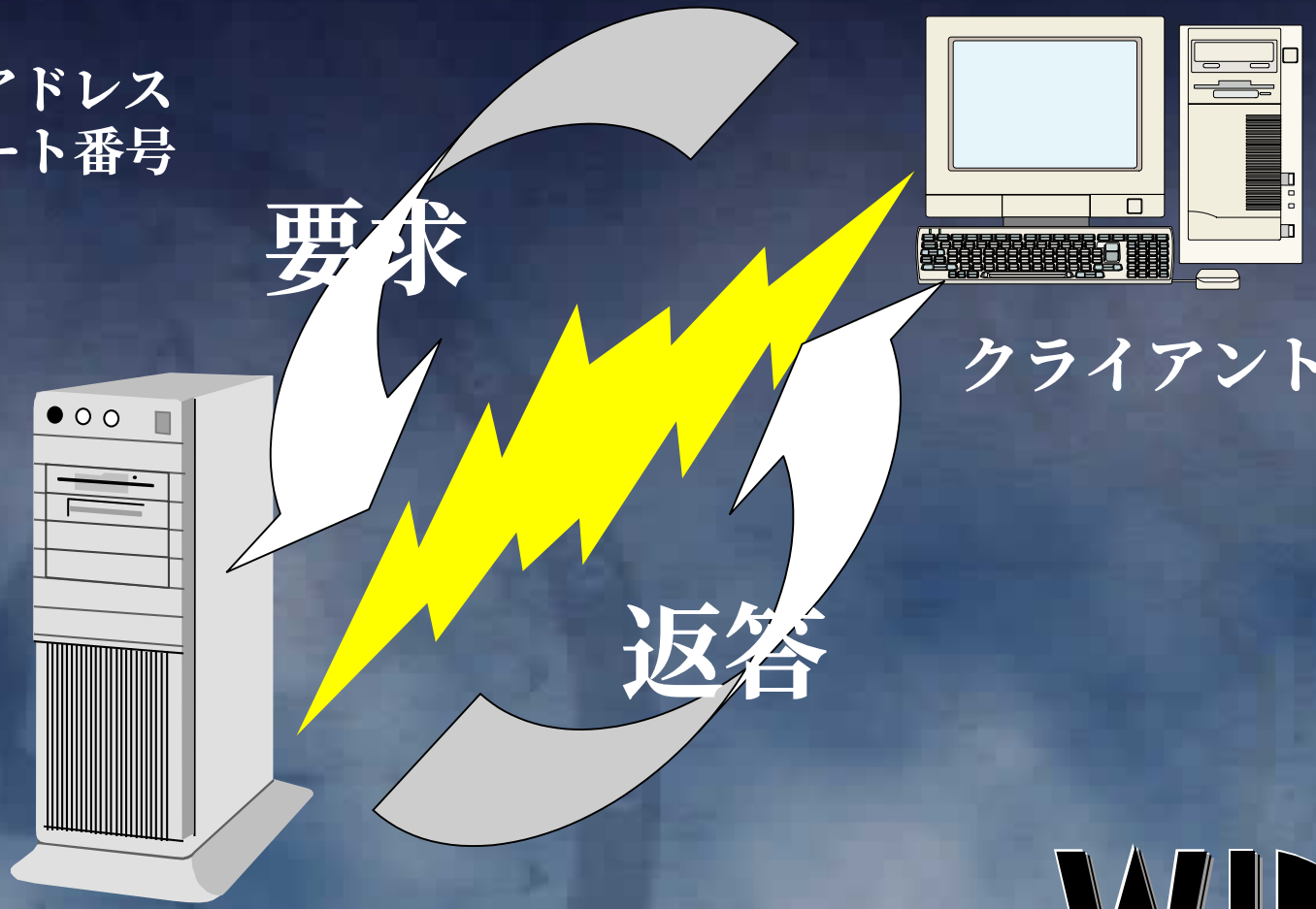


CGI

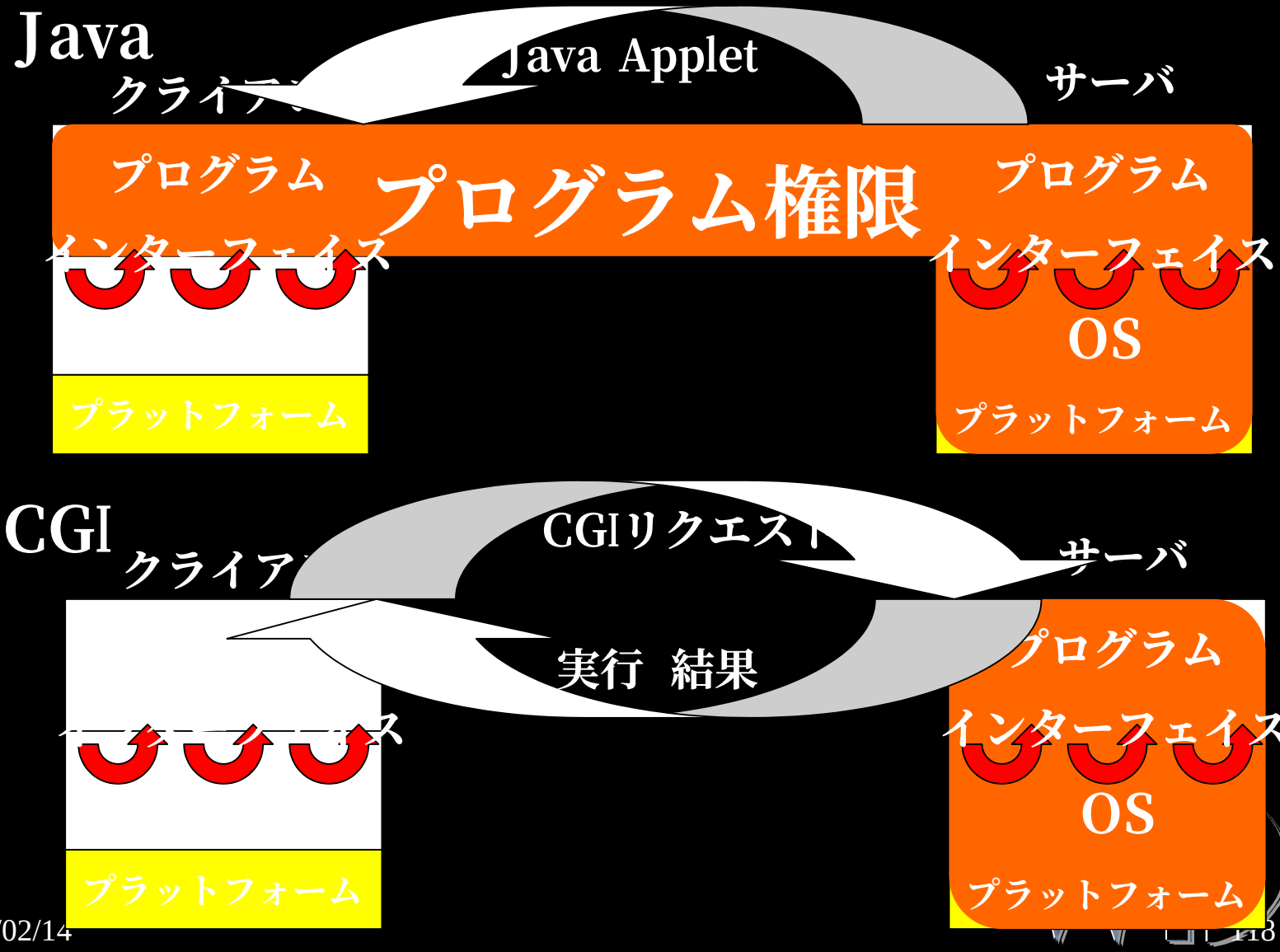
- 動的なContentsを作るための仕組み
  - プログラムインターフェイス
  - 決められた方式でクライアント側からの情報を渡す
    - 2種類のMethod
      - XXX
      - 環境変数
  - 動作する言語は考慮しない
    - なんでもOK

サーバとクライアントの関係

IPアドレス
ポート番号



プログラム実行とその権限

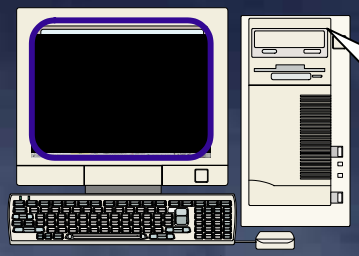


プログラム実行とその権限

Java

- プラットフォームを意識しない
  - Java VM
- ネットワークを介した環境を考慮
  - プログラムのダウンロード
  - 実行ホストへのセキュリティ対策
- WWWのContentsとして普及
  - クライアント側で実行

Java
ネットワーク上の動作

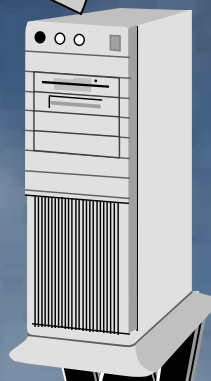


Java

これを実行してね

URL

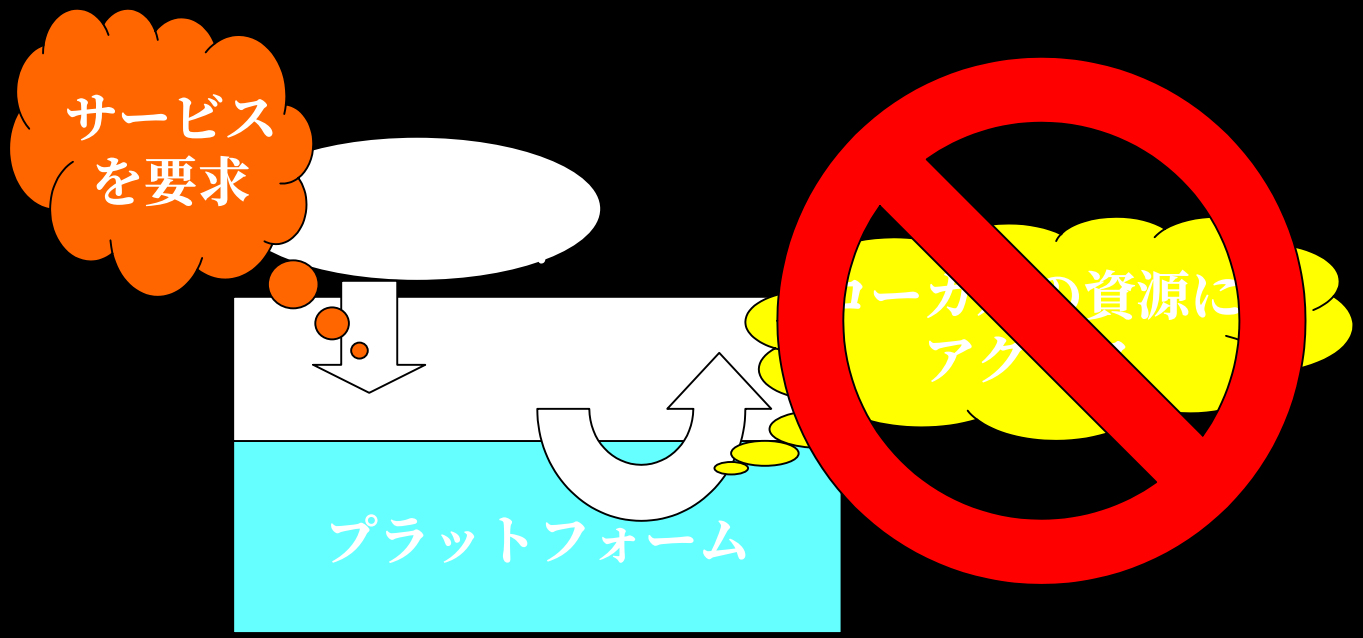
プログラ



WWWサーバ

99/02/14

Java
動作概要



Service Location Protocol as of Atlanta Olympic 1996