

評価方法

- 中間レポートと、期末レポート
- 出席はとらないが、、、
- 質問かコメントを義務付ける
 - 学期中、講義に関する技術的な内容の質問やコメントを最低2回、授業中に行うこと
 - よい質問やコメントは、成績の加点対象
 - 質問者は、講義終了後に名前と学籍番号を申告のこと

インターネット応用特論

1. IPv4、IPv6、UDP、DNS

太田昌孝

mohta@necom830.hpcl.titech.ac.jp

<ftp://ftp.hpcl.titech.ac.jp/appli1j.ppt>

講義の構成

- 前期
 - インターネットインフラ特論
 - 物理層、データリンク層、ネットワーク層
- 後期予定
 - インターネット応用特論
 - トランスポート層、アプリケーション層

通年の講義の目的

- インターネットの原理、インターネット流の
プロトコル設計の「こつ」を理解する
 - エンドツーエンド原理(RFC1958)
 - グローバルコネクティビティ
 - スケーラビリティ
- 今後はプロトコルの時代
 - 多様なアプリケーションに応じたプロトコル
 - APIは二の次

講義の参考となる資料

- RFC(一部和訳もあり)
- 「本当のインターネットをめざして」、情報処理学会誌、全36回(1999年4月号～2002年3月号)
- 「インターネットの真実」、週間東洋経済(2001年1月より2002年4月まで連載)
- 昨年度の講義のOHP
 - ftp://ftp.hpcl.titech.ac.jp/2018/appli*.ppt

今期の講義計画(1)

1. IPv4、IPv6、UDP、DNS
2. トランスポート層：TCP、輻輳制御、ロングファットパイプ、マルチホーミング
3. ファイル転送：TFTP、FTP、リライアブルマルチキャスト
4. 文字通信：文字コードと国際化
5. 文字通信：TELNET、電子メール、SMTP、MIME
6. 文字通信：ウェブ、HTTP、HTML、GIF、JAVA

今期の講義計画(2)

- 7. 文字通信：家電機器制御
- 8. ストリーム通信：RTP、時間同期、クロック同期
- 9. ストリーム通信：電話網とインターネット、インターネット電話
- 10. インターネットと社会：利用者認証、課金、RADIUS
- 11. インターネットと社会：知的所有権、法律
- 12. インターネットと社会：標準化：RFC、運用、実装、プロトコル設計

プロトコルとは？

- ネットワークで通信する手順

ところで、インターネットとは？

- 電子メールのことではない
 - 十年前には大真面目で主張されていた
- ウェブのことでもない
 - 現在でも勘違いしている人が多い
- アプリケーションのことではない
- インターネットはIP（インターネットプロトコル）を用いて、インターネットの原理に基づいて接続された網である

エンドツーエンド論法

<http://web.mit.edu/saltzer/www/publications/endtoend/endtoend.pdf>

- The **function** in question **can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system.** Therefore, **providing that questioned function as a feature of the communication system itself is not possible.** (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)

エンドツーエンド論法の例

データ落ち

- データ落ちは網では防げない
 - ノイズ、機器の故障等
 - バッファしても、バッファごと(多重)故障するかも
- データ落ちを防ぐには
 - 端末が再送するしかない
- 端末も落ちたらどうしようもないが、、
 - 逆に、端末さえ落ちなければ大丈夫(ありえる究極の安全性)

エンドツーエンド論法と エンドツーエンド原理

- エンドツーエンド論法により
 - 網の機能は、端末の助けなしでは、完璧には
なりえない
 - 網は、そこそこの性能があればよい
 - あれこれ頑張っても、どうせ完璧にはならない
 - 足りない部分は端末が補うか、そこそこの性
能で我慢する
- 多くの機能は、端末だけで完璧に実現可

エンドツーエンド原理と 網の中抜き原理

- 端末(エンド)でできることは網側ではやらない
 - 網機器は単能(端末を結ぶだけ)、高速
- 直接関係する端末でできることは他の端末ではやらない
 - スケーラブル(負荷が集中しない)
 - 高信頼(端末が動いてなんらかの経路で通信さえできればシステムは動作)

インターネットじゃないもの(1)

電子メール

- UUNET(JUNET)はインターネットではなかった
- パソコン通信もインターネットではなかった
- 電子メールはインターネット上でも動くアプリケーションの一種
 - その他のネットワーク(電話網)上でも動く
- 昔の(海外)電子メールは有料だった
 - 今でも携帯電話網上の電子メールは有料

インターネットじゃないもの(2)

ウェブ

- ウェブはインターネットではない
 - マイクロソフトはそのへんを誤魔化しているが
- ウェブもインターネット上でも動くアプリケーションの一種
 - その他のネットワーク上でも動く
 - 携帯電話網からウェブを見るのは有料
 - もちろん、携帯電話網はインターネットではない

インターネットじゃないもの(3)

電話

- 電話は電話網ではない
- 電話は電話網上でも動くアプリケーション
 - その他のネットワーク上でも動く
 - インターネット電話自体は無料
- もはや電話はインターネットの販促ツール
 - わざわざそれ自体をサポートする価値なし
 - 携帯電話はモバイルインターネットの販促ツール

インターネットじゃないもの(4)

電話網

- 電話は電話網ではない
- 電話は電話網上でも動くアプリケーション
 - その他のネットワーク上でも動く
 - インターネット電話自体は無料
- もはや電話網は不要
 - 携帯電話網も同じ
 - 電話番号も不要
 - IPアドレスが基本(内線番号があってもいい)

ネットワーク

- 物流網
 - 郵便、宅配便、コンビニ
- 情報通信網
 - 出版網（書籍、新聞、レコード（CD）、映画）
 - 金融網
 - 電話網
 - 放送網
 - インターネット

インターネットは情報通信網を 中抜きする

- インターネットは情報通信の価格破壊
 - 出版網、金融網、電話網、放送網は消える
 - 社会の情報通信コストの削減
 - インターネットビジネス自体は儲からない
- 出版、金融、電話、放送というサービスは
 - インターネット上に移行して残る
 - 社会の活力は増大する

出版網

- 同じ情報を大量に配布
- 情報流通は遅くていい
- 著作権法による保護
- いまのインターネットの好餌
 - 壊滅寸前

金融網

- お金のやりとりを管理
- 物流網でもあるが、今や、情報通信網としての面がはるかに大きい
- セキュリティー！！！！
 - － つまりは、誰が損失をかぶるか

電話網

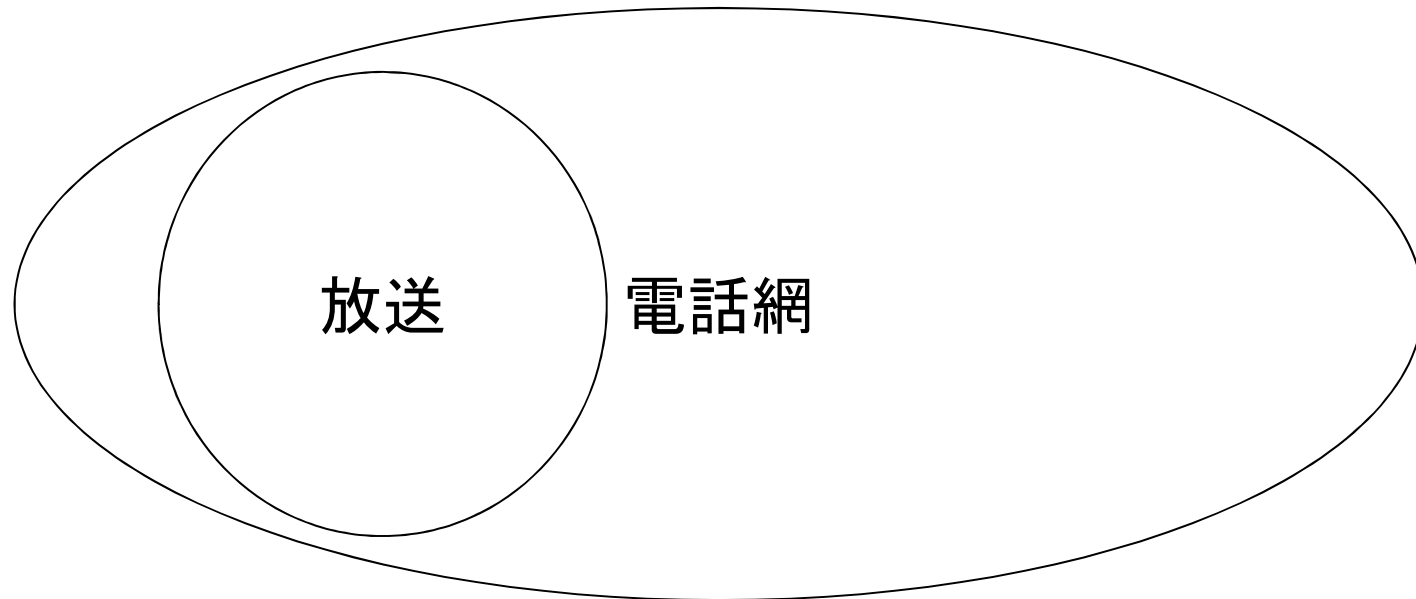
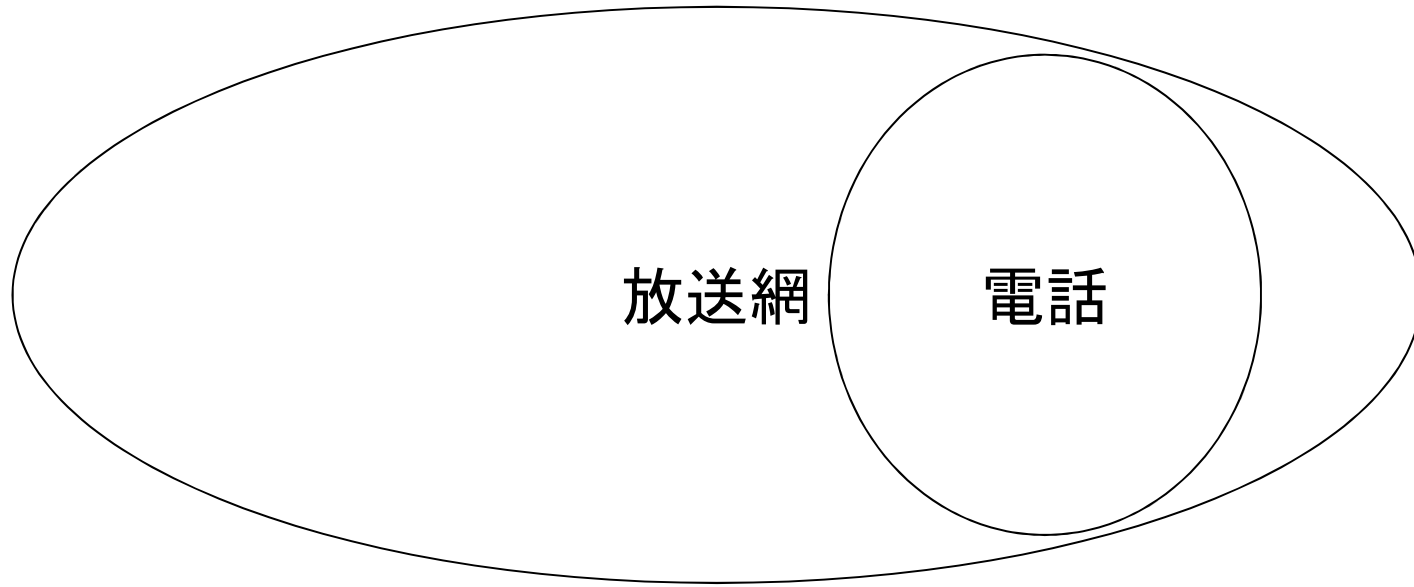
- 音声を実時間で伝送する網
 - 音声伝送の帯域を確保
 - 音声伝送の遅延を最小化(保証)
- 専用線事業も
 - あくまで音声伝送事業が主
- 遅くて高い
- 電電公社として保護、電気通信事業法で開放

放送網

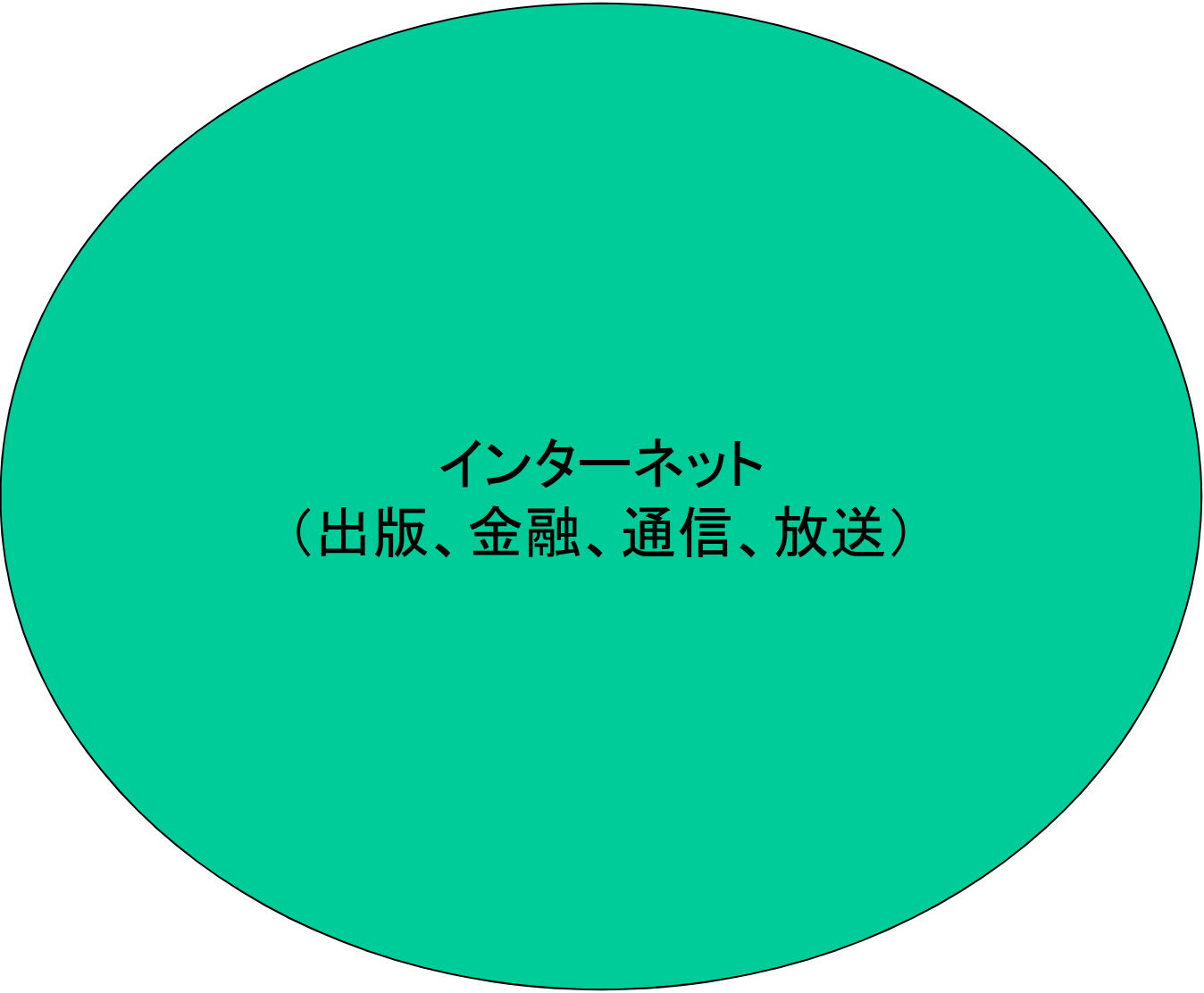
- 音声、画像を実時間で多数に伝送する網
 - 伝送帯域を確保
 - 遅延を最小化
- 電波による広域一対多通信
 - ブロードキャスト／マルチキャスト
- 放送法による保護

放送と通信の融合

- 電話網からみて
 - 電話網でも一対多通信は可能
 - 電話網(BISDN)への放送網の統合
- 放送網からみて
 - 放送のフィードバックを電話網から受ける
 - 電波でも1対1通信は可能
 - 放送網への電話網の統合？
- インターネットからみて
 - ぜんぶインターネットに統合



同床異夢の「放送と通信の融合」



インターネット
(出版、金融、通信、放送)

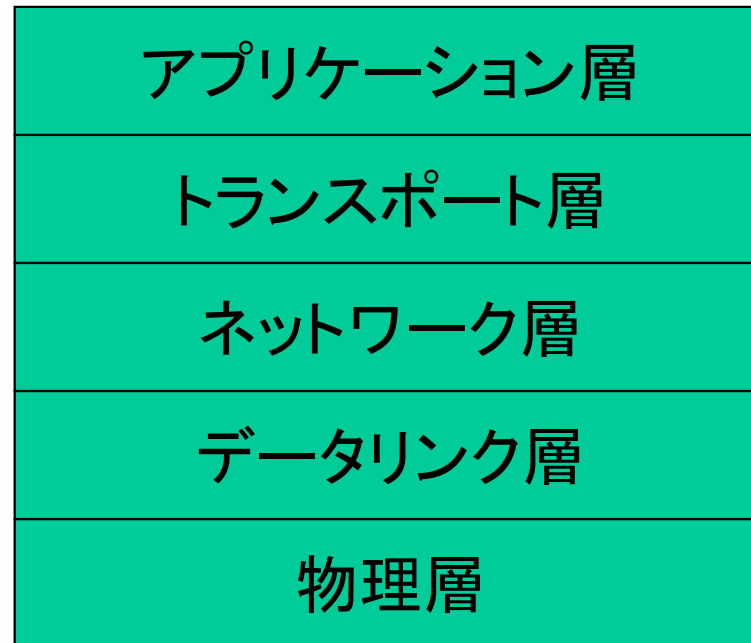
インターネットによる情報通信サービスの統合

プロトコルのレイヤリング

- 扱う対象の抽象度に応じて層化
 - OSIの7層モデル
 - インターネットの5層モデル
- プログラムでいえば、サブルーチン化(構造化)に相当

アプリケーション層	7層
プレゼンテーション層	6層
セッション層	5層
トランスポート層	4層
ネットワーク層	3層
データリンク層	2層
物理層	1層

OSIのレイヤリング構造



インターネットのレイヤリング構造

ネットワーク層

インターネットワーキング層

- 多数のデータリンク層を統合して大域的なひとつの網として動作させる
- 大域的な中継（ルータ、ゲートウェイ）
- プログラミングでいえば、ファイル管理やプロセス間通信

トランスポート層

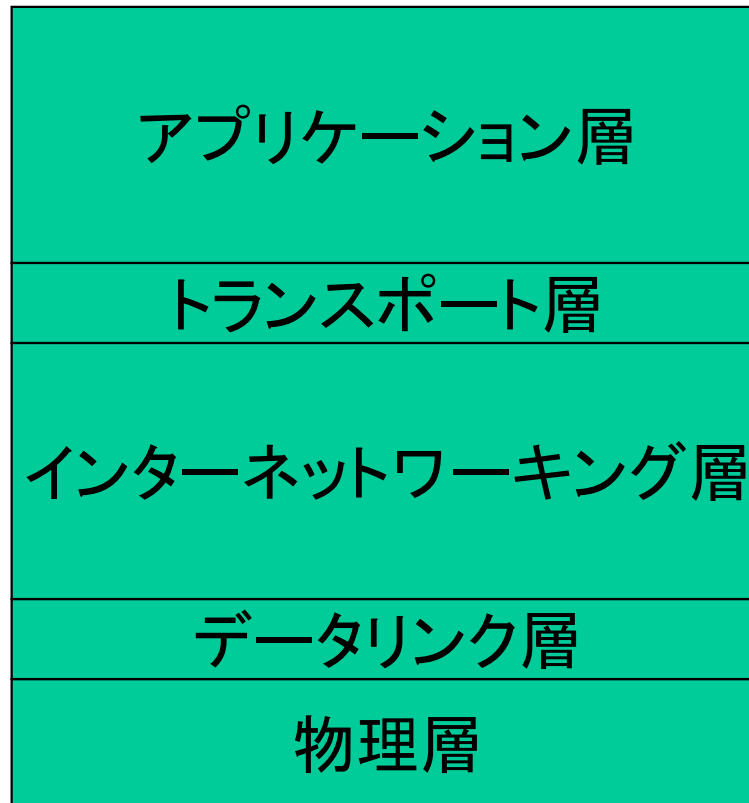
- ネットワーク層では個々の端末を識別
- トランスポート層では個々の通信を識別
 - 同一端末間に複数の通信がありえる
 - 別の通信は別のプロセスで処理
 - ネットワーク中ではそれらの要求する帯域などが異なるかも、、、
 - ベストエフォートインターネットではどうでもいい
- プログラムでいえば、プロセス管理

セッション層、プレゼンテーション層、アプリケーション層

- プログラムでいえば、個々のプロセス内部の構造に相当
- ネットワークでいっても、端末内の個々のプロセス内部の構造に相当
- 網間接続がなければ、区別に意味なし
 - インターネットではアプリケーション層のみ

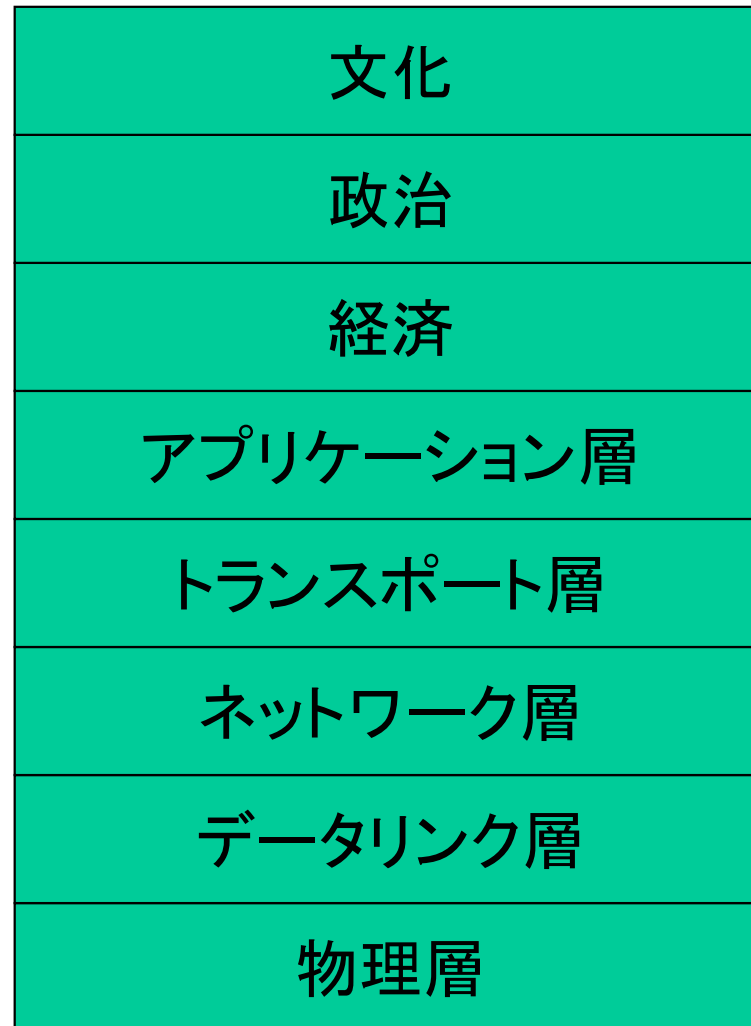
トランスポート層とアプリケーション層

- ベストエフォートでは、端末内部の区別でしかない
- 個別通信の個別プロセスへの振り分けだけはトランスポート層
- それ以上の区別には意味がない
 - 多くのアプリケーションで共通に使うプロトコル（TCP（信頼性確保と帯域管理のプロトコル）等）は慣例的にトランスポート層に分類



こここそインターネット

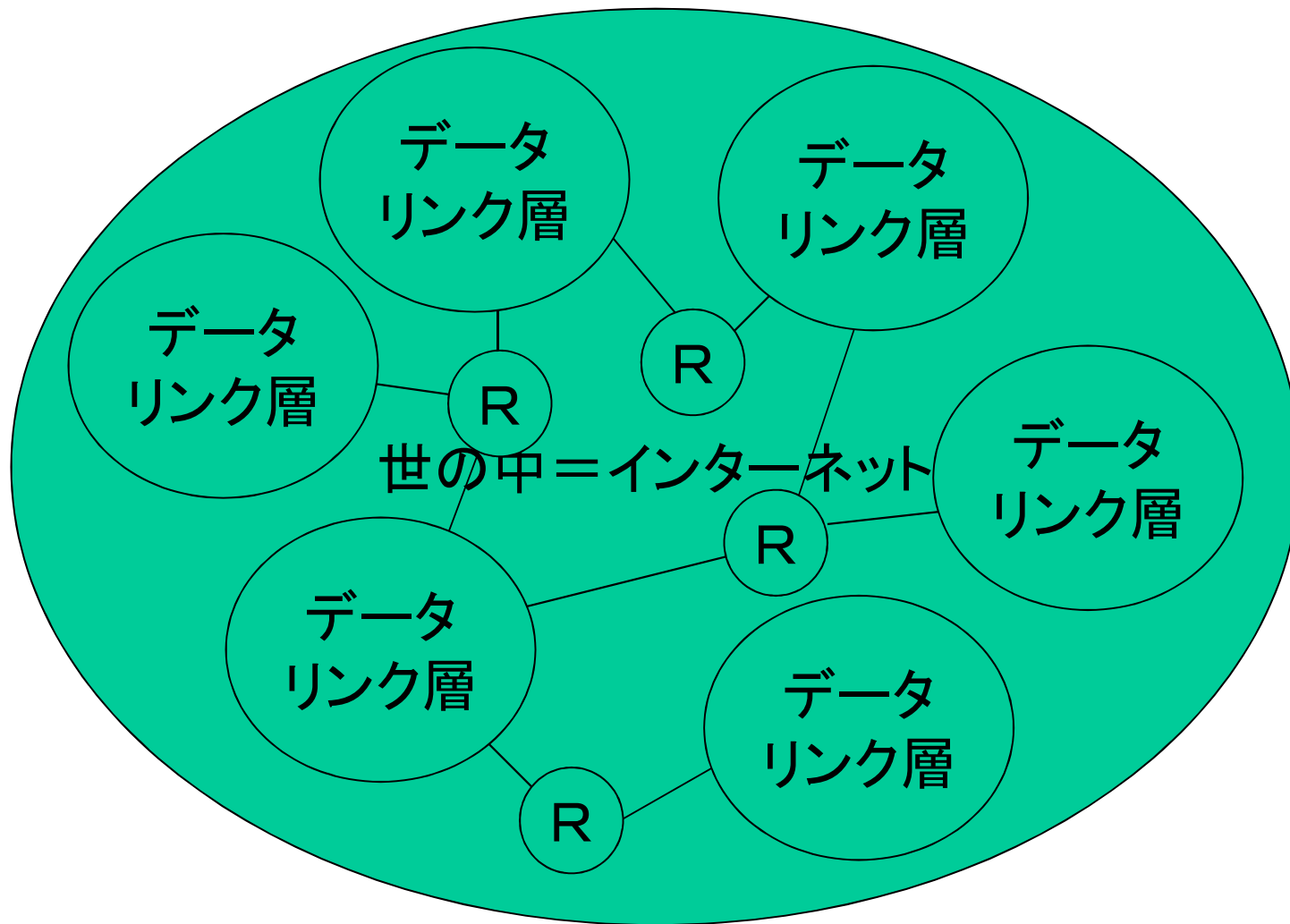
インターネットのレイヤリング構造



インターネットの上のレイヤリング構造

インターネットの構造

- CATENETモデル
 - 多数の小さな(機器の数が少ない)データリンク層をIP(Internet Protocol)ルータで相互接続したもの



R : ルータ

CATENETモデル

インターネットと網の構造

- インターネットの例
 - ダイヤルアップインターネット
- インターネットでない例
 - iモード
 - 携帯電話網からウェブや電子メールを利用
 - NAT
 - イントラネットとインターネットの間でIPヘッダを書き換えてアドレスを節約
 - 多くのプロトコルはNAT装置を越えられない

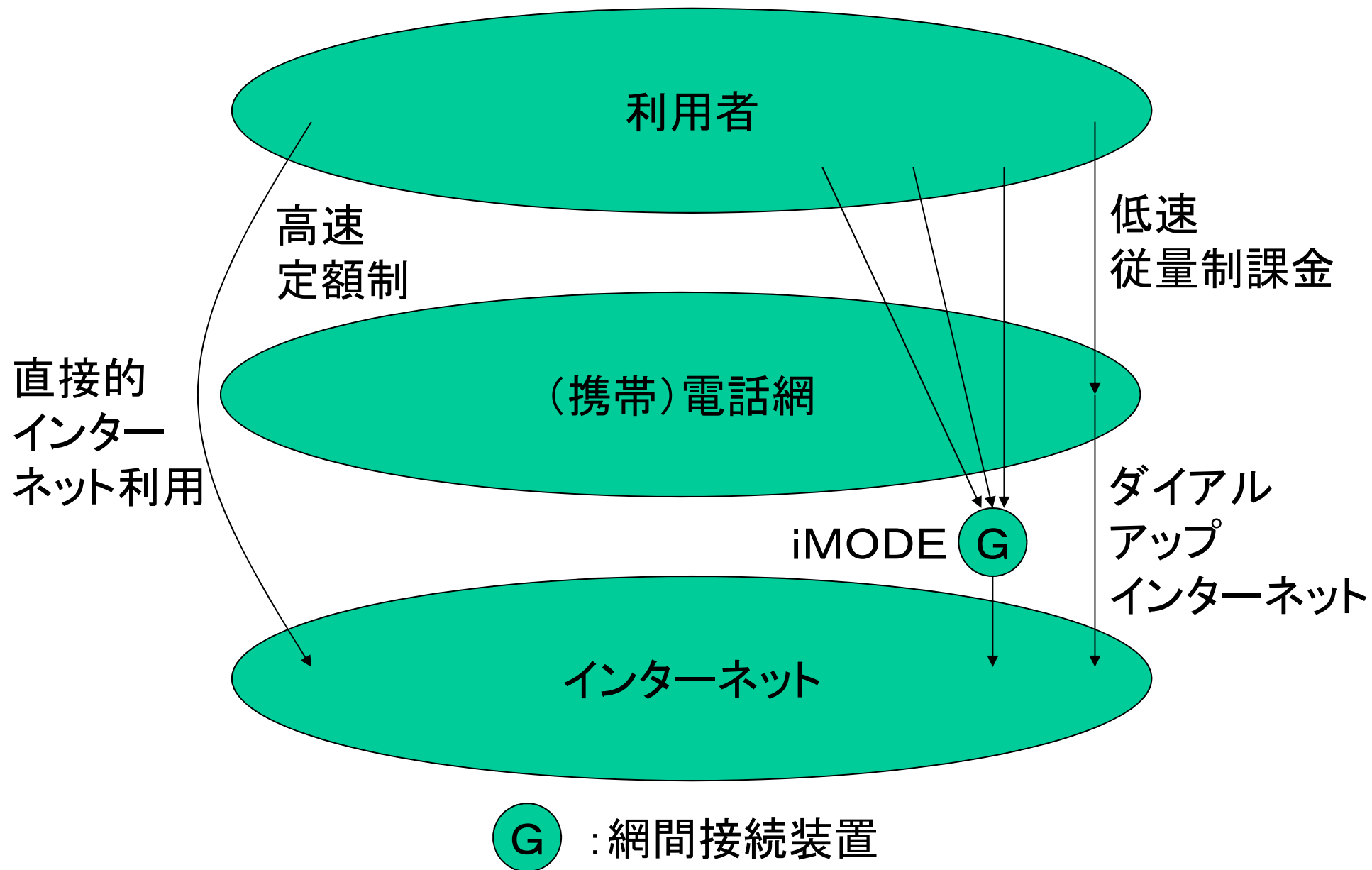
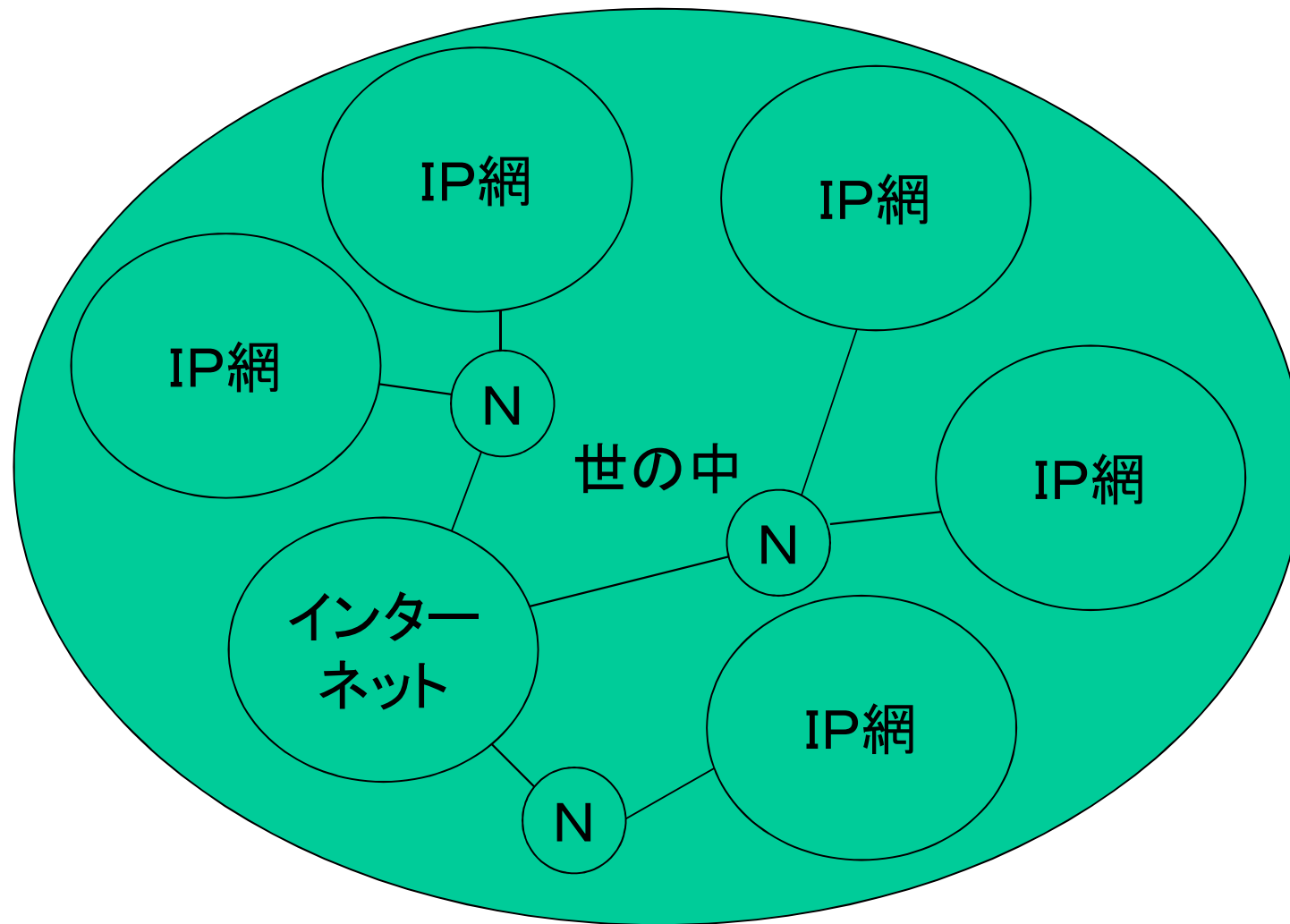


図 電話網とインターネット



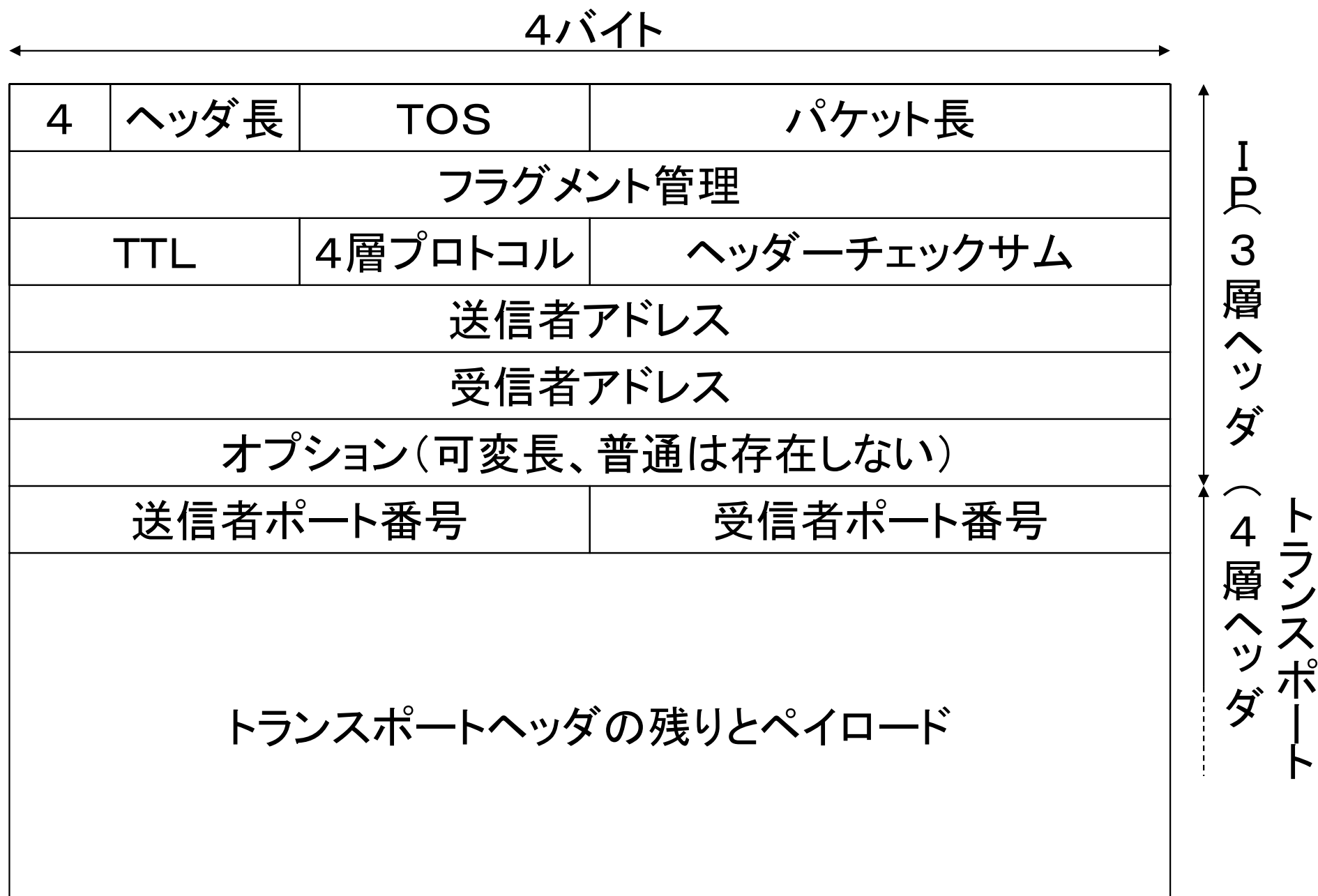
● N : NAT装置

NATとインターネット

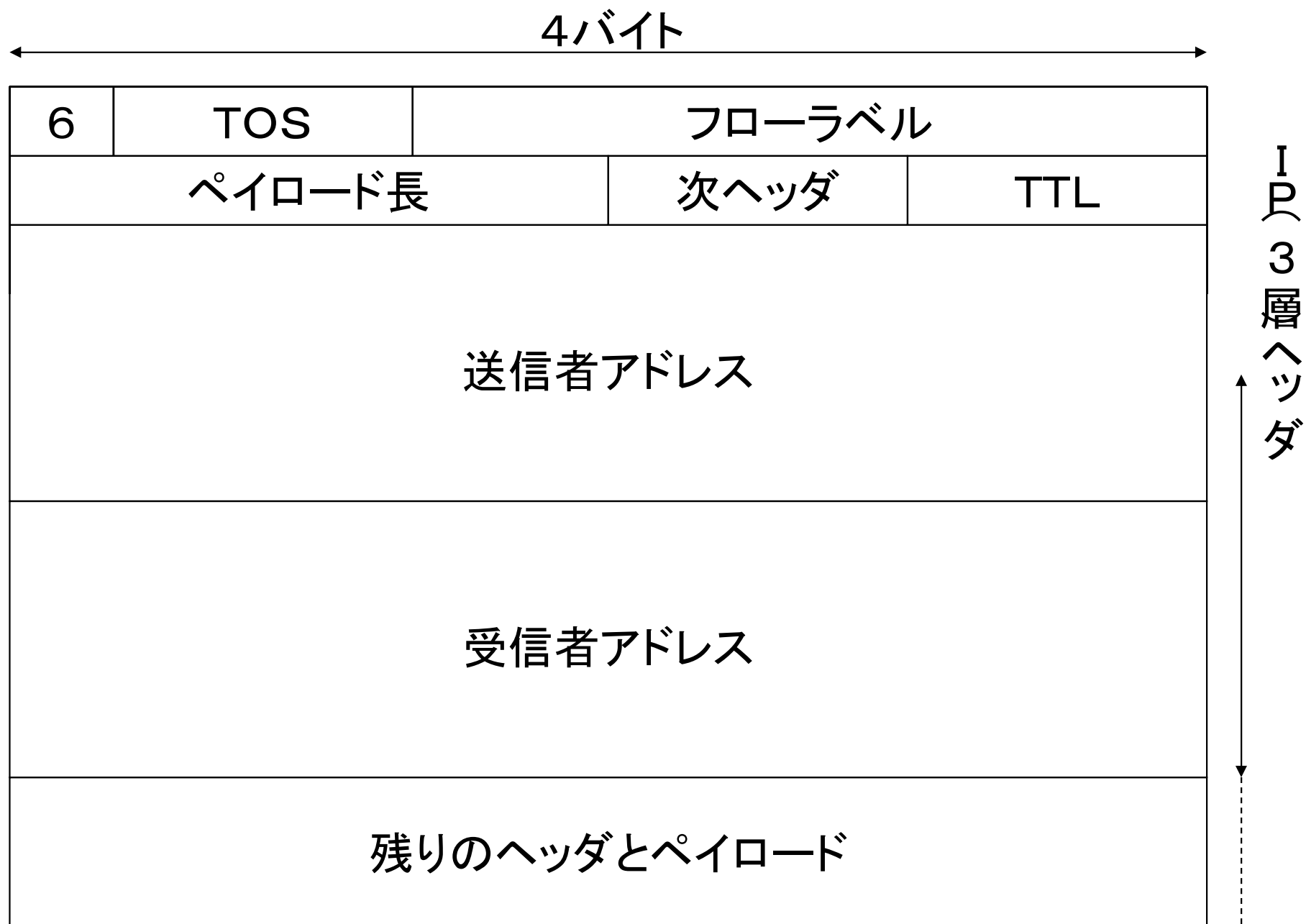
インターネットのデータ形式

IP (Internet Protocol)

- データをパケットにまとめて転送
- パケットは個別に行き先をもつ
 - データグラム
- IPv4では、20バイトのインターネットワーキング層ヘッダを付加
- さらにトランスポート層ヘッダが付加



IPv4パケットフォーマット



IPv6パケットフォーマット

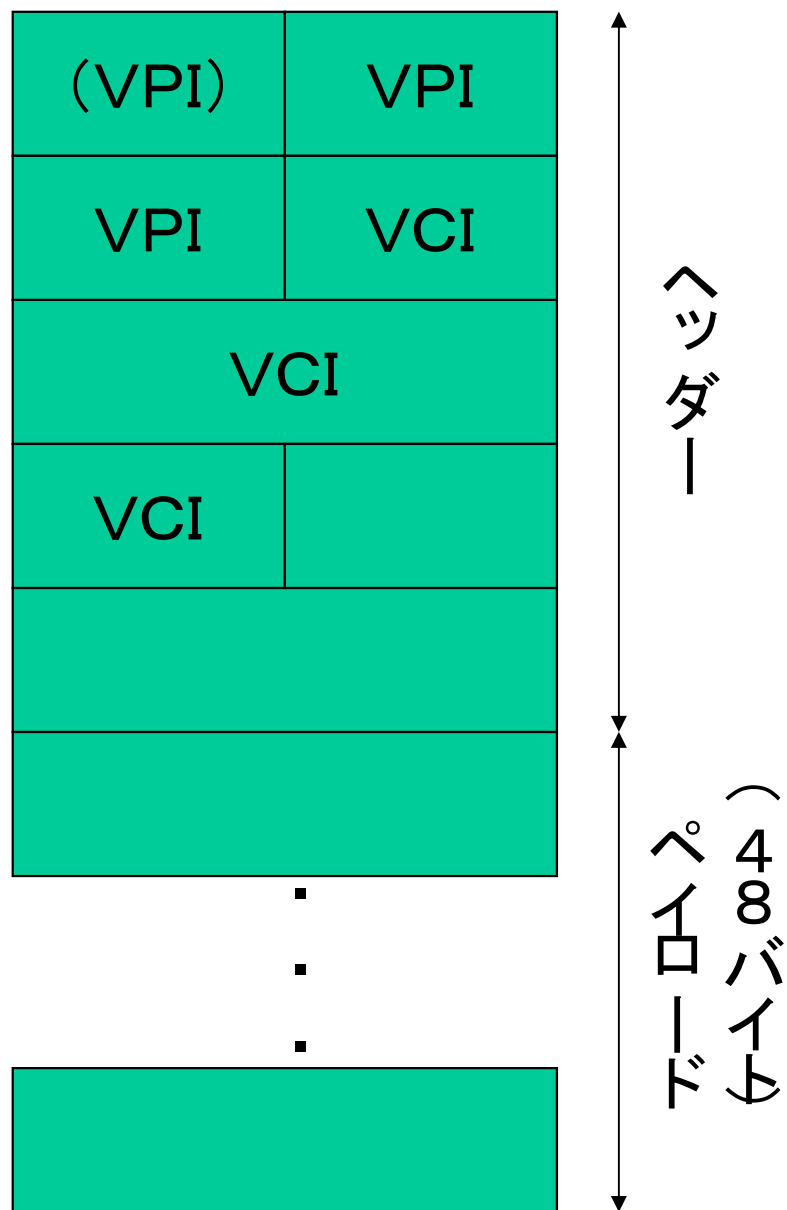
IP層の機能

- TTLを1以上減らしつつ、**受信者アドレスによって**、パケットを目的地にフォワード
 - 事前設定なし、帯域管理等なし
- IPv4ではMTUが小さくなるとパケットを分割(フラグメンテーション)
- TOSはType of Service
- IPv4(RFC791)は、アドレス空間が32ビット、IPv6は128ビット

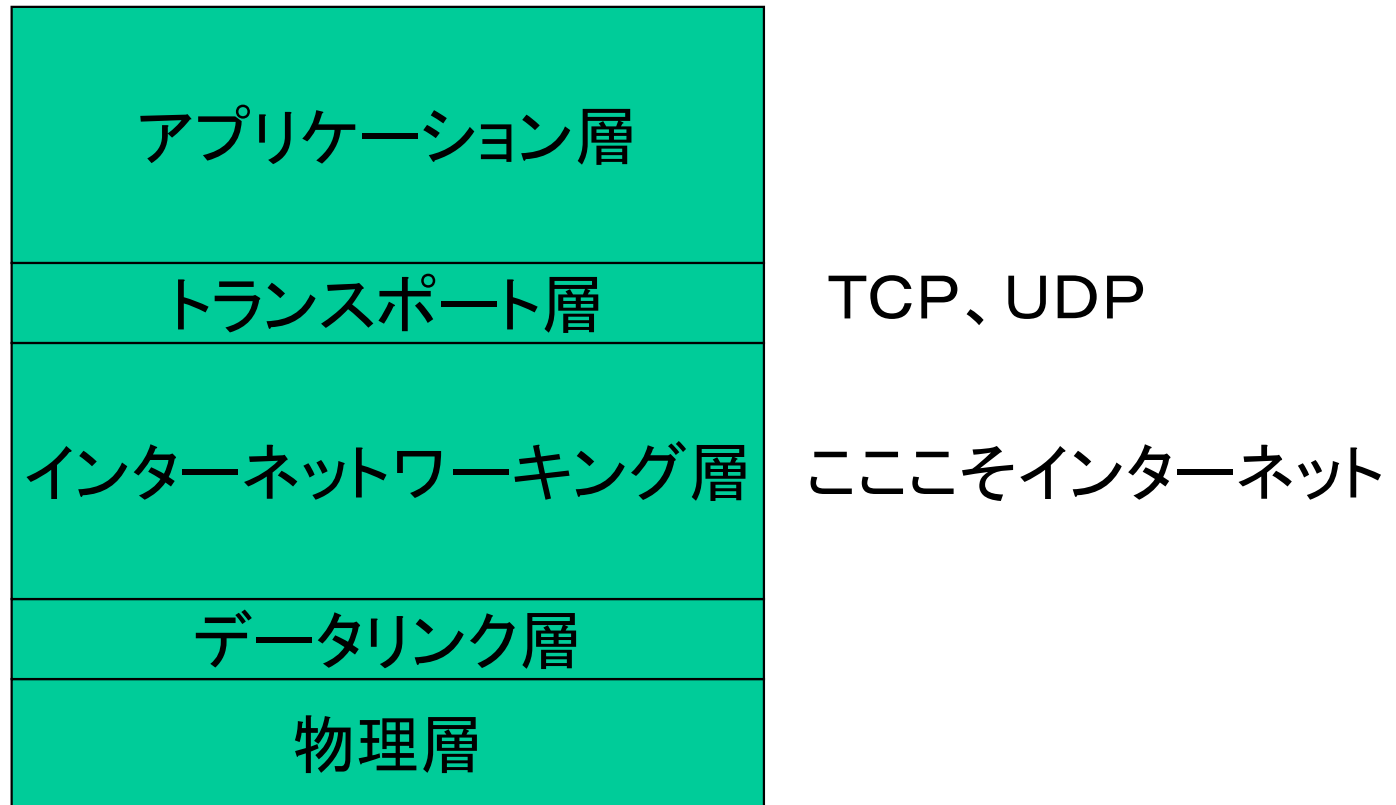
ATM

(Asynchronous Transfer Mode)

- 電話網的、パケット(セル)網
- データを48Bのセル単位で転送
- セルは個別にIDを持つ
 - 事前に、行き先を網に知らせ、IDを網から貰う
 - 時間と手間がかかる(電話では秒単位)
 - QoS保証が可能だが、実際には動作せず
 - バーチャルサーキット
- ヘッダは5B



ATMのセル

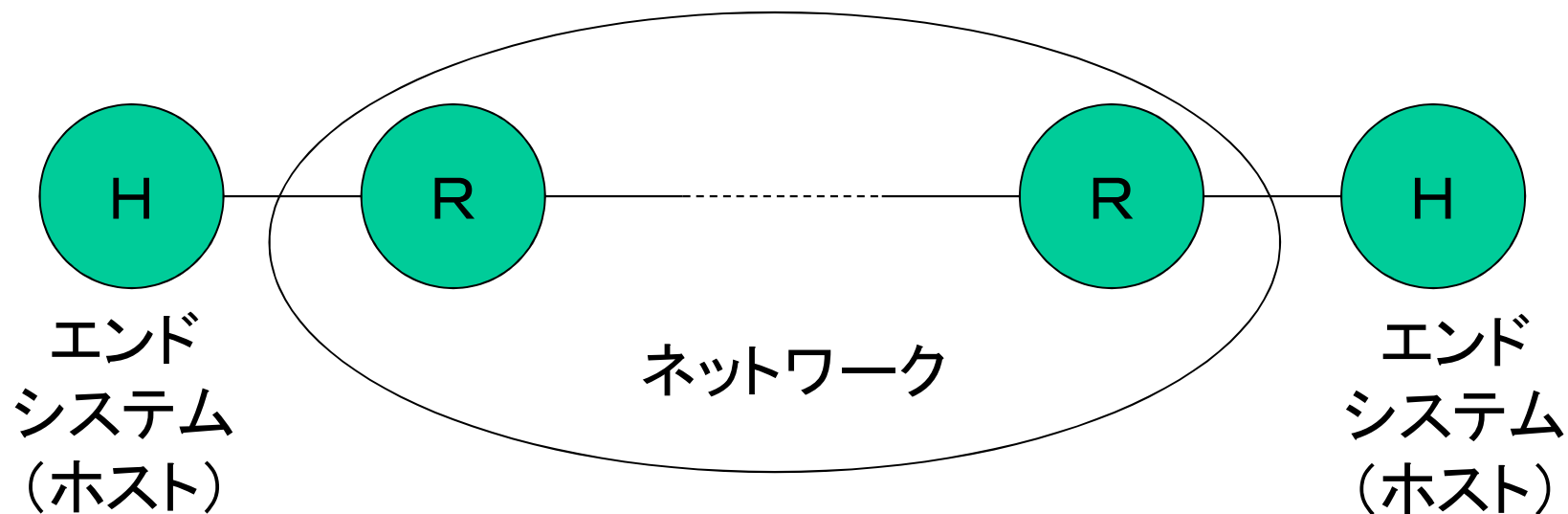


インターネットのレイヤリング構造

アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層

ネットワーク層
データリンク層
物理層

アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層

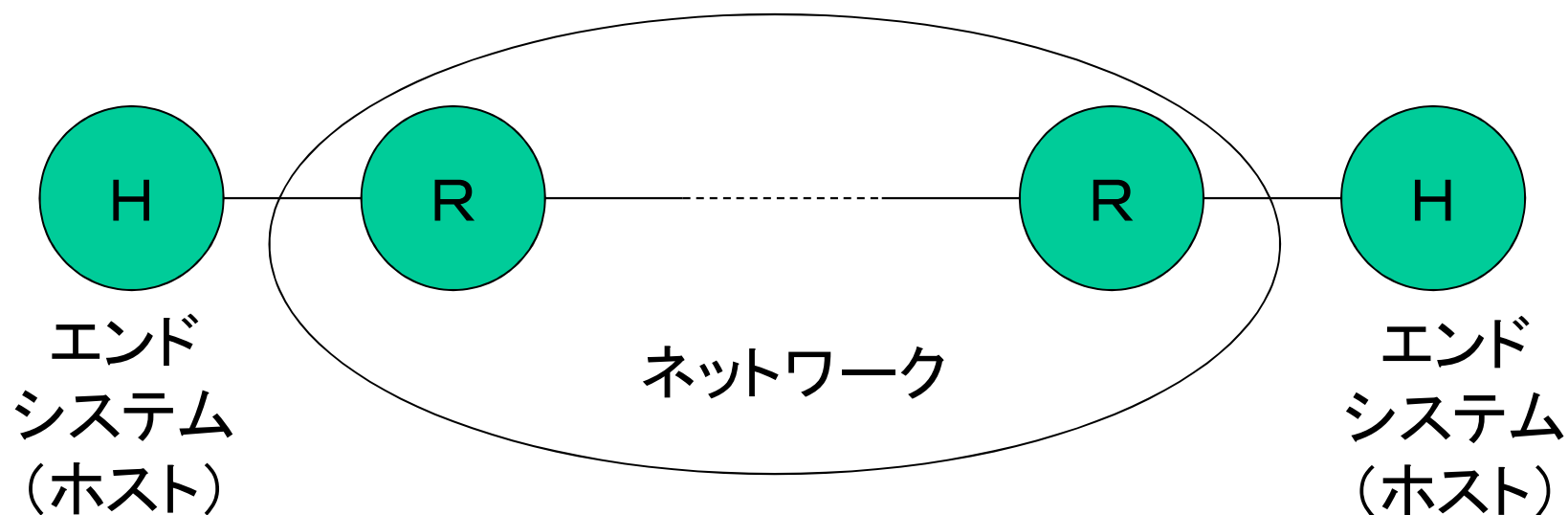


QoS保証のないインターネット

アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層

トランスポート層
ネットワーク層
データリンク層
物理層

アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層



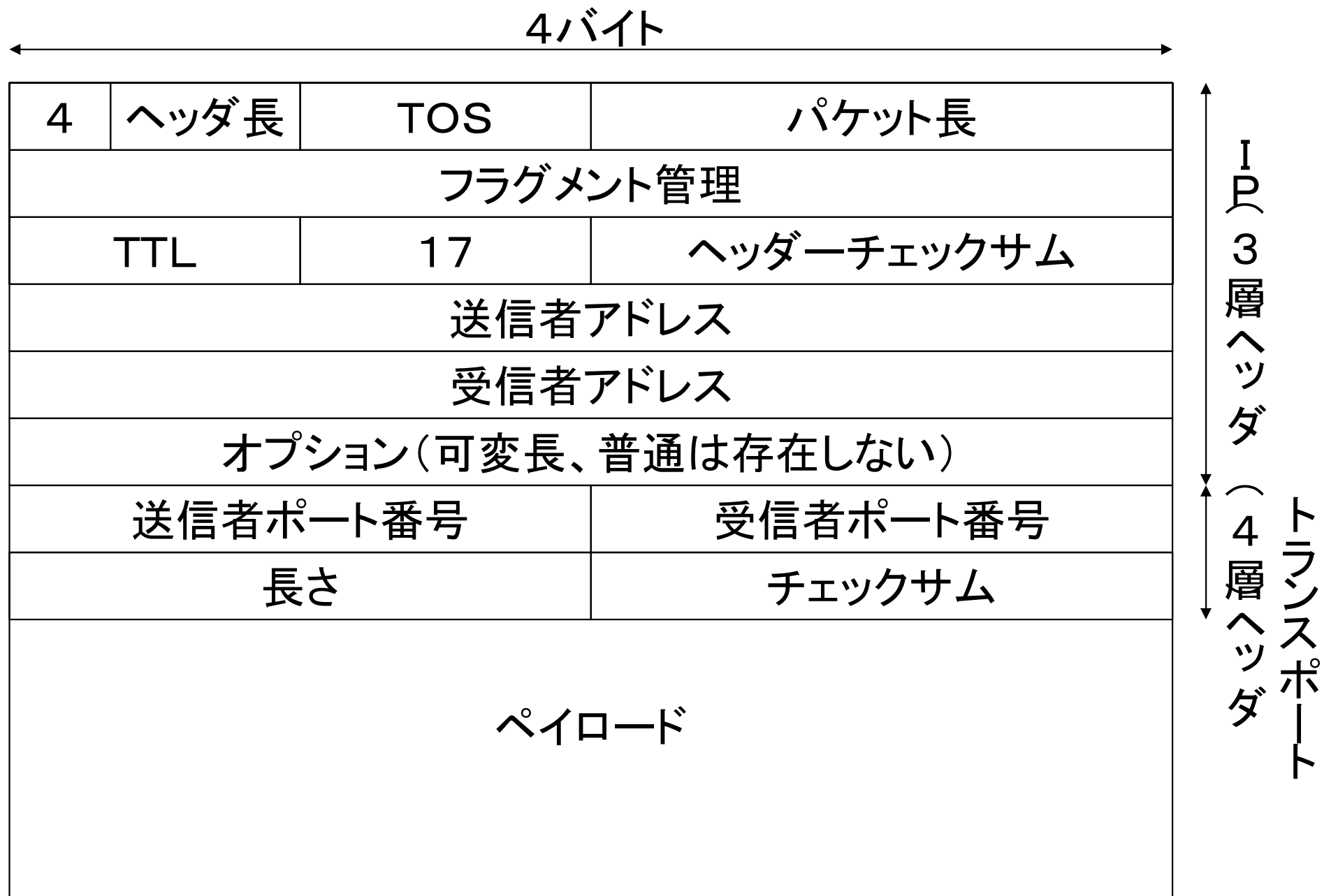
QoS保証付きのインターネット

TCPとUDP

- TCP (RFC793)
 - Transmission Control Protocol
 - データの誤りや欠けを検出して再送
 - 伝送レートの調節
- UDP (RFC768)
 - User Datagram Protocol
 - 何もしない(全てアプリケーションまかせ)
 - アプリケーションへの振り分けだけは、やる

UDP

- 単純
- トランスポート層ではコネクションなし
- 難しいことはアプリケーションまかせ
- TCPでできない通信は、すべてUDPで
 - 片方向通信
 - マルチキャスト、ブロードキャスト
- それ自体は軽い



IPv4 UDP パケットフォーマット

受信者ポート番号

- 受信ホストは、ポート番号に基づいて、パケットを各プロセスに振り分ける
- ポート番号はURLなどからわかる
- サービスごとにポート番号が決まっている場合もある
 - DNSは53

送信者ポート番号

- 返事や転送途中でのエラーを返す場合、返事の受信者ポート番号として利用可能
- なくてもよい(0)

長さ

- UDPヘッダー以降の長さ
- 不要？
 - 1つのIPパケットに複数のUDPパケットが入るわけでもない

チェックサム

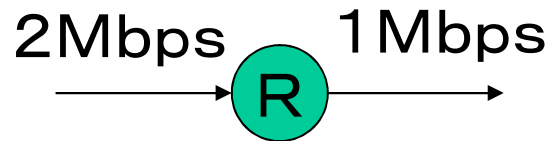
- 1の補数
 - 0は二つの表現(0 . . . 0, 1 . . . 1)をもつ
- IPv4では省略可能
 - 0 . . . 0は省略を意味する
- IPv6では省略不可
 - 0は不可

電話網とインターネット

- どちらがエラーが多い？

パケット落ちの原因

- 伝送エラーがあると、パケットは失われる
 - そうそうあることではない
- ルータのバッファがいっぱいになると、パケットをおとすしかない
 - インターネットでのパケット落ちの主因



混雑制御

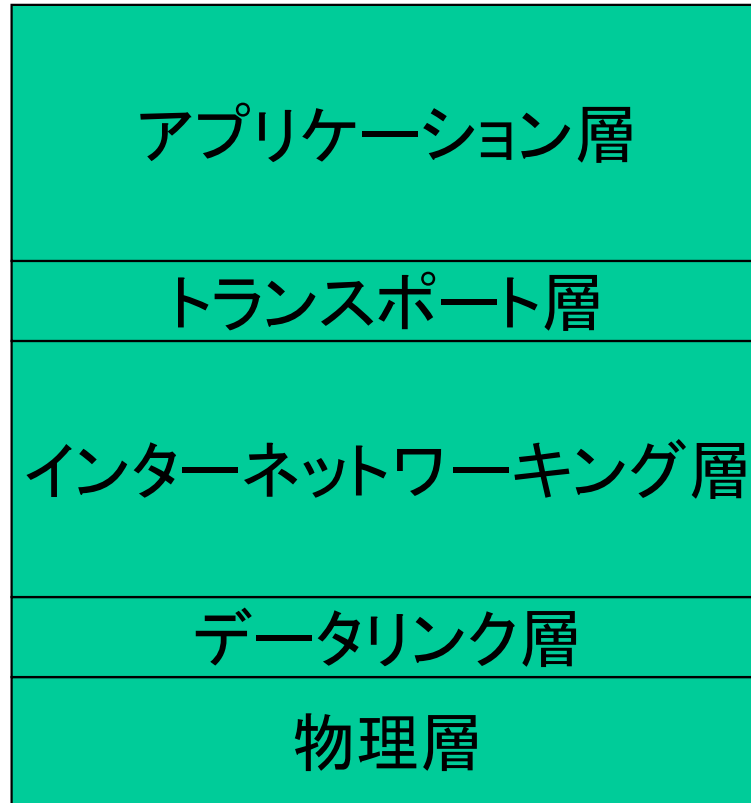
- インターネット内では帯域を管理しない
- みんながパケットを好きに送ると、パケット落ちが大量に発生
- パケット落ちがおきるかおきないかぎりぎりの速度で送ると、みんなが幸せ
- 紳士協定でしかないが
 - TCPに組み込まれて広く普及
 - 両端で協力しないと協定は破れず

混雑制御の実際

- インターネットの混雑状況に応じてTCPの速度を制御
 - 混んでいる場合、(送信側)ウィンドウを小さくする
- 混雑とは？
 - パケット落ち
- パケット落ちは、タイムアウトか、アクノレックジ番号が増加しないことにより検出

UDPでの混雑回避

- 一般論はない
 - 音声ストリームなどでは、そもそも無理
- パケット落ちの検出
 - タイムアウト
 - アプリケーションによっては特殊な方式も可能
- パケット落ちへの対処
 - パケットの再送信
 - パケット落ちが多いと再送信間隔を増やす



DNS(ドメイン名)、
RTP(ストリーム転送)、
UDP

インターネットのレイヤリング構造

DNS (Domain Name System) (RFC1034、1035)

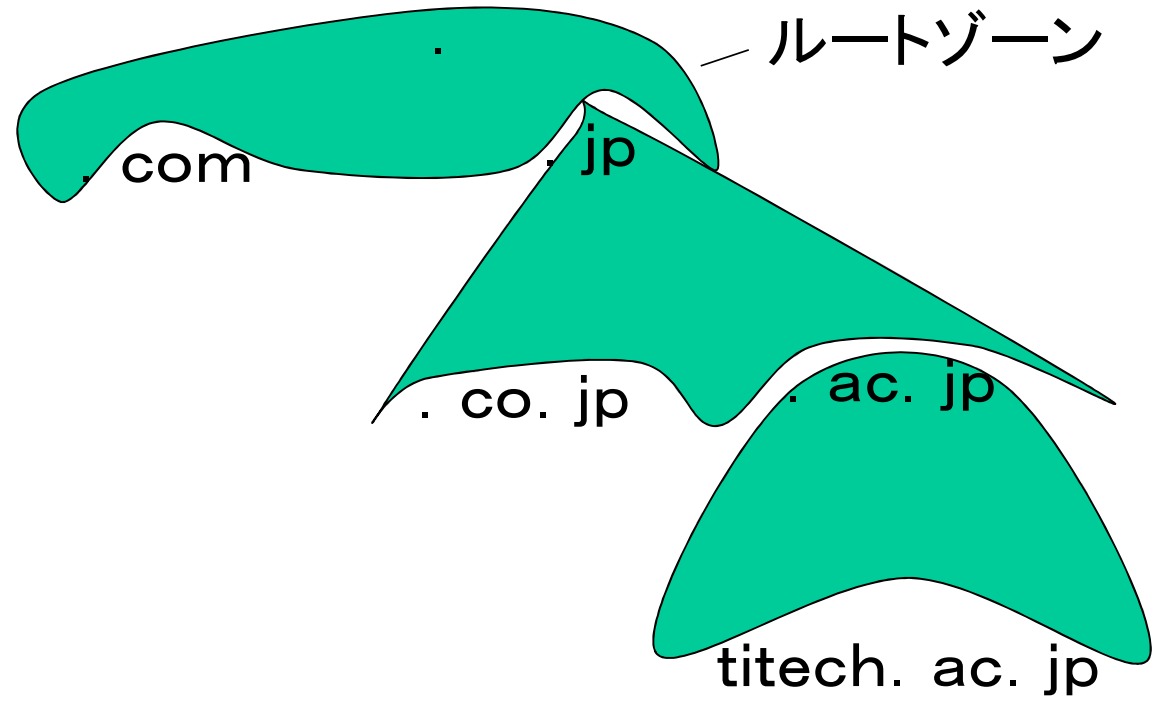
- ドメイン名からその値(IPアドレス等)を知るデータベース
 - ドメイン名は“.”で区切り階層的に管理
 - necom830. hpcl. titech. ac. jp.
- インターネットで最も重要なアプリケーションプロトコル
- もっぱらUDPを利用(TCPも一部で利用)
 - ポート番号は53

ドメイン名とネームサーバ

- “.”を根とする木
 - 最後の“.”は省略可能
- 木構造にしたがって、権限を委譲
 - nocom830. hpcl. titech. ac. jp.
 - IANA → JPNIC → 東工大 → 太田研
- 木構造の各部分(ゾーン)をそれぞれ複数のネームサーバで分担
 - 緩やかに統合された分散データベース

ネームサーバの序列

- 親のゾーンで子のネームサーバ情報を提供
- プライマリサーバ(1台)
 - ゾーン情報のマスターを保持
- セカンダリサーバ(1台以上)
 - ゾーン情報を他のサーバから取得
 - 他のサーバのバージョンを定期的にチェック
 - 更新があればゾーン転送



リゾルバとネームサーバ

- リゾルバとは
 - 上位アプリケーションからの依頼を受け、DNSデータベースを検索する機構
 - アプリケーションに組み込みのことも
 - ネームサーバとのパケットのやりとりを担当
 - データのキャッシュも行う
- 最初の問い合わせ相手は静的に知っている

リカーシブリゾルバ

- 一度で答えが返らない場合、他のネームサーバに再問い合わせするリゾルバ
- NS、CNAMEなどが再問い合わせの原因
- そうでないリゾルバがスタブリゾルバ
 - スタブリゾルバは、リカーシブリゾルバ機能を備えたネームサーバーを静的に知っている
 - キャッシュが集中

ノードの持つ情報

RR (Resource Record)

- RRは、
 - クラス
 - 使われてない
 - タイプ
 - IPアドレス、メールサーバ等
 - TTL
 - キャッシュすべき時間
 - データ
 - 各タイプに応じたデータ

DNS用RRタイプ

- SOA
 - ゾーンの情報
- NS
 - ネームサーバ
- CNAME
 - エイリアス
- (グループ)A
 - ネームサーバのIPアドレス

SOA (START OF AUTHORITY) RR (1)

- ゾーンの根で指定
 - プライマリネームサーバ
 - ゾーンのプライマリネームサーバのドメイン名
 - 管理者のメールアドレス
 - ドメイン名形式で
 - シリアル番号
 - ゾーン情報のシリアル番号

SOA RR (2)

- SOA
 - リフレッシュ間隔
 - シリアル番号をチェックする間隔
 - リフレッシュのリトライ
 - リフレッシュ失敗後のリトライ間隔
 - 有効期限
 - この期限の間は、リフレッシュできなくてもよい
 - 最小TTL
 - ゾーンのRRのTTLの最小値(デフォルト値)

ゾーン転送

- プライマリサーバからセカンダリサーバへのゾーン情報の転送
 - TCPを利用
- セカンダリサーバどうしでの転送も可能
 - 転送元は静的に設定
 - あいてのシリアル番号をまずチェック
 - あいてのほうが新しければ、ゾーン転送
- 電子メールやFTPでやってもよい

NS (NAME SERVER) RR

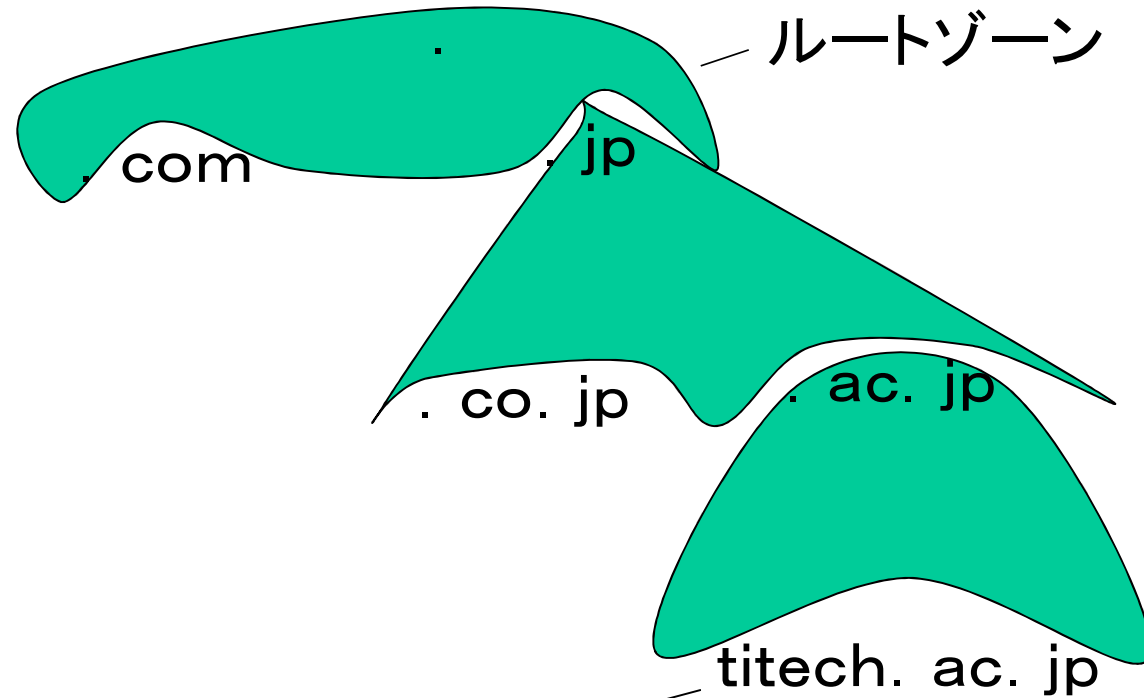
- NSのドメイン名を指定
- ゾーンの根で
 - そのゾーンのNSを指定
- ゾーンの末尾で
 - 次のゾーンのNSを指定
 - グルーAレコードが付随すること

CNAME (Canonical Name) RR

- あるドメイン名の別名を記述

グループA (ADDRESS) RR

- 子のNSが子のゾーン内にあると
 - 子のNSに問い合わせないと子のNSのアドレスがわからない
- 親のゾーンの末尾で子のNSのアドレスを指定



titech. ac. jp NS ns1. noc. titech. ac. jp
ns1. noc. titech. ac. jp A 131. 112. 125. 4

ネームサーバとグループA

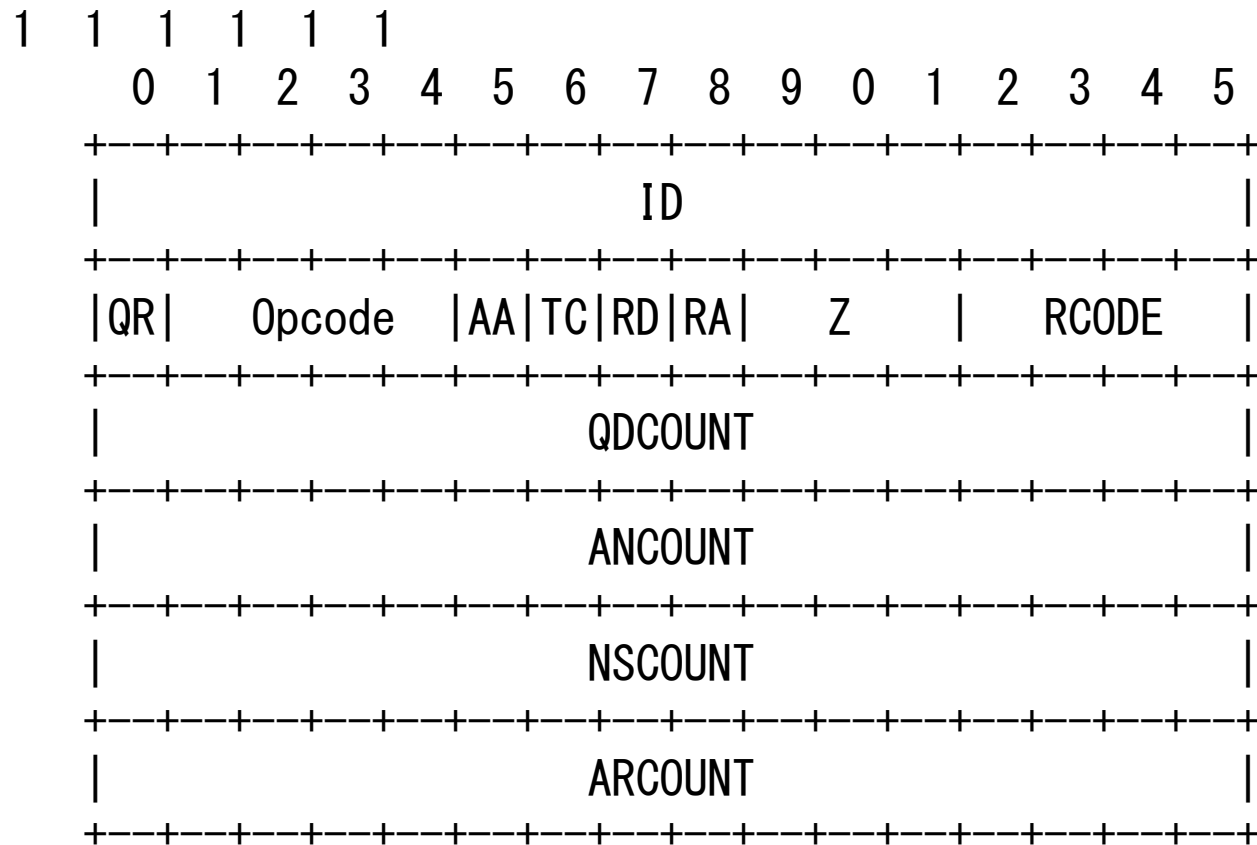
アプリケーション用RRタイプ

- A
 - IPv4アドレス
- MX
 - メールサーバのドメイン名
- PTR
 - IPアドレスからホスト名を逆引き

メッセージ(UDPのパayload) フォーマット

Header	
Question	the question for the name server
Answer	RRs answering the question
Authority	RRs pointing toward an authority
Additional	RRs holding additional information

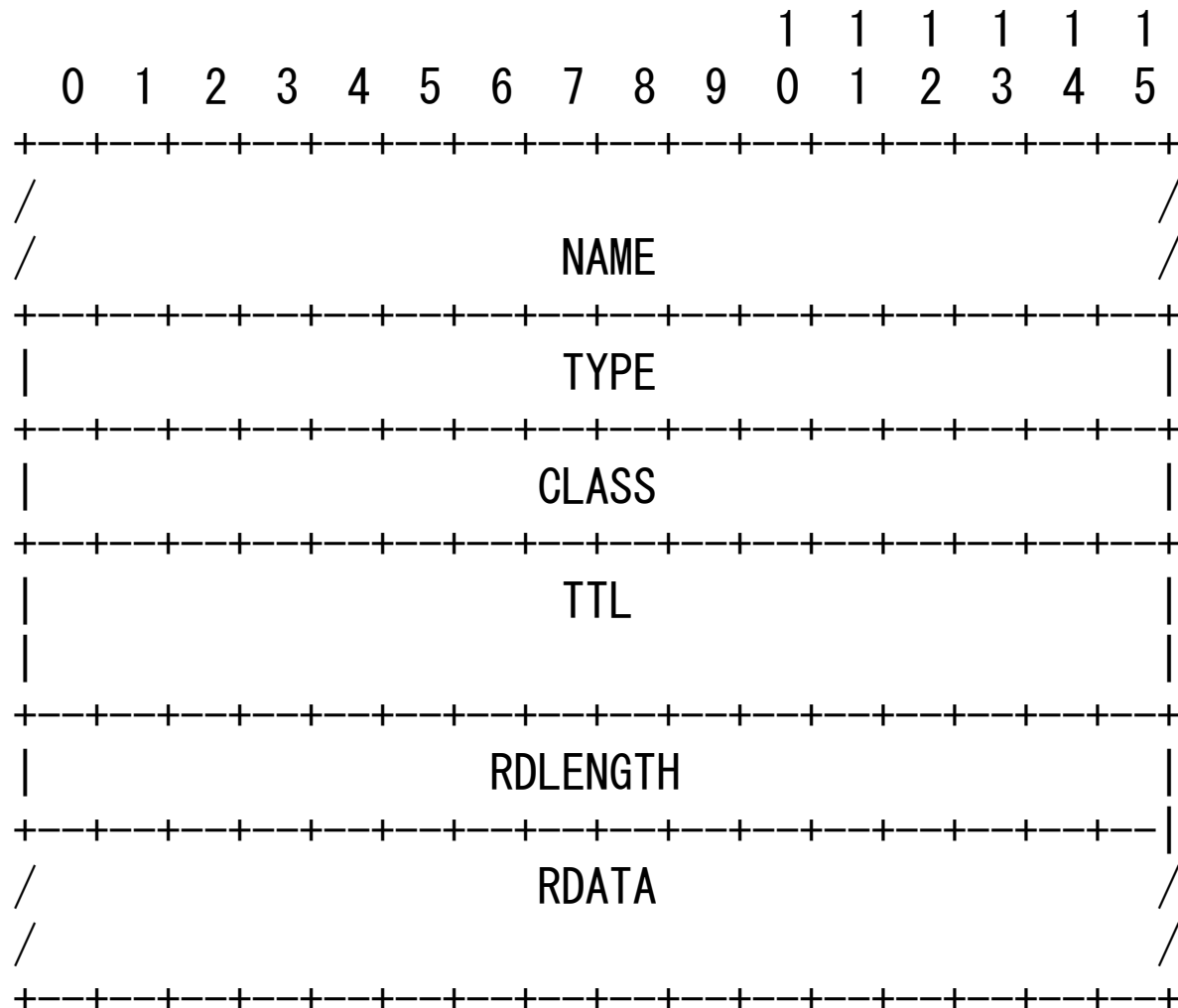
Header フォーマット



Question Section フォーマット

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	
+--+																
/	QNAME															/
/																/
+--+																
	QTYPE															
+--+																
	QCLASS															
+--+																

Resource Record フォーマット



DNSの仕組み

- アプリケーション
 - リゾルバーに検索を依頼
- リゾルバー
 - ネームサーバに問い合わせる
 - 中途半端な答えには再問い合わせ
 - スタブリゾルバーは、他のリゾルバに問い合わせ
- ネームサーバ
 - 問い合わせに単純に答える
 - リカーシブネームサーバは最終的な答を返す

DNSとキャッシュ

- リゾルバは一度検索した情報をキャッシュ
 - ネームサーバの負荷を減らす
- キャッシュの時間は、RRのTTLで個別に指定
 - 頻繁に更新される情報は、TTLを短く
 - めったに変わらない情報は、TTLを長く
 - 特に、“.”の情報
- Additional Sectionと共に負荷を低減

リゾルバの動作

- 答えがキャッシュにあれば
 - それを返す
- なければ
 - ネームサーバに問い合わせる
- 他のネームサーバにふられたら
 - そのネームサーバに再問い合わせ
- 答えがCNAMEなら
 - 新しい名前で再問い合わせ

ネームサーバの動作

- 自分の管理するゾーンなら
 - それを返す
- 答えがキャッシュにあれば
 - それを返す
- でなければ、
 - わかるかぎりの直上のドメイン(最悪“.”)を返す

DNSメッセージサイズ

- IPv4ホストは576バイト以下のIPパケットを受け取ればよい(フラグメントしても)
 - ヘッダーは最長60バイト
- UDPのペイロード長は508(512?)バイトまでしか保証されない
- 長いDNSデータはUDPで送れない
 - 複数パケットの順序付け? 再送?
- 長いDNSデータはTCPで送る

DNSとエンドツーエンド原理

- DNSはASCII文字列によりエンドシステム（のIPアドレス）を大域的に識別する
- DNSの木構造はネットワークトポロジーと無関係
 - 識別を端末周辺で局所的にやるのは不可能
 - ネットワーク中に特殊なサーバを置くしかない
 - エンドの位置とは無関係
 - つまり、エンドツーエンド原理に従っていない

DNSのエンドツーエンド 原理違反

- エンドツーエンド原理違反ということは
 - (エンドが落ちてなくても)サーバダウンに弱い
 - プロトコル自体で複数のサーバをサポート
 - エンドツーエンドマルチホーミング
 - サーバに負荷が集中
 - 特にルートサーバでは問題
 - UDPによる軽いプロトコルなので、なんとかなる？
 - Anycastルートサーバ？
 - 同一アドレスを共有する極めて多数のルートサーバを持ち、ルーティングシステムにより最寄りのものを選択

DNSの拡張

- IXFR (Incremental Zone Transfer)
- Notify
- DYNUPD (Dynamic Update)
- Secure DNS
- AROOT (Anycast Root Servers)

IXFR (Incremental Zone Transfer) (RFC1995)

- ゾーンの更新があったとき
 - これまでのDNSではゾーンの全情報を転送
 - 差分だけを転送したい
- これまでのDNSのゾーン転送のフォーマットを拡張
 - SOAにより区切り
- 差が小さいときは、UDPによるゾーン転送も可能

IXFRの例

JAIN.AD.JP. IN SOA serial=3
JAIN.AD.JP. IN SOA serial=1
NEZU.JAIN.AD.JP. IN A 133.69.136.5
JAIN.AD.JP. IN SOA serial=2
JAIN-BB.JAIN.AD.JP. IN A 133.69.136.4
JAIN-BB.JAIN.AD.JP. IN A 192.41.197.2
JAIN.AD.JP. IN SOA serial=2
JAIN-BB.JAIN.AD.JP. IN A 133.69.136.4
JAIN.AD.JP. IN SOA serial=3
JAIN-BB.JAIN.AD.JP. IN A 133.69.136.3
JAIN.AD.JP. IN SOA serial=3

NOTIFY (RFC1996)

- ゾーンの変更を他のサーバに通知する
- これまでは、セカンダリは他のサーバのSOAをリフレッシュ間隔で定期的にチェック
- NOTIFYは、変更があった場合、プライマリ側からセカンダリに通知

DYNUPD (DYNAMIC UPD ATE)

- ゾーンの内容を変更するプロトコル
 - これまでは、ゾーンファイルを手で編集
- 必要以上に複雑
- セキュリティが大問題
 - SECURE DYNUPD?
- DYNUPD – > NOTIFY – > IXFR

AROOT

(Anycast Root Servers)

- ルートサーバーには負荷が集中
- ルートサーバーはDNSセキュリティの要
- ルートサーバーを各自で運営しよう！
 - 同じIPアドレス(エニキャストアドレス)を共有
 - 手近のサーバーが使われる
 - ISP間での運用には工夫(エニキャストASの導入)が必要

セキュリティとは？

- 秘匿
 - 情報を第三者の目から隠すこと
 - 暗号化
- 認証
 - 身分証明
- 相互に関連
 - 認証のための情報は秘匿しなければならない
 - 認証された人には情報を秘匿しなくていい

秘匿の方式

- 共有秘密鍵
 - 暗号文 = $E(\text{平文}, \text{秘密鍵})$
 - 平文 = $D(\text{暗号文}, \text{秘密鍵})$
- 公開鍵
 - 暗号文 = $E(\text{平文}, \text{公開鍵})$
 - 平文 = $D(\text{暗号文}, \text{秘密鍵})$

認証の方式

- ハッシュ関数(疑似乱数)を利用
- 共有秘密鍵
 - 認証情報 = $H(\text{平文}, \text{秘密鍵})$
- 公開鍵
 - 認証情報 = $A(H(\text{平文}), \text{秘密鍵})$
 - $D(\text{認証情報}, H(\text{平文}), \text{公開鍵})$

WEAK SECURITY

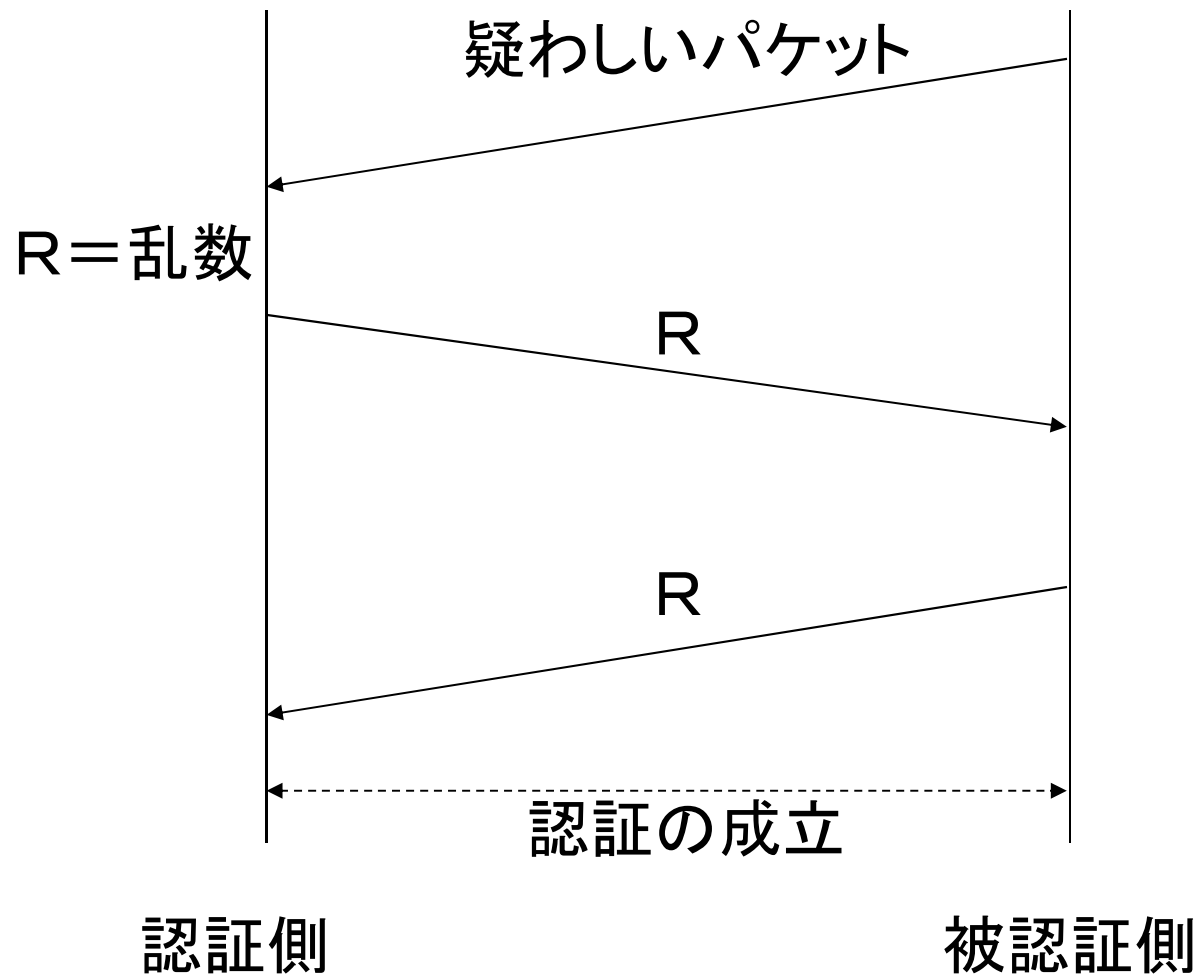
- インフラを無条件に信頼するセキュリティ
 - 自分と相手の間のISPが信頼できれば、指定したIPアドレスの相手とセキュアに通信できる
- 電話網のセキュリティも、この程度
 - 自分と相手の間の電話会社が信頼できれば、指定した電話番号の相手とセキュアに通信できる

電話網のセキュリティ

- 電話事業者は信頼できると仮定
 - ある電話番号にかければ、その電話番号の電話につながる
 - 途中で盗聴もない
 - 電話番号により相手を特定可能
- あくまで仮定にすぎない

インターネットの WEAKセキュリティ

- インターネット事業者は信頼できると仮定
 - あるIPアドレスにパケットを送れば、そのIPアドレスの相手に届く
 - 途中で盗聴もない
 - IPアドレスにより相手を特定可能
 - ただし、あるパケットが届いても、そのソースIPアドレスからきたパケットとは限らない
 - ハンドシェークによる確認(自分の送った疑似乱数が相手から帰ってくる)は必須



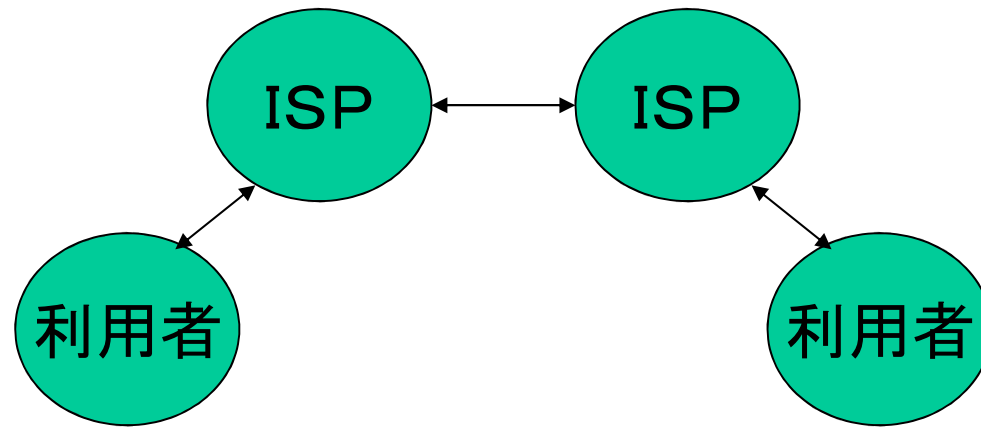
WEAKセキュリティのための相手IPアドレスの認証

まともな(強い)セキュリティ？

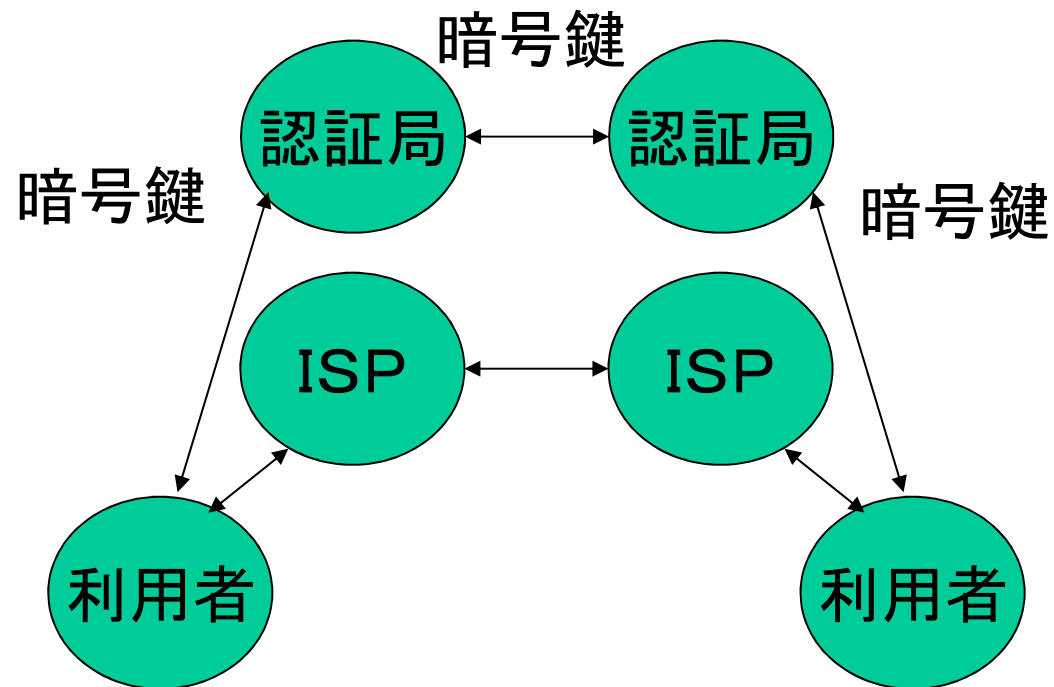
- 秘密情報(鍵)を共有
- 鍵をどう交換するかが問題
 - PKI？
 - Public Key Infrastructure
 - DNS？
 - DNSで鍵を配るには
 - DNSデータが認証されないと駄目

そもそも公開鍵暗号系は 無意味では？

- 共有鍵の数(N^2)は問題ではない
- 共有鍵暗号系でも鍵配布局を利用すれば
認証局と同様のことが可能
 - ただし、鍵配布のたびに通信は必要
 - インターネットでは通信は無料
- 認証局は信頼するしかない
 - ISPを信頼するのと同じ
 - Weak Securityでしかない



ISPを利用した通信



認証局を利用した暗号化通信

信用とエンドツーエンド原理

- 暗号は信用を運ぶための道具にすぎない
 - 信用とは、例えば与信限度額
- 信用は一対一で形成される
 - 同時に共有鍵の交換が可能
 - 認証局は信用を提供しない
 - 認証局は中間にある無用の長物
- 信用するエンド同士が直接秘密鍵を共有するのが、インターネットのあるべき姿

DNSのセキュリティ

- 通常のDNSは、WEAKLY SECURE
 - メッセージのID(16ビット)が暗号
 - ここは、かなりINSECURE
 - インフラ(NSの木とそことの間のISP)が信用できればセキュア
 - ルートNSとその間のISPが信用できればルートNSの答え(下位NS情報)は信用できる
 - あるNSとその間のISPが信用できればそのNSの答え(下位NS情報、最終的答え等)は信用できる

SECURE DNS (RFC2535)

- DNSに本当のセキュリティ(認証)を
- 公開鍵暗号の木構造とゾーンの木構造を対応づけ
 - ルートゾーンの公開鍵を共有
 - 親のゾーンの公開鍵で、子のゾーンの公開鍵の認証を順次確認
 - DNSの階層と認証の階層が一致していれば有用
- まったく普及していない

まとめ

- インターネットとは情報通信インフラ
 - アプリ(ウェブ、電子メール)のことではない
- エンドツーエンド原理がインターネットの基本
- IPはインターネットの基本プロトコル
- UDPは軽い
- DNSはUDPの上のプロトコル
 - 公開鍵暗号系は、使えそうで使えない