インターネットインフラ特論
7. NAT、DHCP

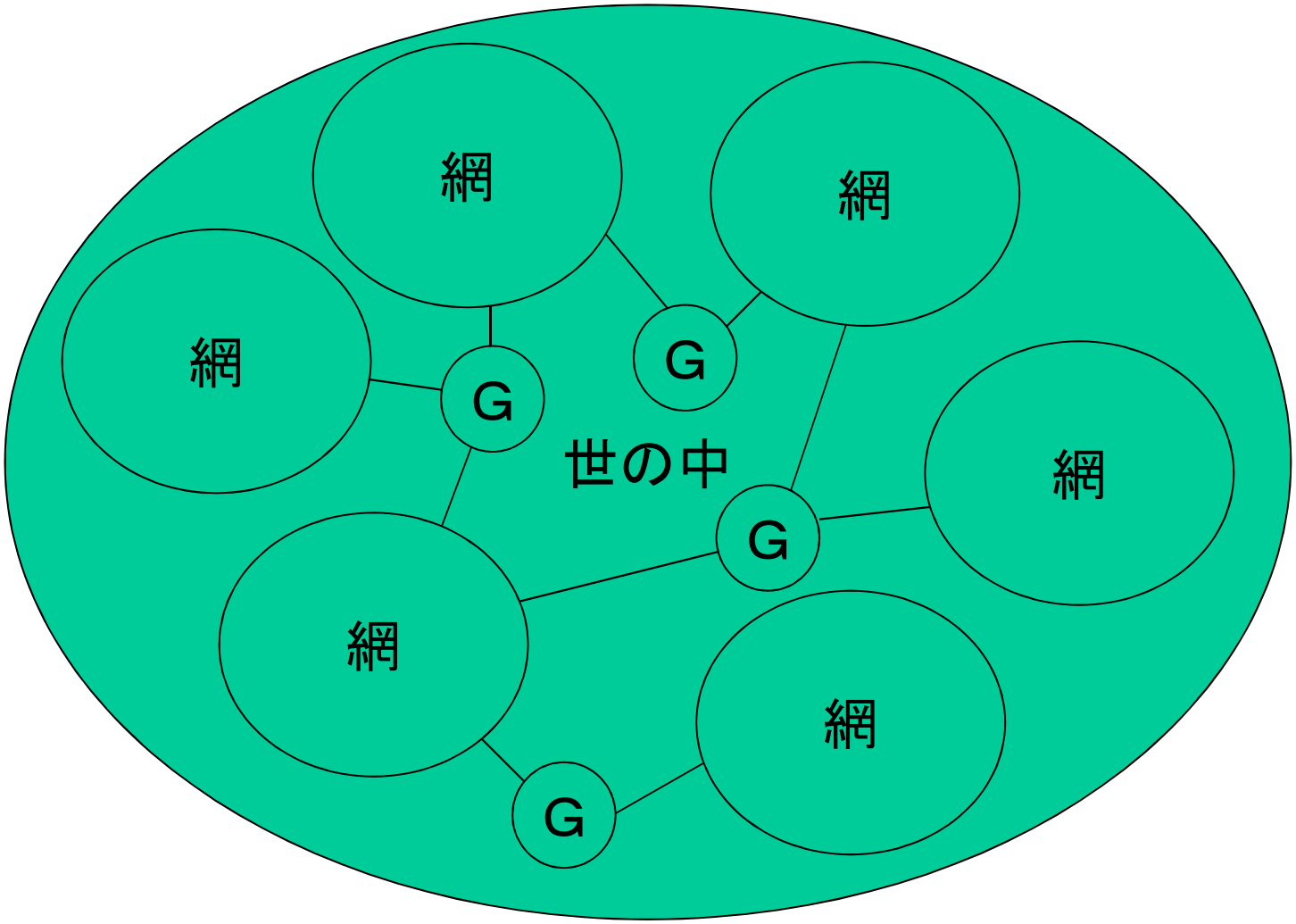
太田昌孝

mohta@necom830.hpcl.titech.ac.jp

ftp://chacha.hpcl.titech.ac.jp/infra7.ppt

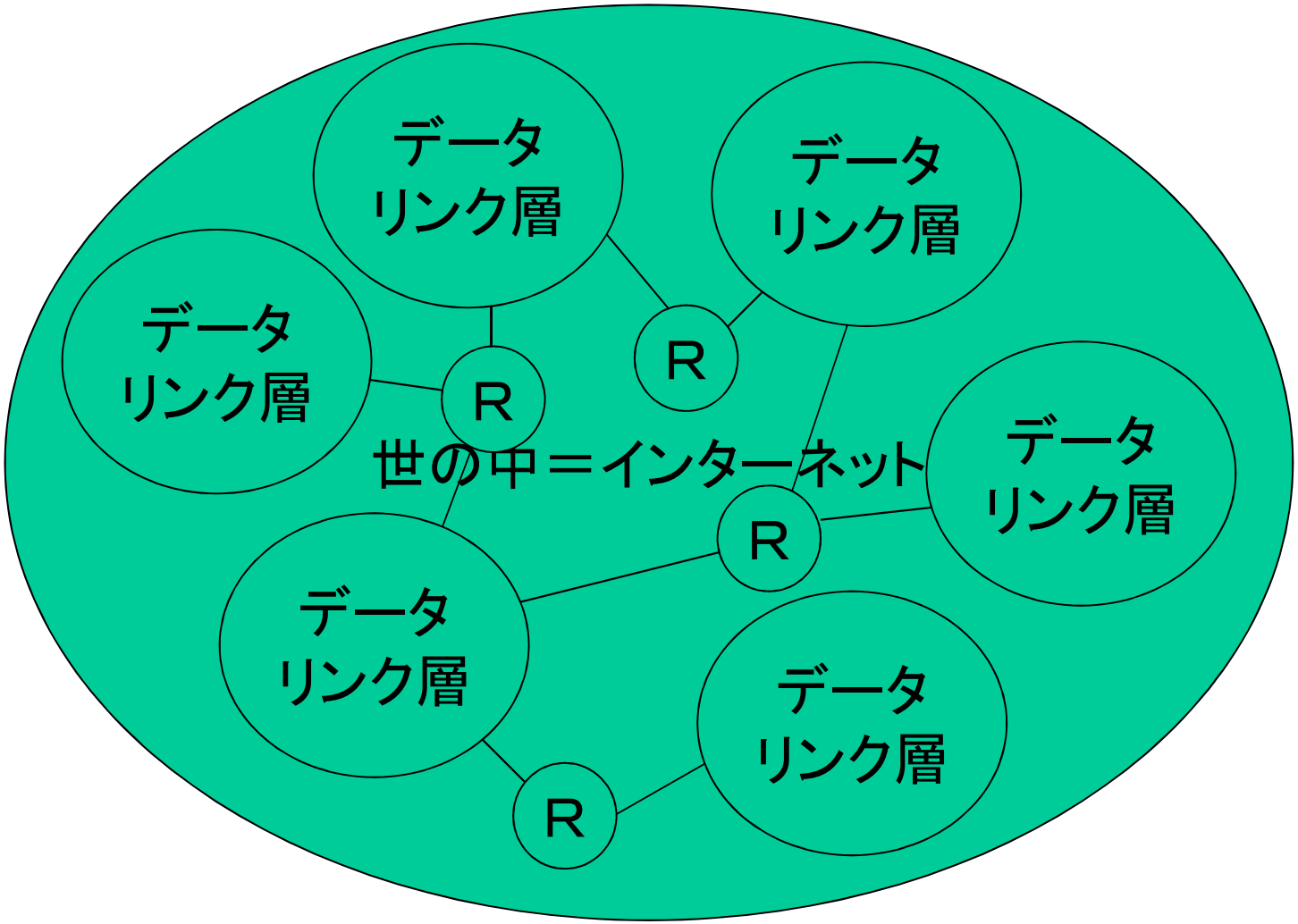
インターネットの構造

- 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



G　：網間接続装置

世の中とネットワーク層（インターネット以前）



R　：ルータ

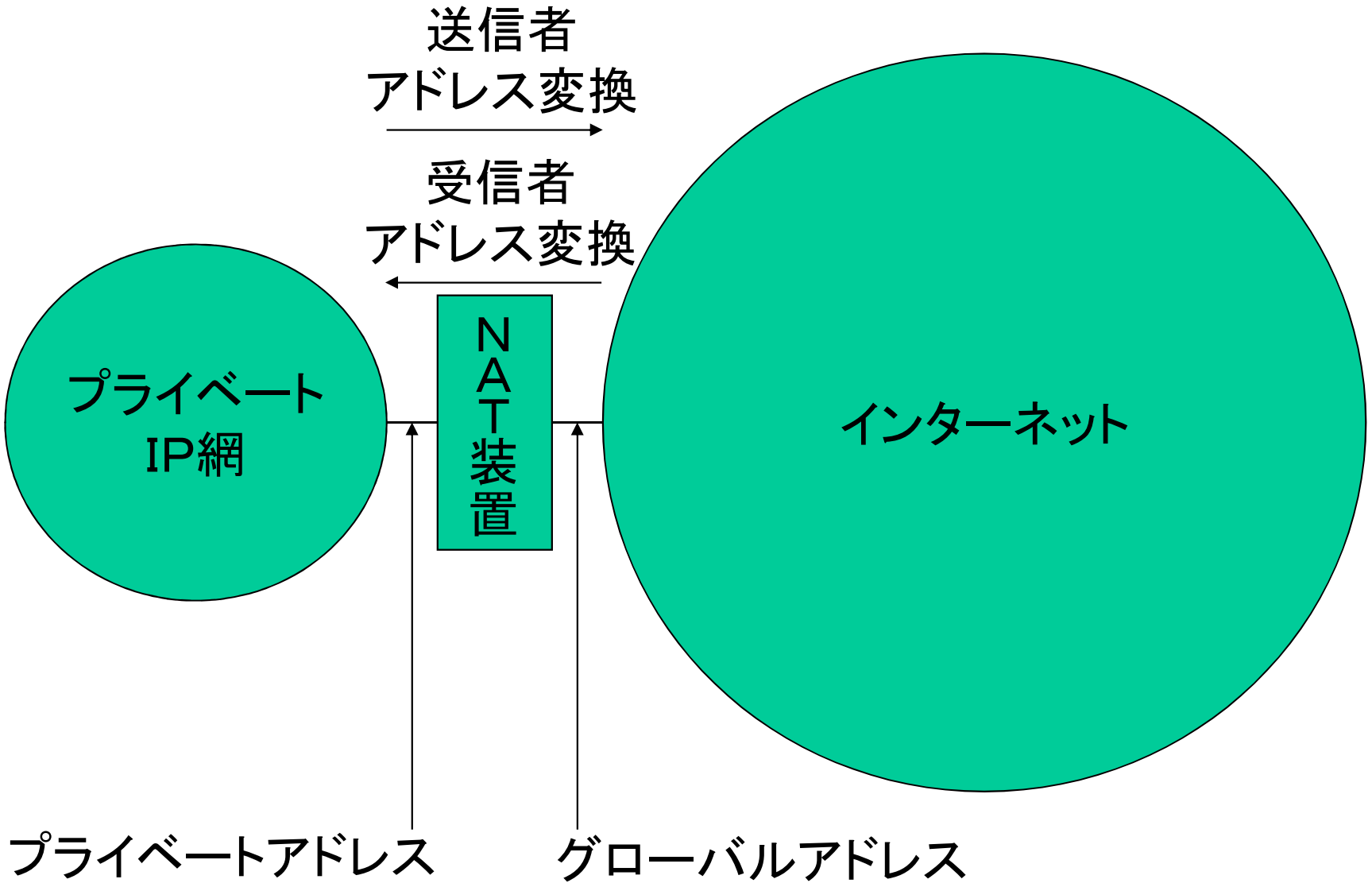
CATENETモデル

NAT （Network Address Translator、 RFC1631）

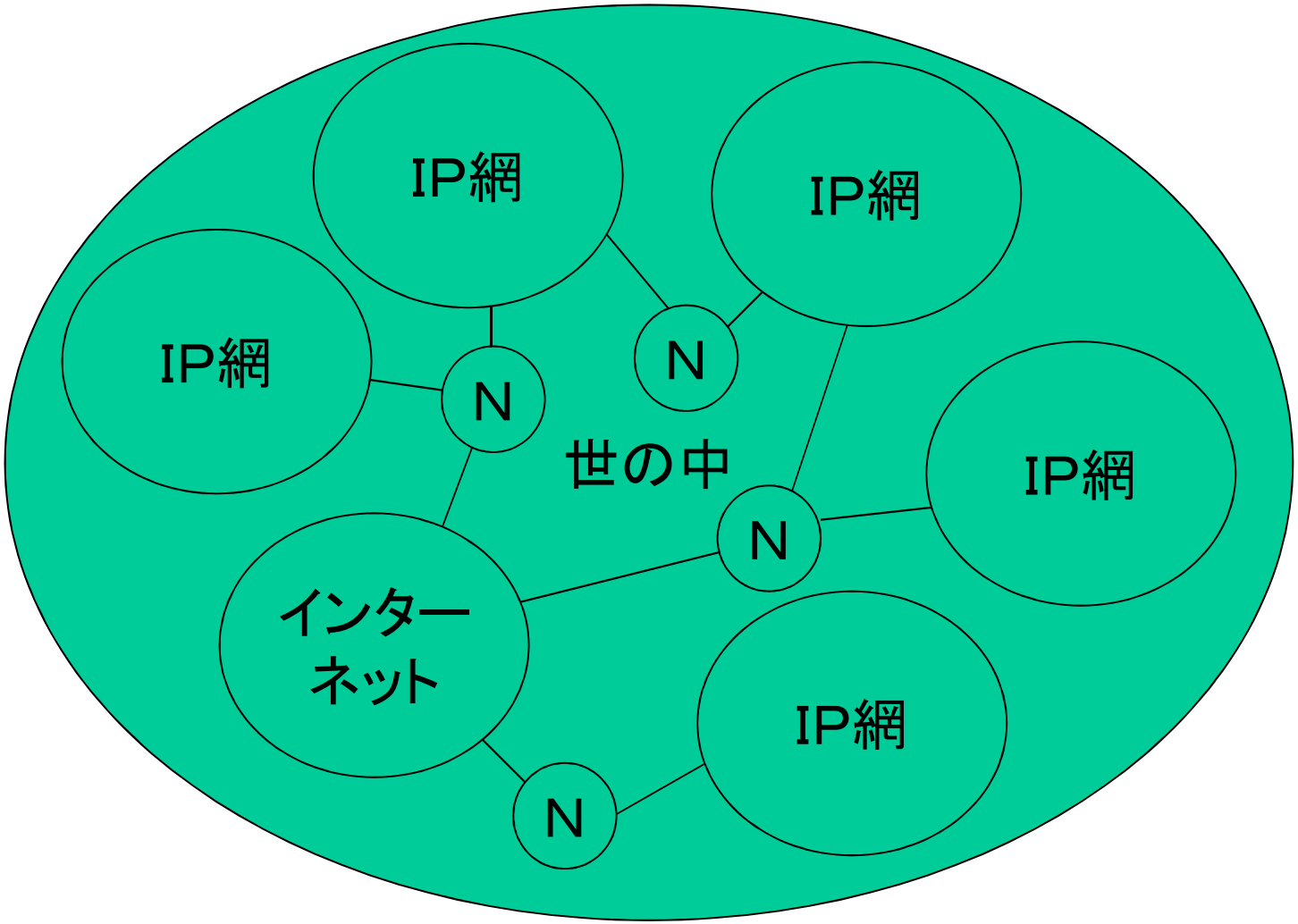
- インターネットとプライベートIPネットワークの境界に設置
- 少数のインターネットアドレスを管理
  - プライベートIPネットワーク中のアドレスをインターネットアドレスに（動的に）対応付け
  - IPv4アドレスの不足から生まれた考え
    - インターネットと固定アドレスで通信する必要のあるホスト（サーバ）は多くない？

ダイアルアップとNAT

- ダイアルアップが優勢な時代には
  - 一般利用者は断続的にインターネットに接続
    - 一般利用者はクライアントしかもてない
    - サーバは特別なISPが持つ（別料金）のが当然
  - 常時接続している企業も
    - サーバはごく少数で、NATでアドレス節約
  - ISPがインターネットを支配
- 常時接続時代には通用しない（はず）

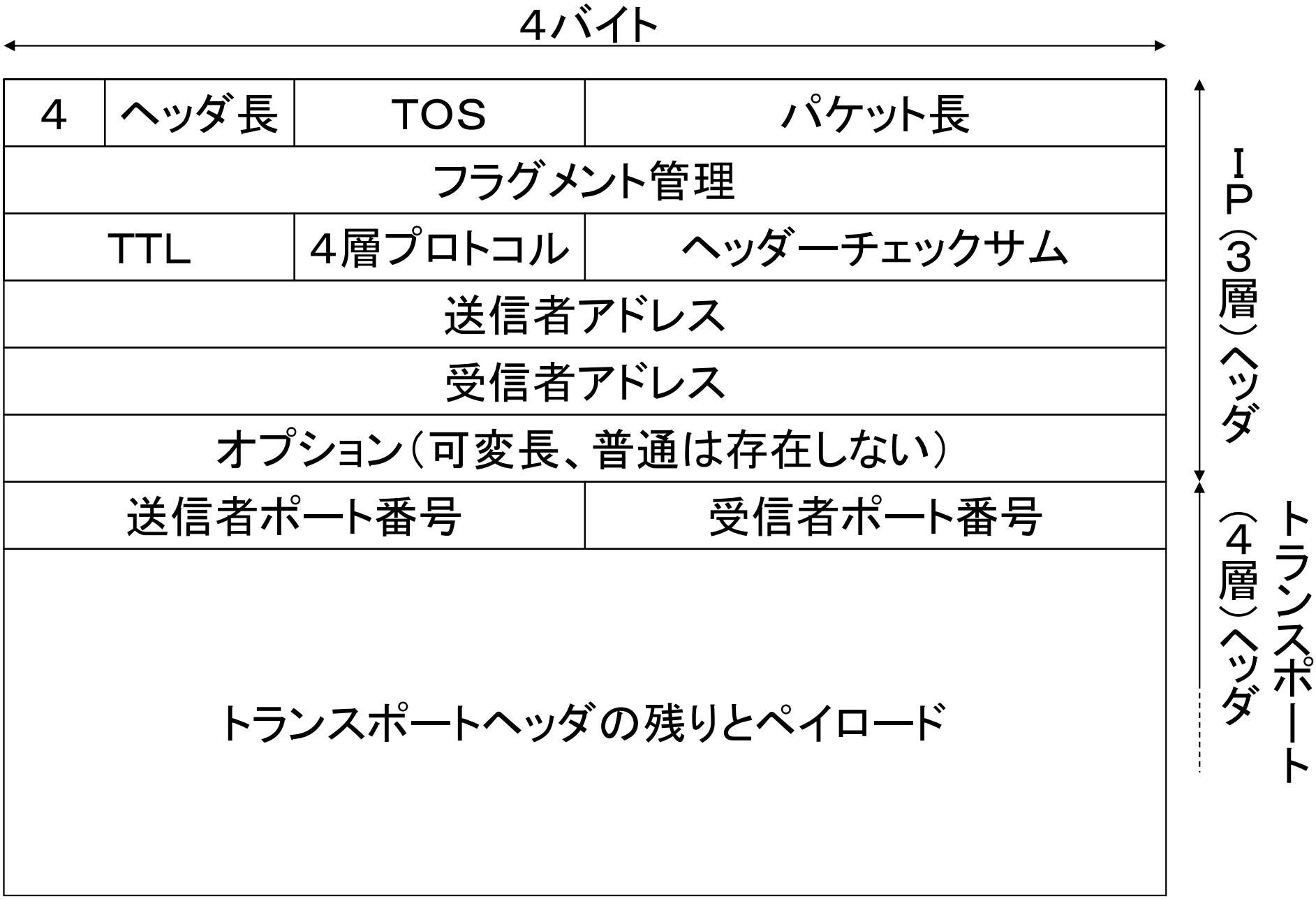

送信者
アドレス変換
受信者
アドレス変換
プライベート
IP網
NAT装置
インターネット
プライベートアドレス
グローバルアドレス

NATの仕組み

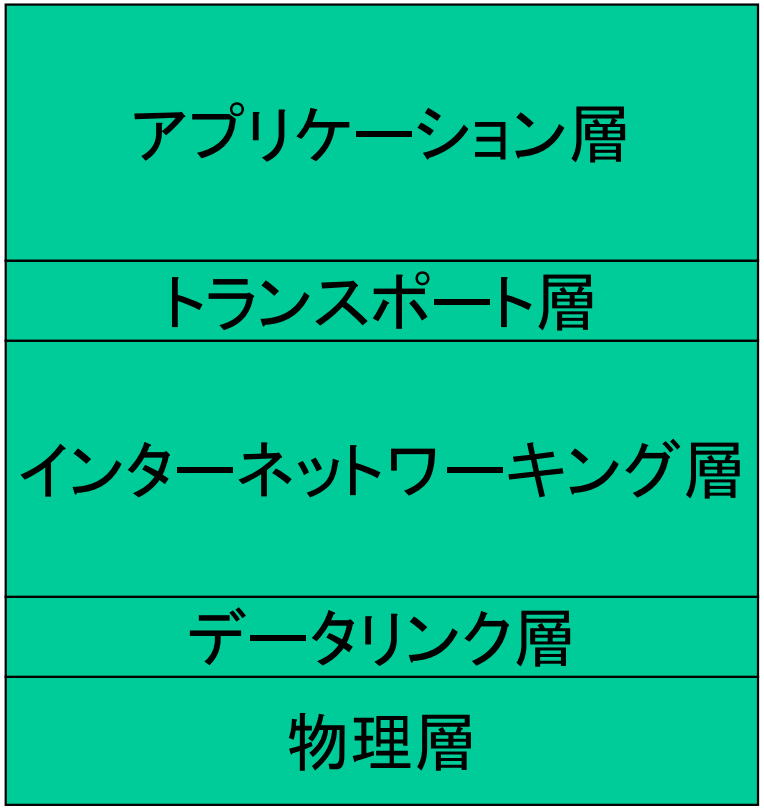


N　：NAT装置

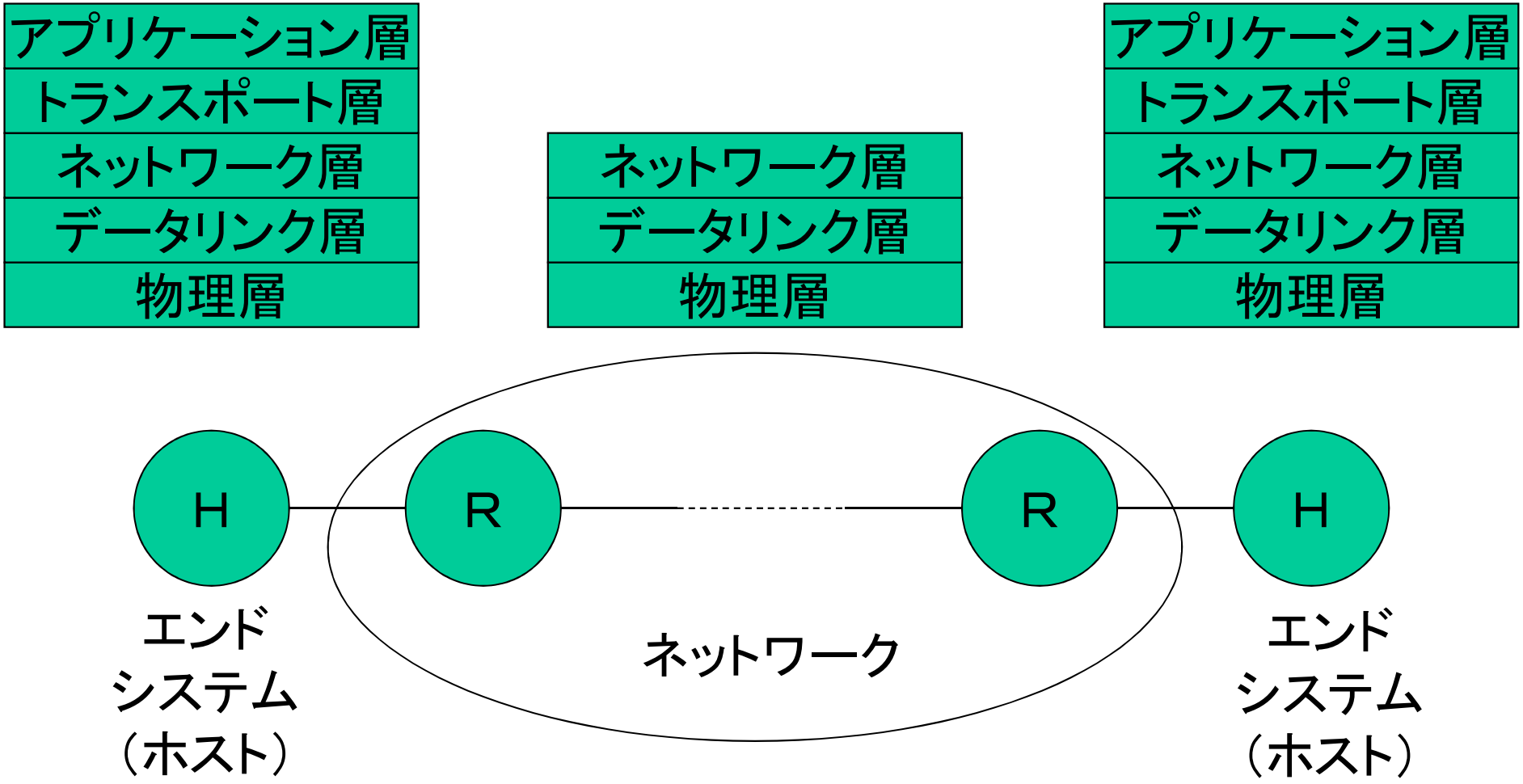
NATとインターネット

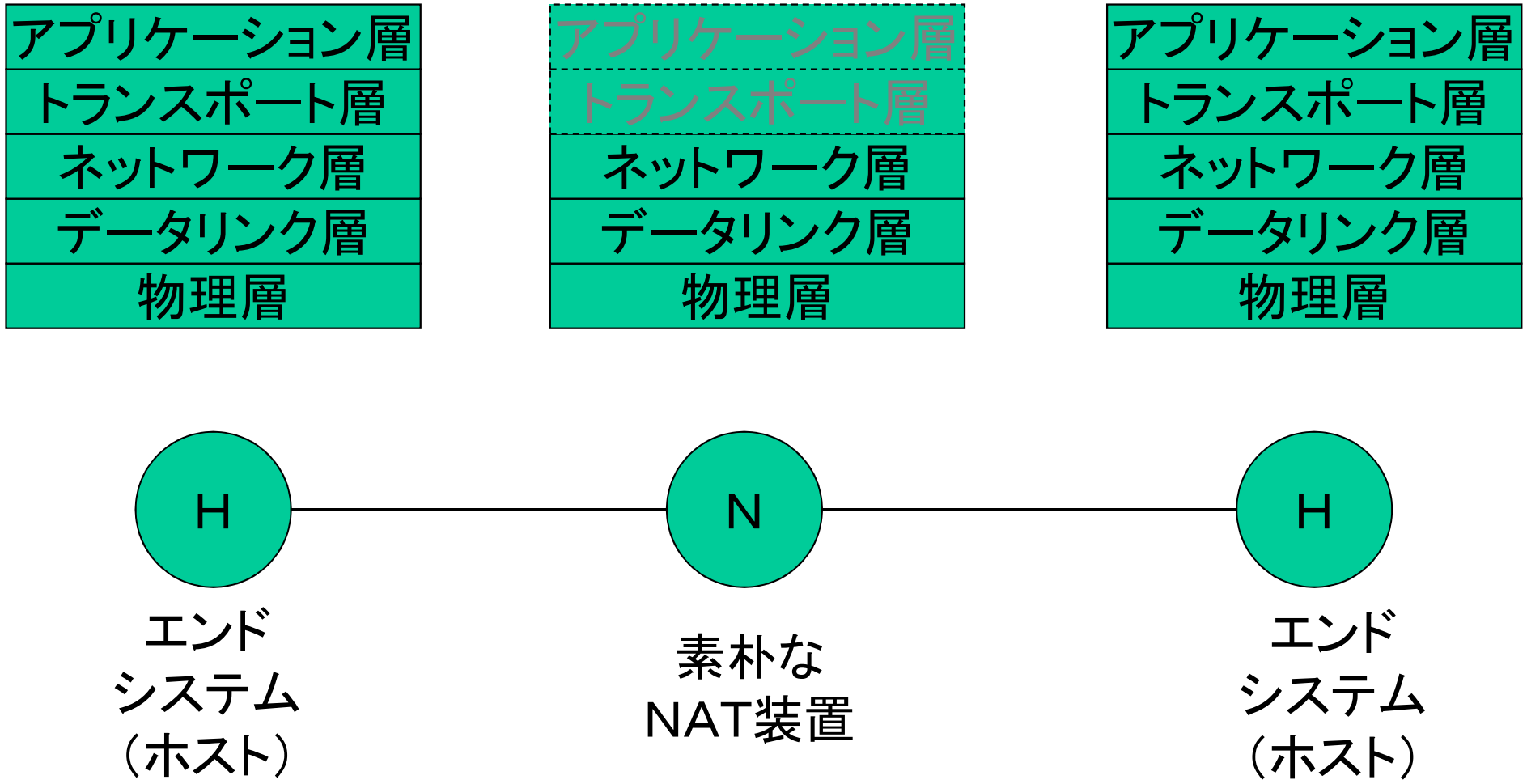


IPv4パケットフォーマット（RFC791）



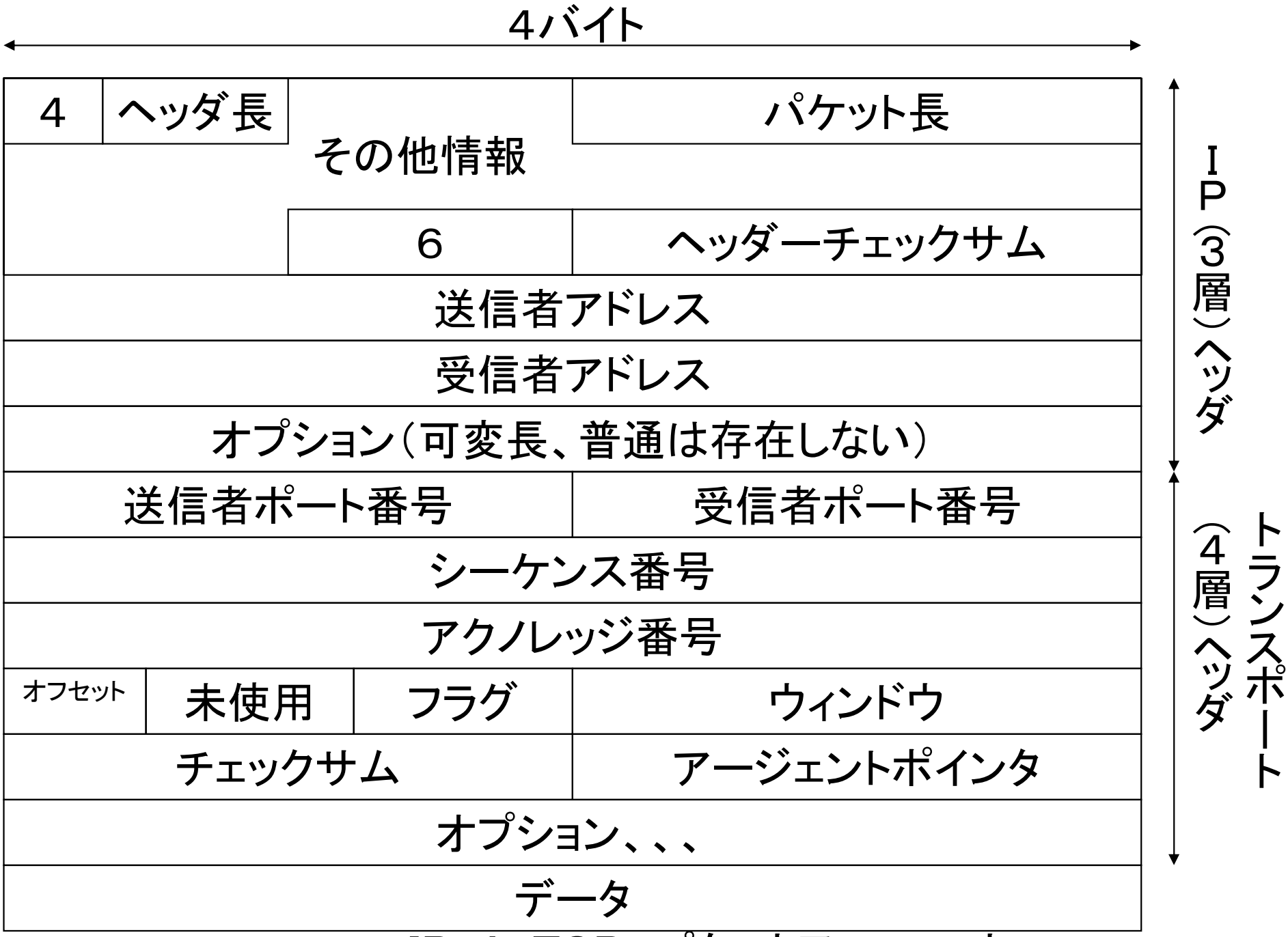
インターネットのレイヤリング構造


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
H
R
R
H
エンド
システム
（ホスト）
ネットワーク
エンド
システム
（ホスト）

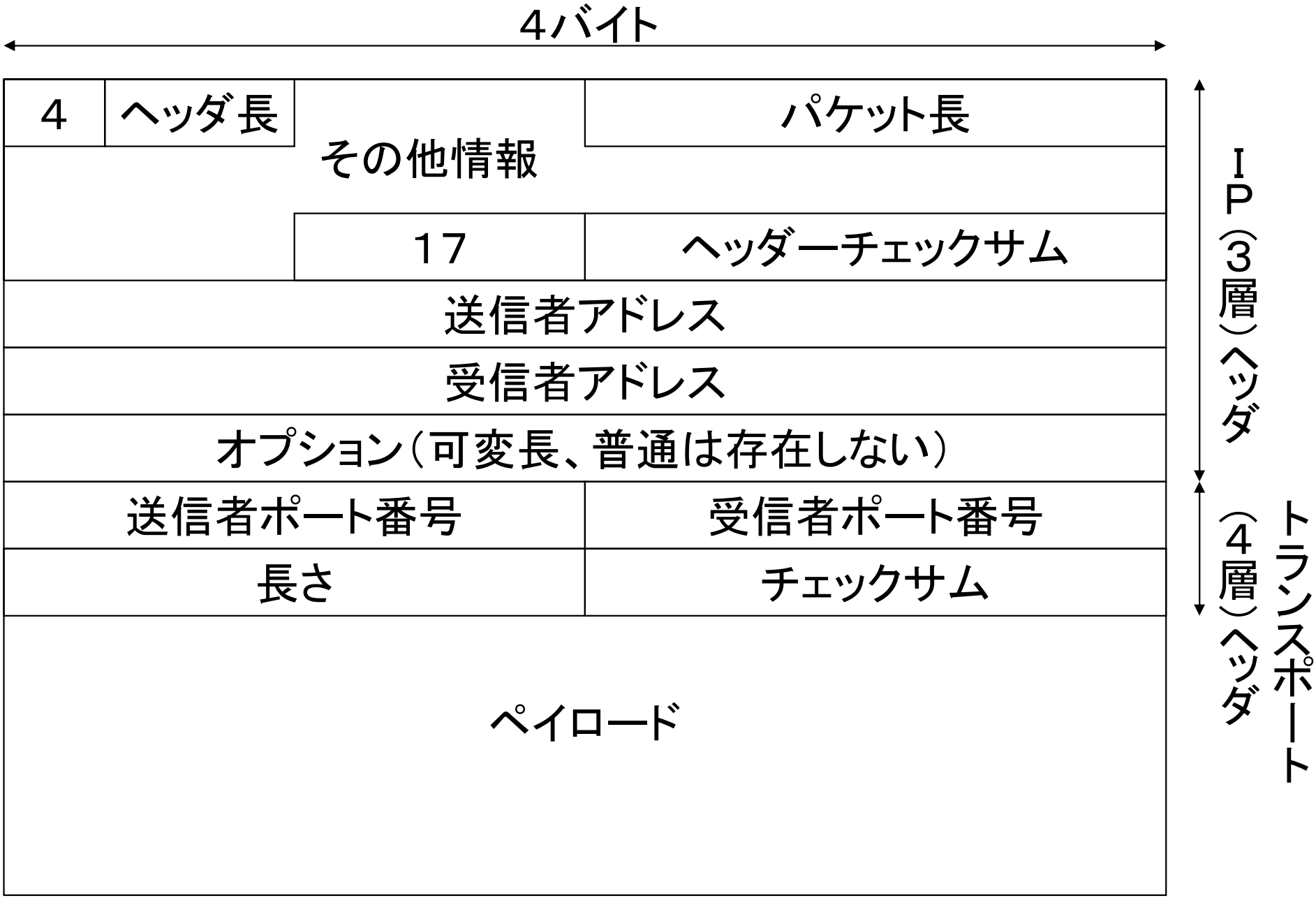

アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層
アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層
アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層
H
N
H
エンド
システム
（ホスト）
素朴な
NAT装置
エンド
システム
（ホスト）

素朴なNAT

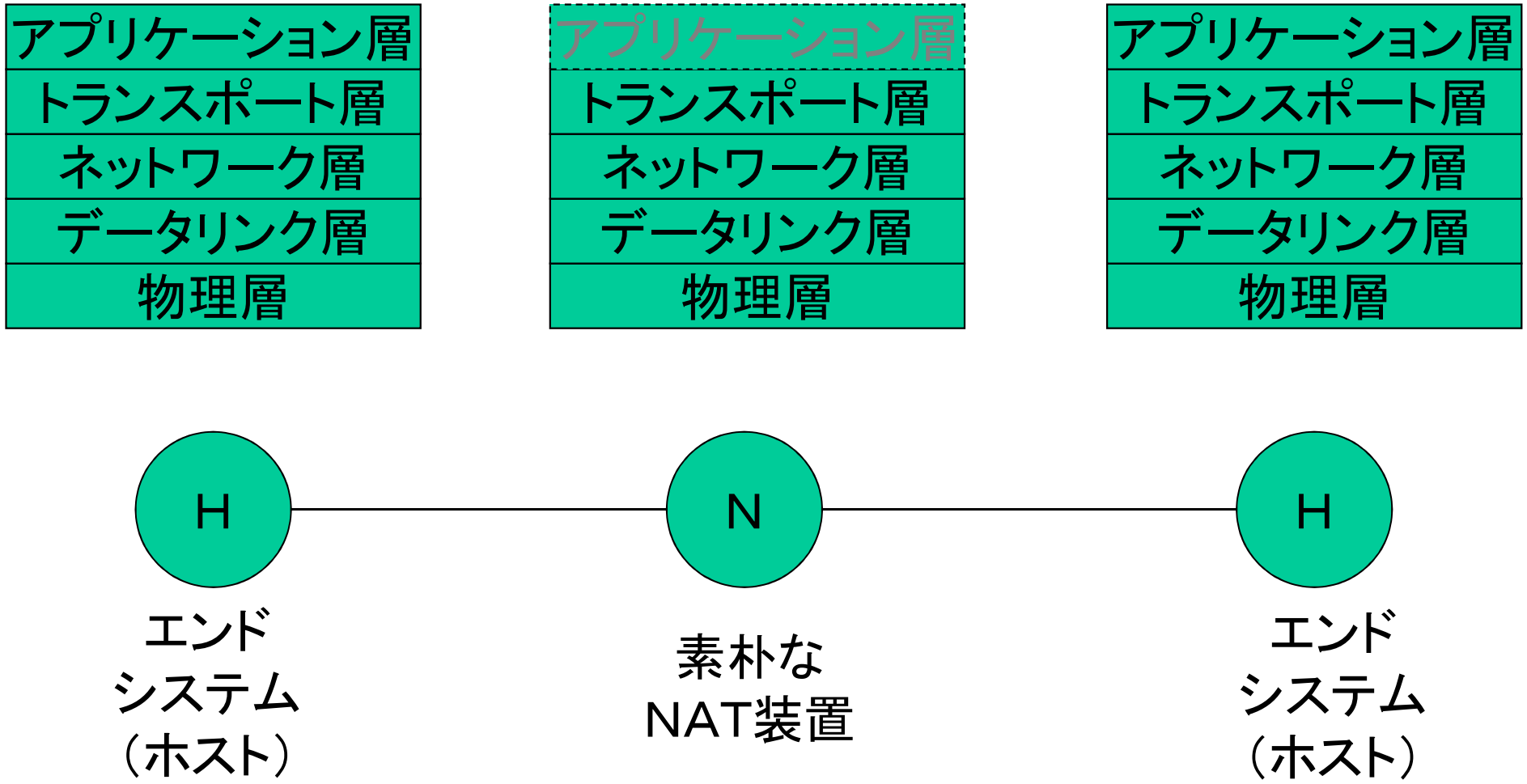
- アドレスをつけかえ
  - IPヘッダーをいじるだけ？
    - 送信者、受信者アドレス、チェックサム
    - あまりエンドツーエンド原理違反ではない？
    - ペイロードにあるアドレスは変換できない
      - FTP、ICMP等、アプリケーション毎には対応せず
  - トランスポートヘッダーのチェックサムも変更
    - NAT装置はトランスポートプロトコル（のチェックサムの計算方法）も知る必要
      - 新たなトランスポートプロトコルはNAT装置をとおらない



IPv4　TCP　パケットフォーマット

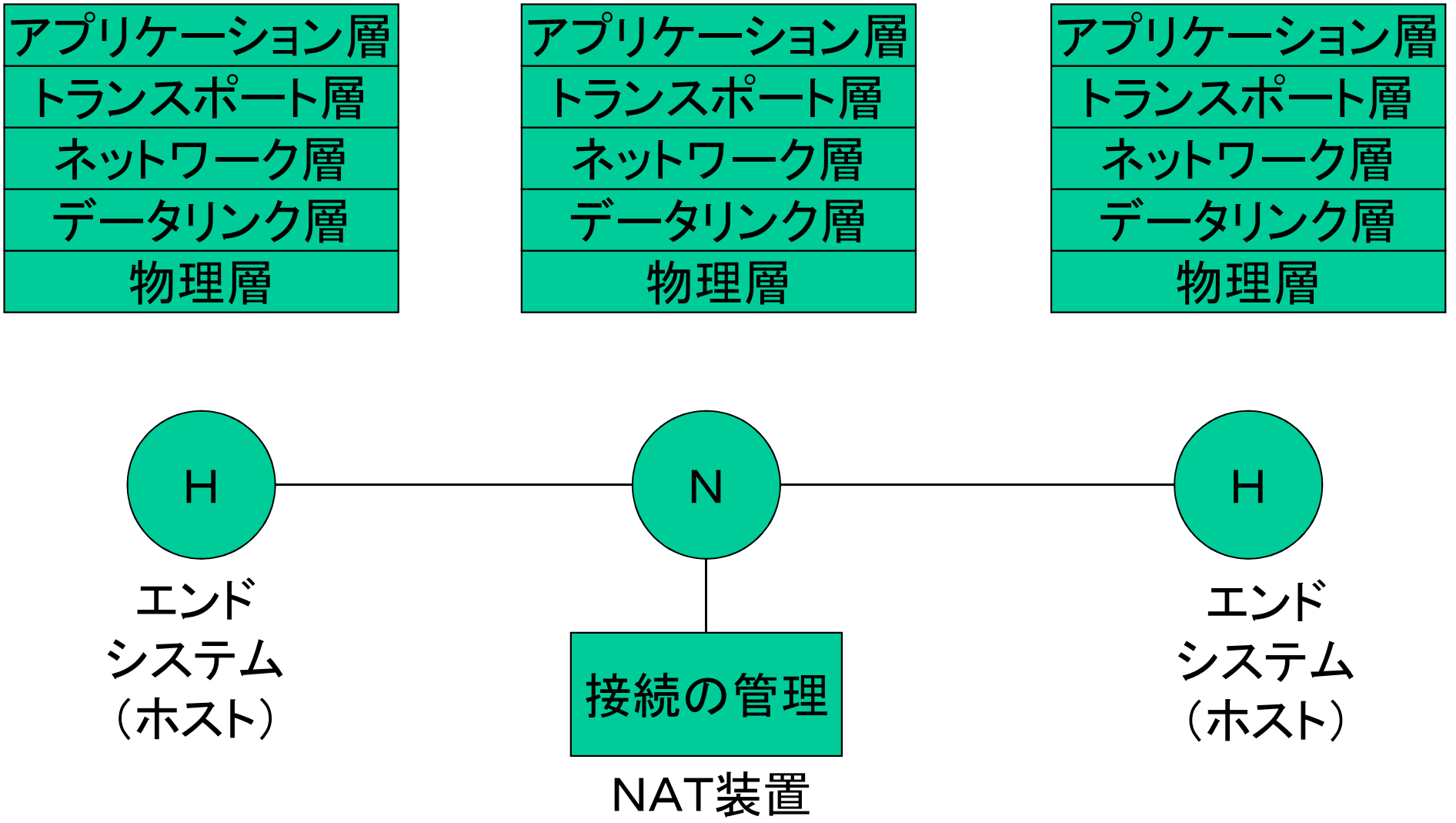


IPv4　UDP　パケットフォーマット


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
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
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
H
N
H
エンド
システム
（ホスト）
素朴な
NAT装置
エンド
システム
（ホスト）

NATによるアドレスの節約

- プライベートIP網内のホストのうち
  - 動作しているものにだけグローバルアドレスを割り当て
  - どのホストが動作しているか判断できない
    - タイムアウトにより管理？
      - 不正確
    - TCPパケットを監視して、開始、終了を判断
    - UDPには無力
      - UDP上のアプリケーションごとに対応

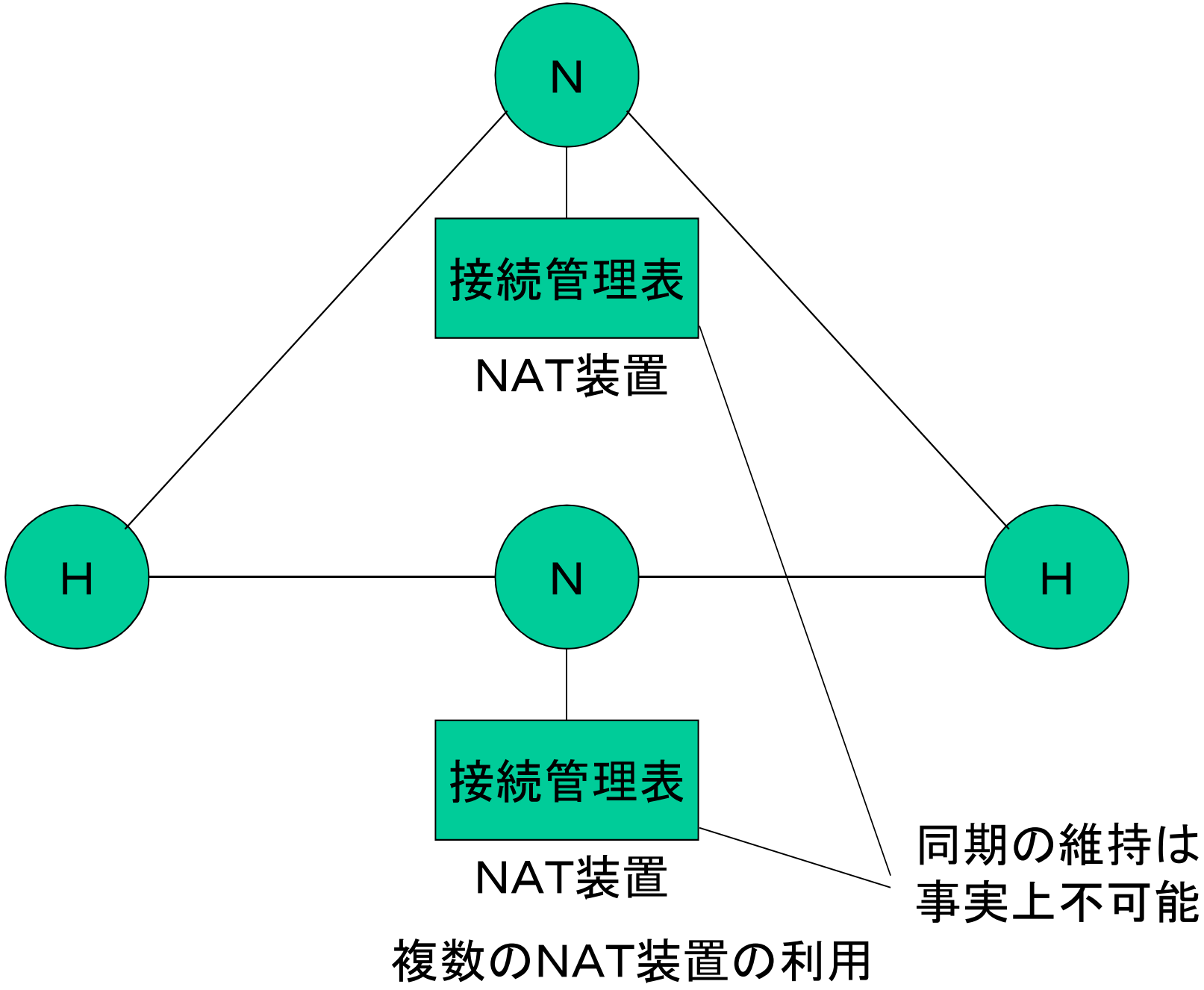

アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層
アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層
アプリケーション層
トランスポート層
ネットワーク層
データリンク層
物理層
H
N
H
エンド
システム
（ホスト）
接続の管理
NAT装置
エンド
システム
（ホスト）

本格的なNAT

- 各種アプリケーションプロトコルに対応して接続を管理
  - ICMPメッセージは適宜変換
  - FTPのコマンド列中のアドレスも変換
  - 各種UDPプロトコルにも対応
  - ポート番号まで変換
    - 常時接続では、アドレス使い回しは節約にならず
    - １つのグローバルアドレスを、複数のホストで共有

NATの問題点

- 新たなアプリケーションごとにNAT装置の対応が必要
  - 新たなアプリケーションは実質的に導入不可
- エンドツーエンド原理違反
  - 遅い（パケットの内容の監視は大変）
  - 冗長性なし、負荷が集中
    - 複数のNAT装置を並列に置いても
      - 相互の接続の管理に整合性がとれない



複数のNAT装置の利用

NATとセキュリティ（？）

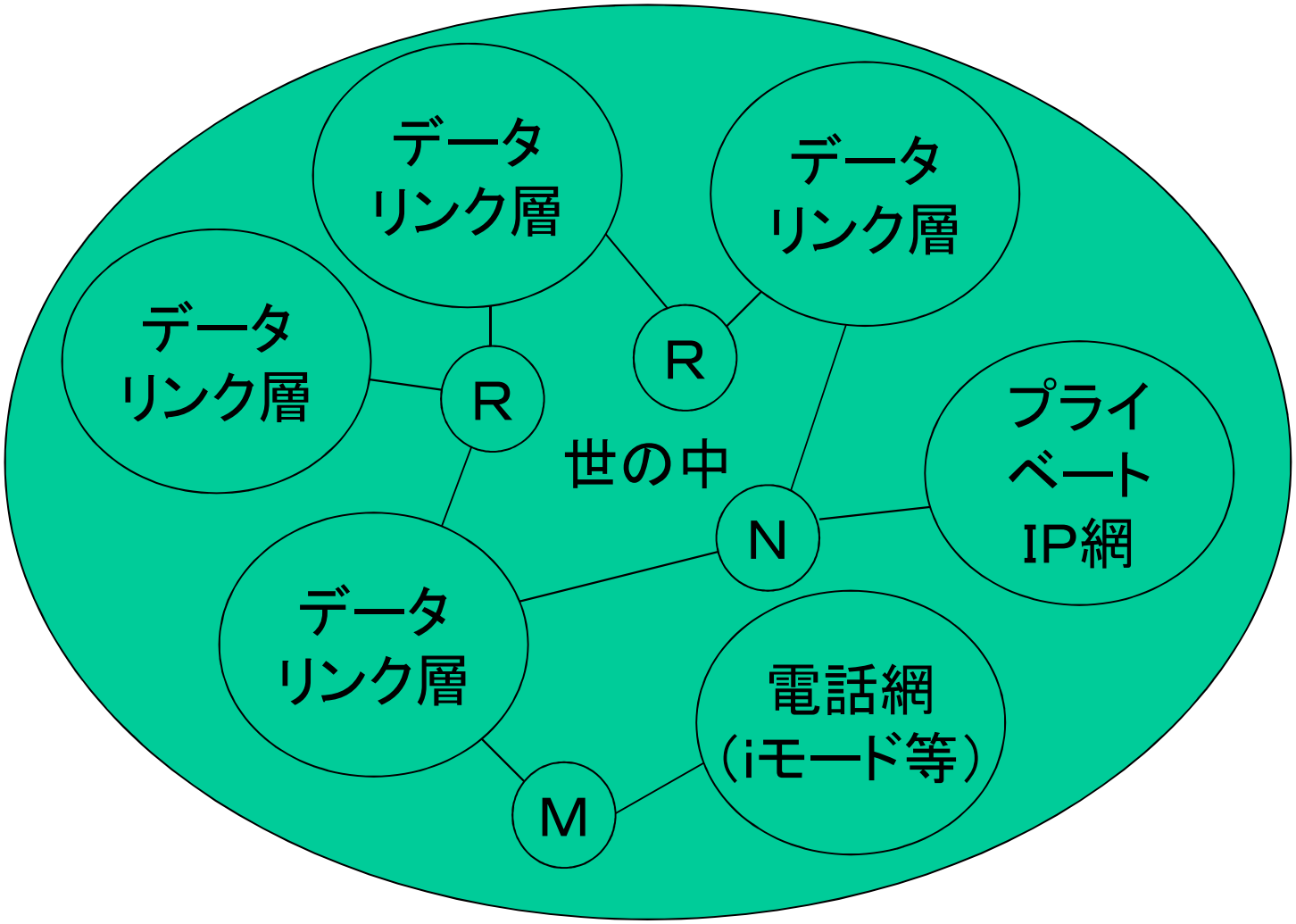
- NAT装置では、特定のポート番号しか通さない等各種の設定が可能
  - あぶないプロトコルはとおさない
  - どうしても必要なプロトコル（SMTP、HTTP）も、特定のホスト（プロキシ）にしか通さない
    - アドレスの節約にもなる
    - 他のホストはプロキシ上のアプリケーションゲートウェーを利用
  - それ以外のプロトコルは利用不可能

インターネットセキュリティの本質

- エンドツーエンドセキュリティ
  - インターネットの基本原理がエンドツーエンド原理
- すべてのエンド（アプリケーション）をセキュアーに
  - 王道はない
  - 魔法もない

NATを受け入れるということは

- NATはIP網間の装置
  - IPなら、まあよかろう？
- NAT対応プロトコルは、IPヘッダ以外にIPアドレスを含まない
  - NAT対応プロトコルは、IP網以外でも動作
  - 網がIPである意味はない
    - iMODE等
- IPは不要

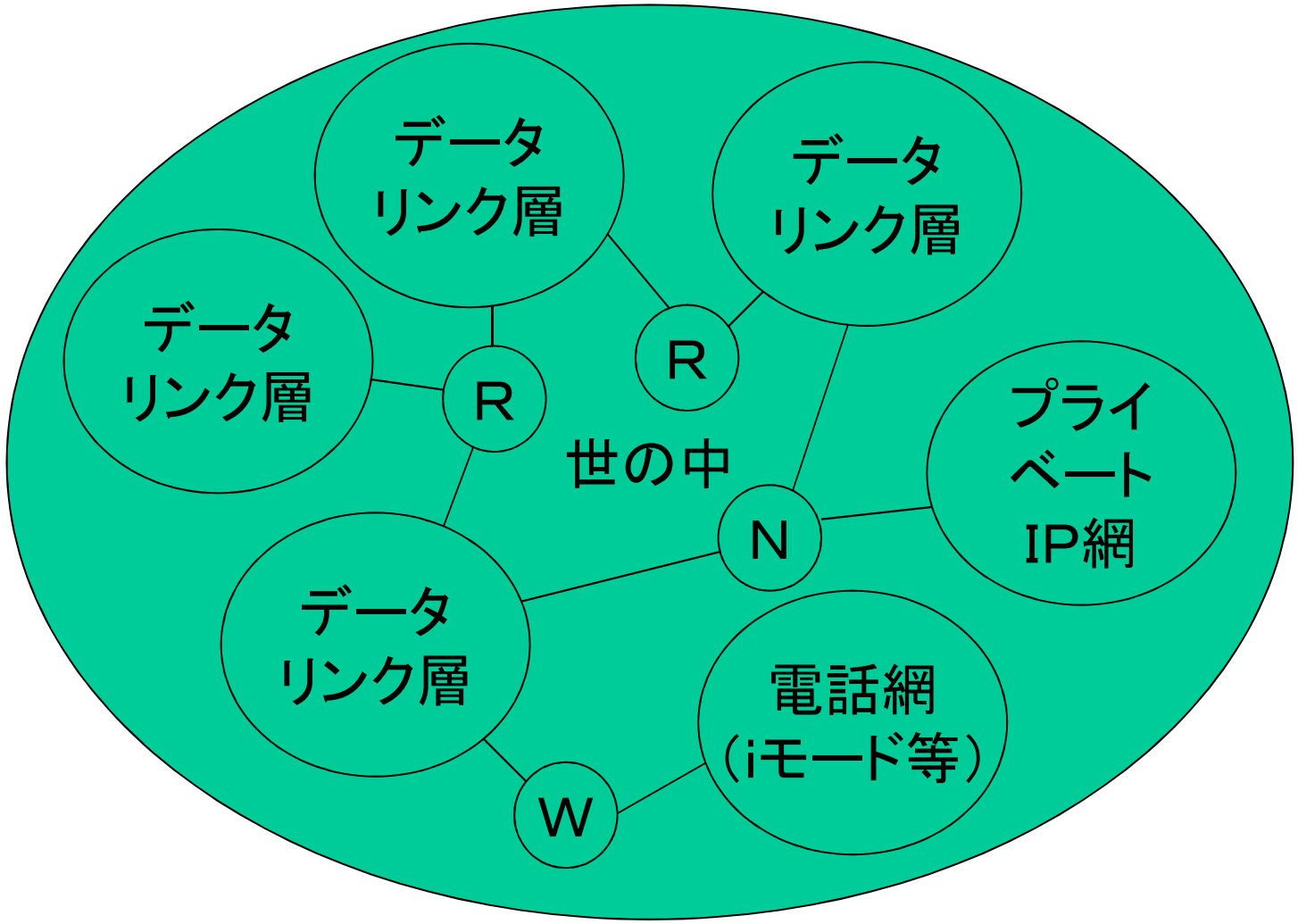


R　：ルータ　　N　：NAT装置　　M　：メイルゲートウェイ

現在の電子メイル環境

インターネットと網の構造

- インターネットの例
  - ダイア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
  - NAT



R　：ルータ　　N　：NAT装置　　W　：ウェブゲートウェイ

現在のウェブ環境

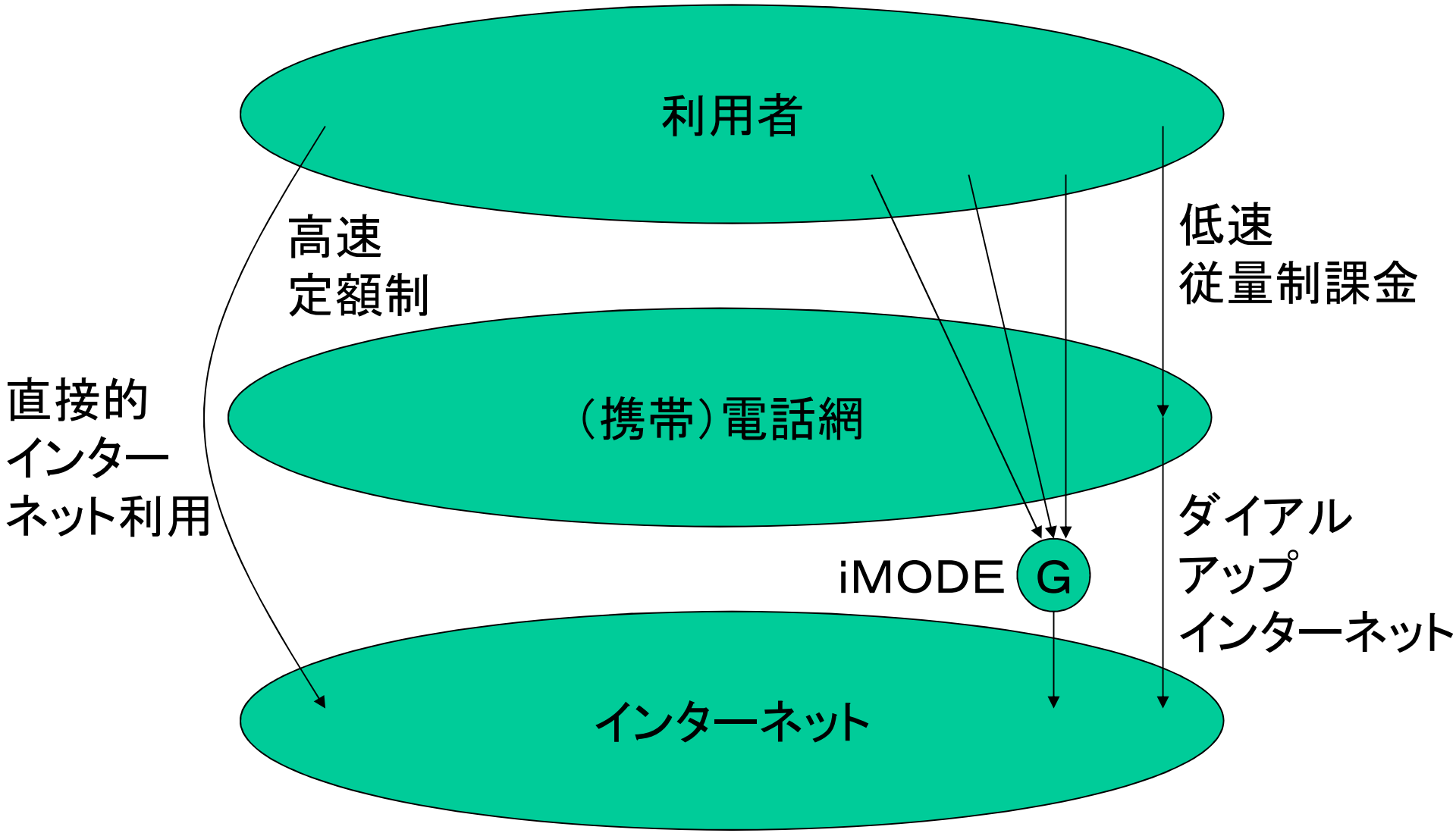


図　電話網とインターネット

インターネットと
新しいアプリケーションの作り方

- プロトコルを設計する
- ホスト上に実装する

アプリケーションゲートウェイと新しいアプリケーションの作り方

- プロトコルを設計する
- ホスト上に実装する
- アプリケーションゲートウェイ上に実装する
  - 網事業者側の対応が必要
  - アプリケーションごとの個別対応が必要

アプリケーションゲートウェイとアプリケーションの発展

- かっては電子メイルが主流
  - インターネットとは電子メイルのことである
- 現在はウェブが主流
  - インターネットとはウェブのことである
- アプリケーションゲートウェイはSMTPやHTTPやDNSには対応
  - 新たなアプリケーションはHTTP上に作成？
  - あぶないアプリケーションも動作可能

インターネットの崩壊

- Renumberingは受け入れられない
  - DHCP／PPPによるアドレス割り当ての発達
    - 常時接続性の喪失
      - 少数の常時接続サーバによるプロキシー
      - ダイアルアップクライアント
        - » End to End原理の崩壊
- NATの発達
  - Global Connectivity原理の崩壊

悪循環（現実）

- NATがはびこるとIPv4アドレス空間はぎりぎり枯渇しない
  - IPv4アドレス空間がぎりぎり枯渇しないならIPv6は普及しない
    - IPv6が普及しないので、ぎりぎりのIPv4のアドレス空間割り当ては慎重になる
      - IPv4アドレスが割り当てられないと、NATがはびこる
- NATはインターネットを崩壊させる

IP over HTTP
IP over DNS

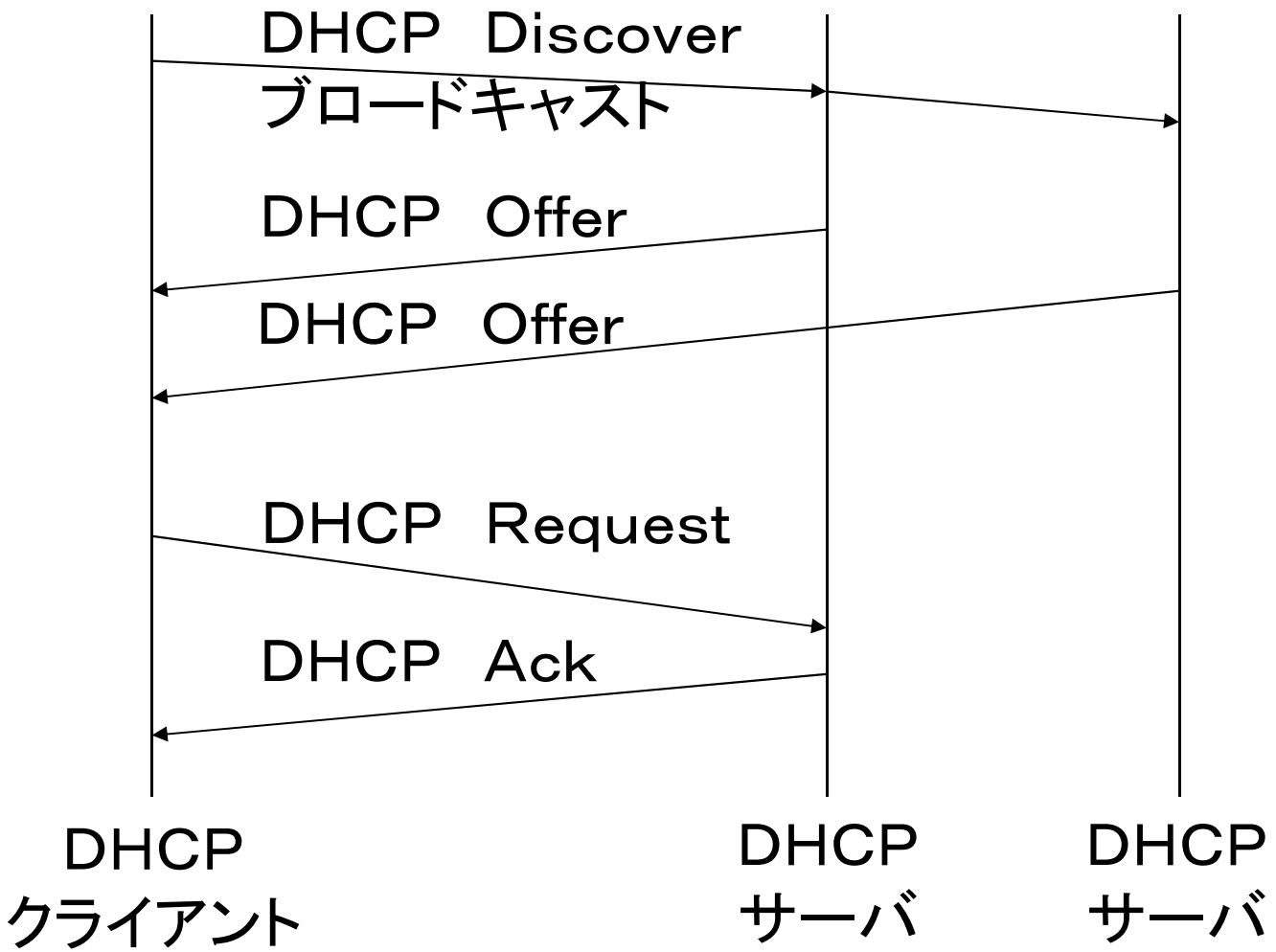
- プライベートインターネット内部で動作
- インターネット側の協力ホストとHTTPやDNSで通信
  - HTTPやDNSにIPパケットを載せる
- 最近話題のSOFTETHERはEthernet over HTTP等

DHCP （Dynamic Host Configuration Protocol、 RFC2131）

- DHCPサーバが、ホスト（DHCPクライアント）のIPアドレス等の設定情報を供給
- UDPによるリンクブロードキャストを利用して管理の手間を軽減
- サーバ側の管理は必須
- セキュアと仮定できるリンクでのみ有効
  - 暗号を利用するとクライアントにあらかじめ鍵を設定することは必須

DHCP （2）

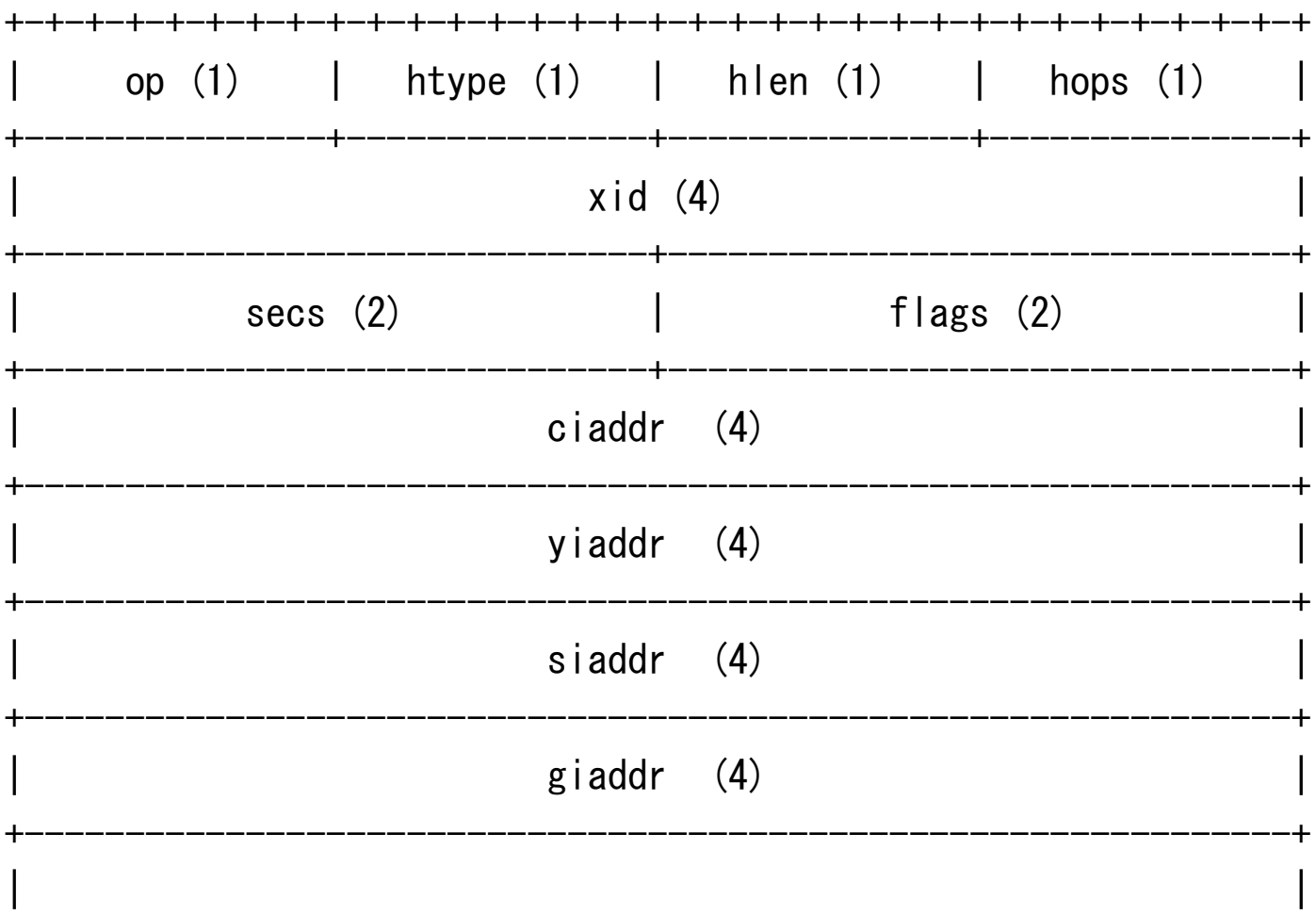
- 前身はBOOTP
  - ディスクレスサーバ（ローカルには状態なし）の立ち上げ
- 現在の主目的はアドレスの動的割り当て


DHCP Discover
ブロードキャスト
DHCP Offer
DHCP Offer
DHCP Request
DHCP Ack
DHCP
クライアント
DHCP
サーバ
DHCP
サーバ

DHCPのパケット形式（１）

- アドレスがわからない場合
  - 受信者アドレスはブロードキャストアドレス
    - DHCP　Discoverのみ
  - 送信者アドレスは０．０．０．０
    - アドレス取得後はそれを利用
- UDP
  - サーバポート番号67
  - クライアントポート番号68

DHCPのパケット形式（２）



DHCPのフィールド（１）

- op
  - １：リクエスト、２：リプライ
- htype
  - ハードウェアタイプ（１＝10Mイーサ）
- hlen
  - MAC（ハード）アドレス長（イーサは6）
- hops
  - リレーされた場合のホップ数（最初は0）

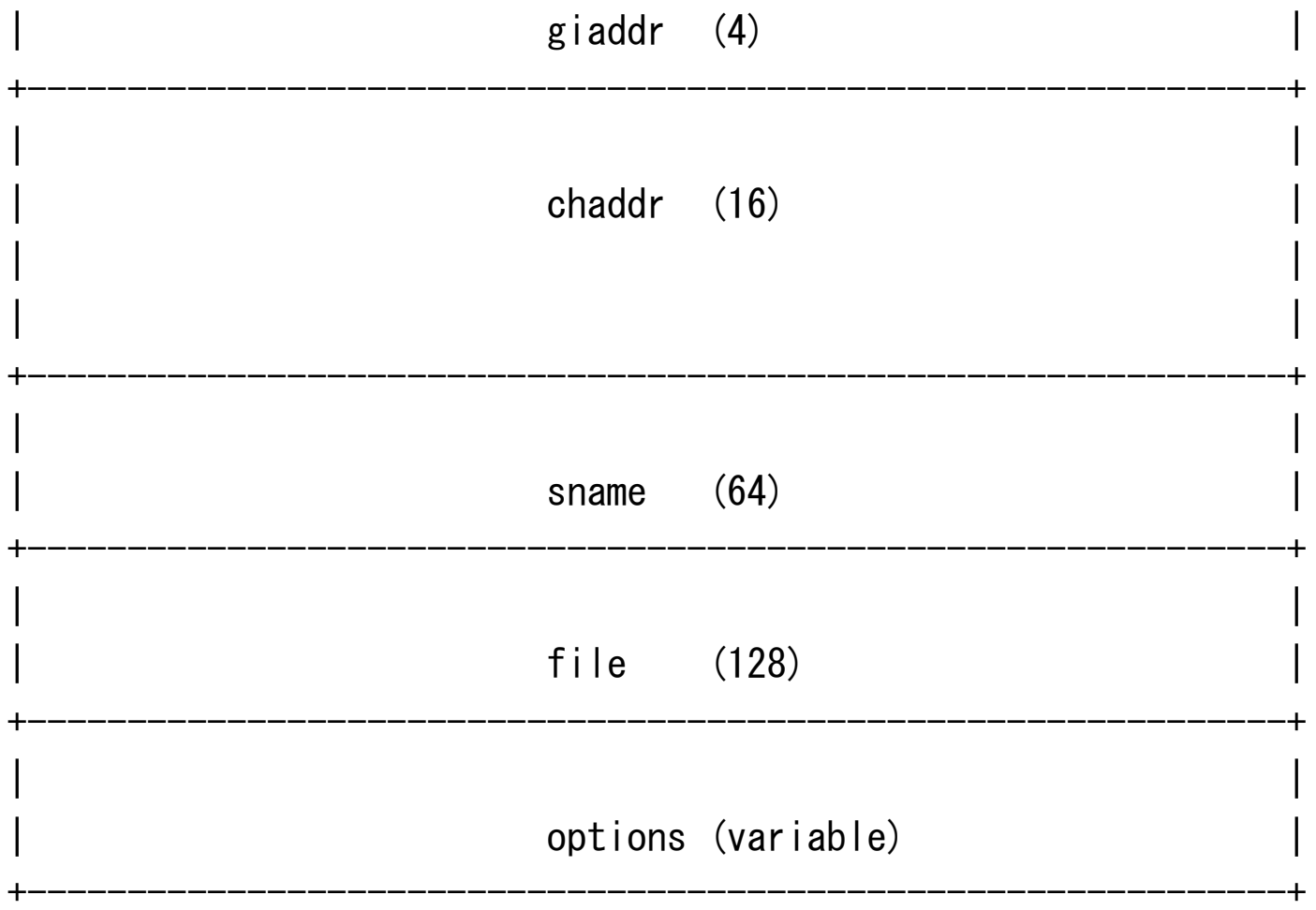
DHCPのフィールド（２）

- xid
  - トランザクションID
- secs
  - 最初に要求して以来の秒数
- flags
  - フラグ
- ciaddr
  - クライアントのIPアドレス

DHCPのフィールド（３）

- yiaddr
  - クライアントに割り当てたIPアドレス
- siaddr
  - サーバのIPアドレス
- giaddr
  - リレーのIPアドレス

DHCPのパケット形式（３）



DHCPのフィールド（４）

- chaddr
  - MAC（ハードウェア）アドレス
- sname
  - サーバーホスト名（文字列）
- file
  - 設定ファイル名（文字列）
- options
  - オプション列

DHCPとDNS（１）

- インターネットのホストはDNSにより
  - ホスト名ー＞IPアドレスを変換（正引き）
  - IPアドレスー＞ホスト名を変換（逆引き）
- DHCPで得たIPアドレスのDNS登録は？
  - DDNS（RFC2136）により可能
  - 問題はセキュリティ
    - 逆引きはDHCPサーバの責任で
    - 正引きはホストの責任で（鍵の設定は？）

DHCPとDNS（2）

- DNSサーバのアドレスをどう設定？
  - （IPv6では）NDによる割り当て
    - NDになんでもやらせすぎ
    - そもそもNDが常に利用可能とは限らない
  - DHCPによる割り当て
    - DHCPが利用可能なら適切
  - ANYCASTによる割り当て
    - どこでも利用可能
    - DHCP等による上書きも可能

まとめ

- これまでのNATは
  - エンドツーエンドモデル違反
  - 新たなアプリケーションが育たない
  - 実は、問題ないNATも可能で、IPv6も不要
    - 詳しくはおまけで
- DHCPは
  - それなりに使える
  - コンフィギュレーション不要は幻想

エンドツーエンドNAT

太田昌孝
東京工業大学情報理工学研究科
mohta@necom830.hpcl.titech.ac.jp

IPアドレスが足りない！！

- それでもIETFなら、、、IETFならきっと何とかしてくれる！？
  - いまだに、何ともなっていない、なりそうもない
    - NATによるアドレス節約
      - エンドツーエンドインターネットを破壊
      - いろんなプログラムが、まともに動作しない
    - IPv6によるアドレス拡張
      - 実運用を考えない政治的妥協の産物
      - オプションヘッダ、PMTUD、Stateless AC、Link Local Address等有害無益な機能を盛込み過ぎ

SALTZER等の原論文でのエンドツーエンド論法

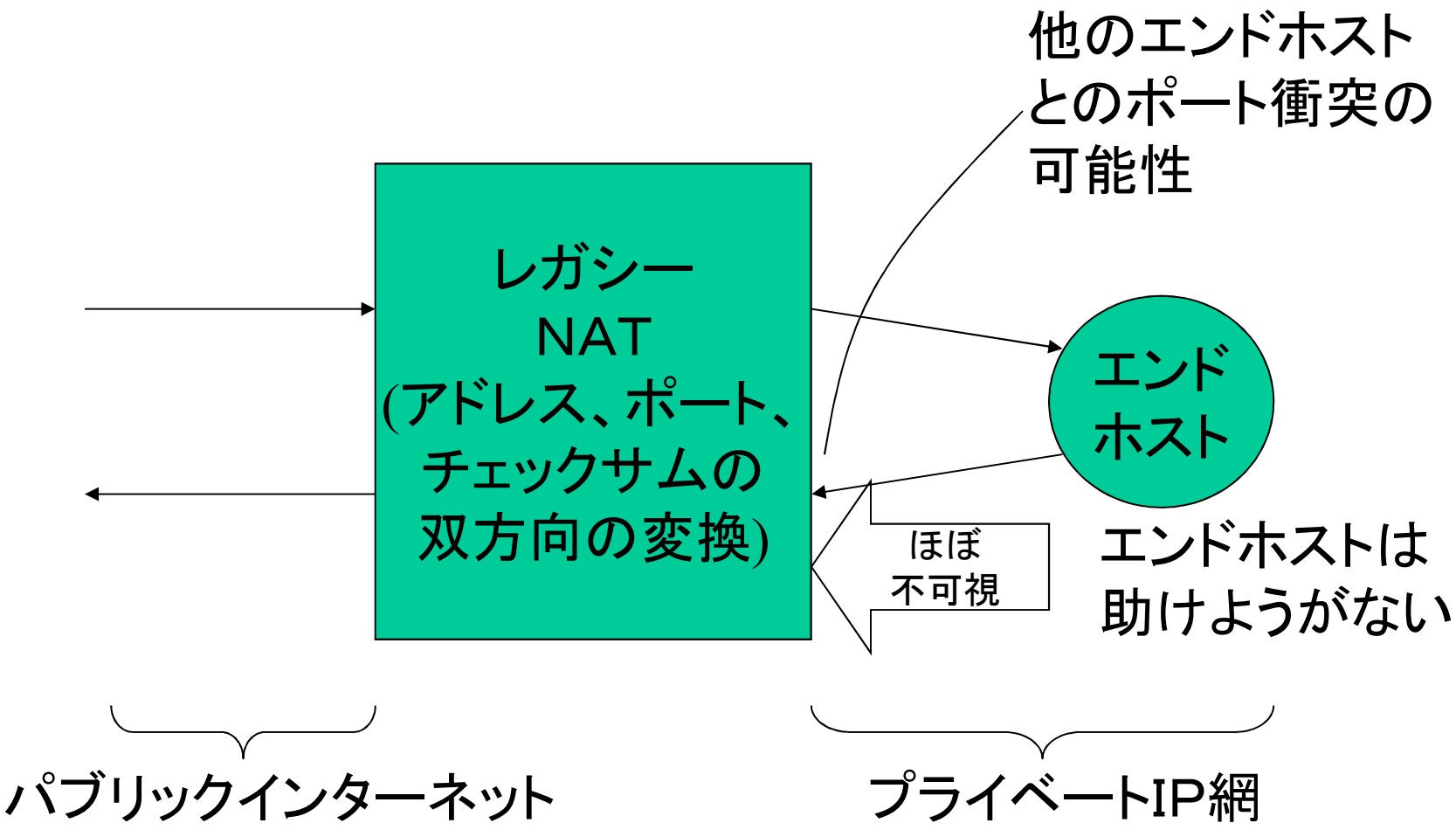
http://groups.csail.mit.edu/ana/Publications/PubPDFs/End-to-End%20Arguments%20in%20System%20Design.pdf

- 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.)

背景

- エンドツーエンド論法によると、
  - NAT can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system
    - エンドホストの知識と助けを利用していない今のNATは、不完全で不正確でエンドツーエンド透過性をもたない
- ならば、逆は成り立つのか？
  - With the knowledge and help of the application standing at the end points of the communication system
    - Can NAT be implemented completely and correctly?

レガシーNAT


他のエンドホスト
とのポート衝突の
可能性
レガシー
NAT
(アドレス、ポート、
チェックサムの
双方向の変換)
エンド
ホスト
ほぼ
不可視
エンドホストは
助けようがない
パブリックインターネット
プライベートIP網

エンドツーエンドNAT

ー　NATの存在を端に積極的に見せる　ー

- プライベート網内の端末にNATGWの知る
  - 各端末で共有するパブリックアドレス
  - 各端末に割り当てたポートの範囲
  - NATGWとの通信方法（アドレス、ポート）

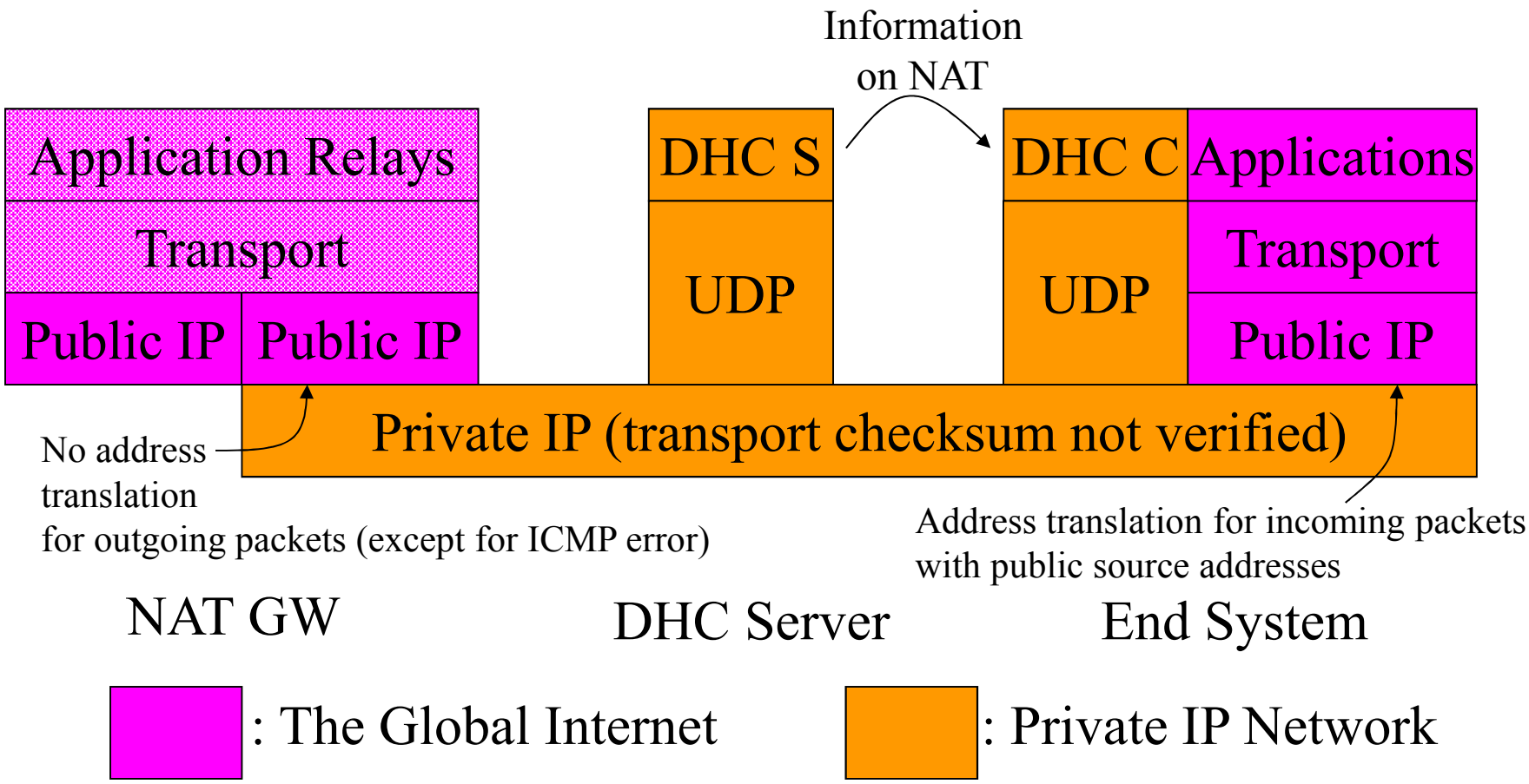
等の情報を、DHCPやPPP等で通知

- 各端末は
  - 自らの知識により、NAT動作が完全かつ正確なものになるように、補完する

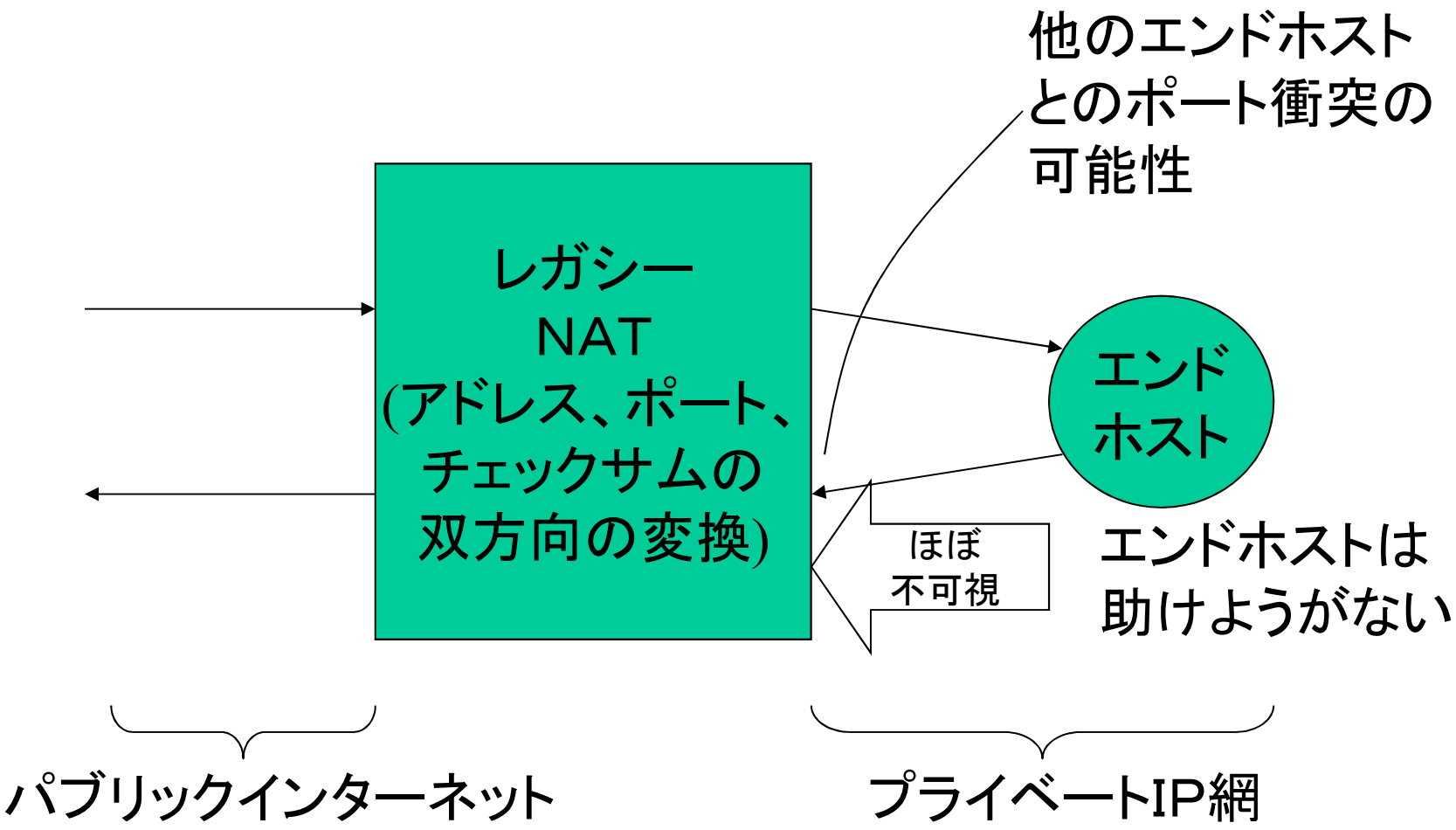
エンドツーエンドNATの動作

- NATゲートウェイ
  - 受信者ポート番号により、パケットの受信者アドレスをパブリックアドレスから変換
    - ポートやトランスポートチェックサムは変換せず
- NAT背後のエンドホスト
  - 受信者アドレスをパブリックアドレスに逆変換
    - トランスポートチェックサムは正しくなる
  - 送信者ポート番号を、エンドホストに割り当てられたものに制限

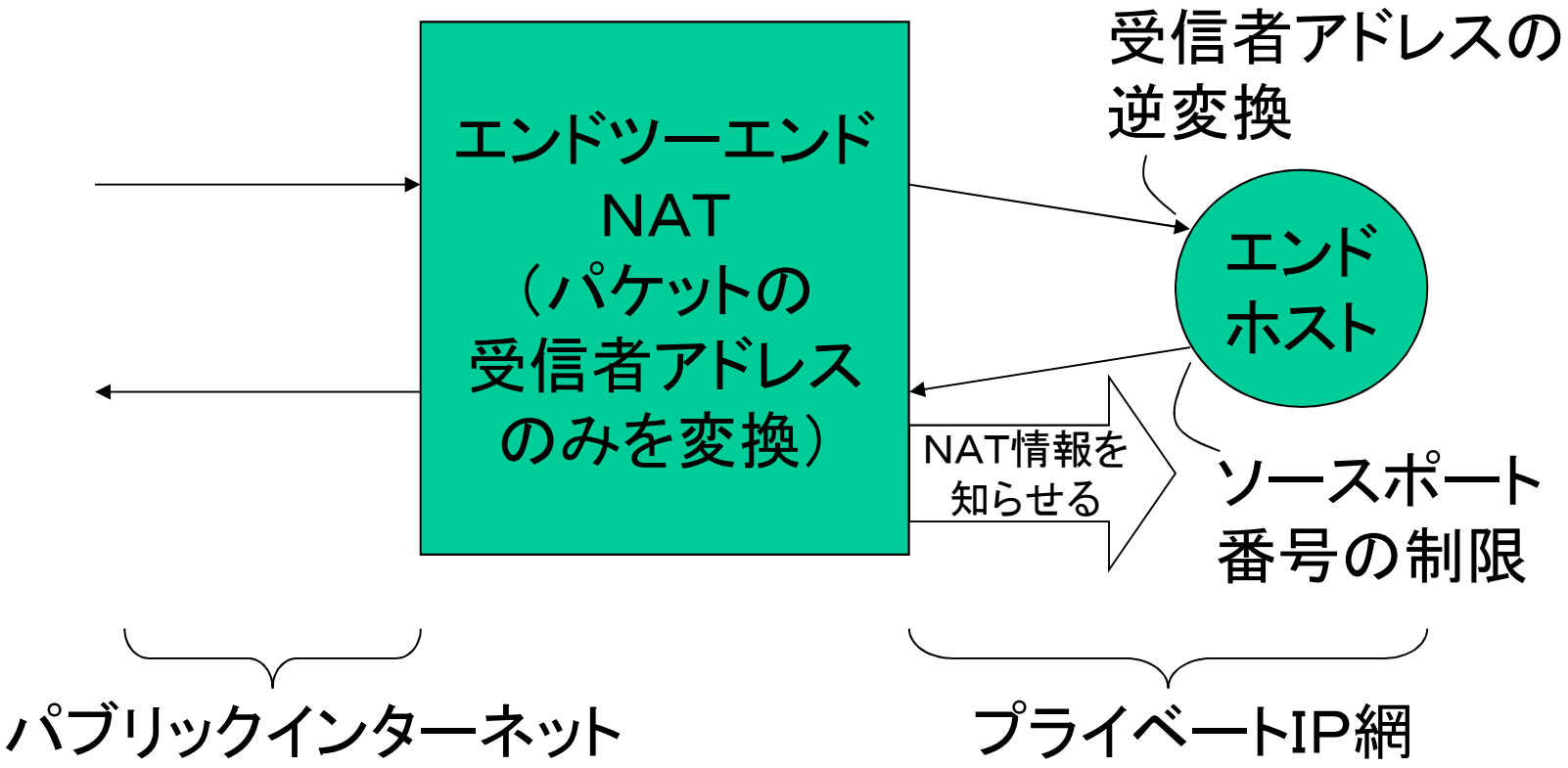
エンドツーエンドNATの
レイヤー構造



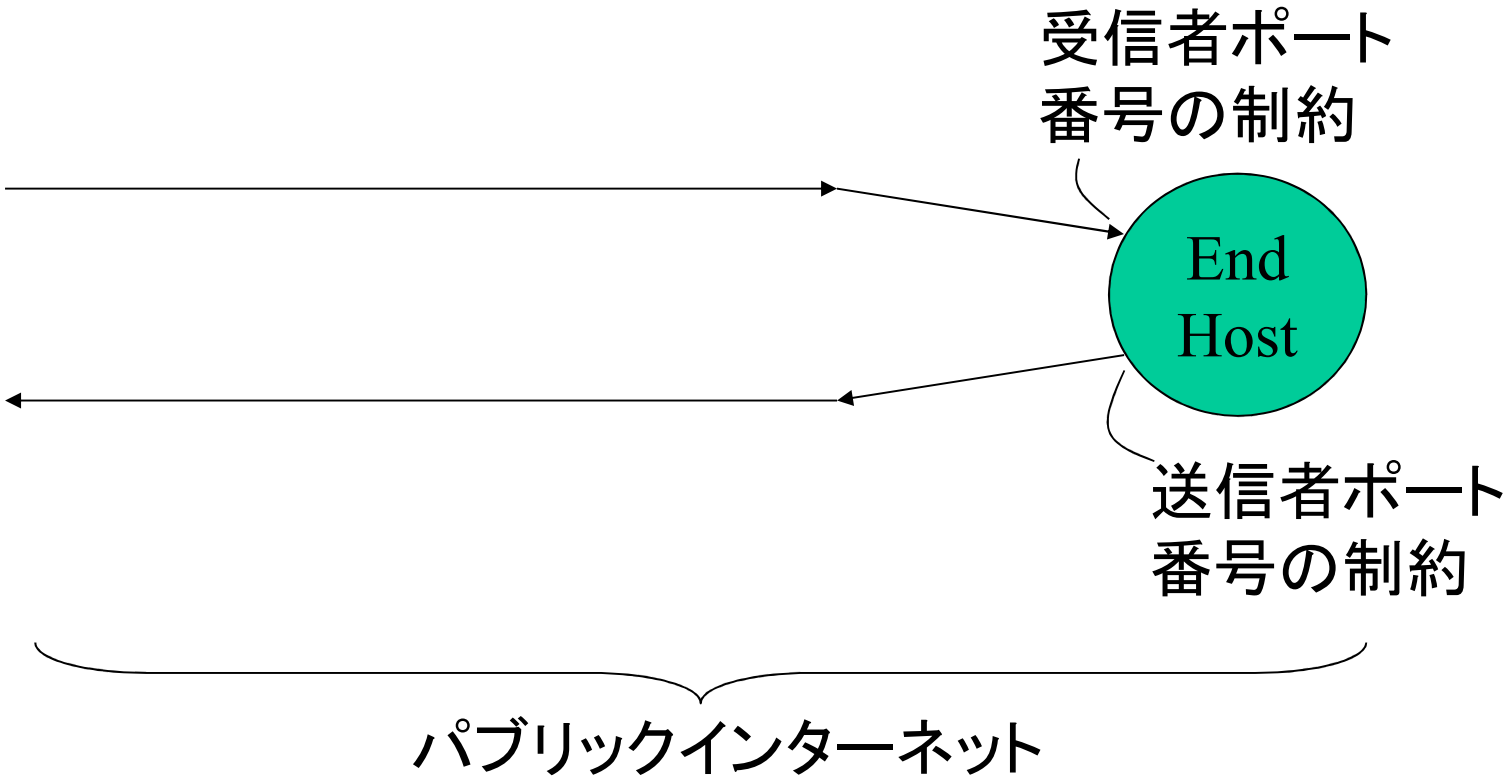
レガシーNAT


他のエンドホスト
とのポート衝突の
可能性
レガシー
NAT
(アドレス、ポート、
チェックサムの
双方向の変換)
エンド
ホスト
ほぼ
不可視
エンドホストは
助けようがない
パブリックインターネット
プライベートIP網

エンドツーエンドNAT



エンドツーエンドNAT背後のエンドホストはインターネット直結と等価



エンドツーエンドNATの性質

- 完全なエンドツーエンド透過性
  - ポート番号か同等物さえ存在すれば
  - ftpのポートコマンドも自然にNATを超える
- 多段ネスティング可能
- コンパチビリティも豊富
  - レガシーNAT、ICMP、(ポート番号も含め)DNS逆引き、マルチキャスト、モビリティ(ポート単位の移動のため要拡張)、IPSEC、、、

スタティックNATと
ダイナミックNAT

- スタティックNAT
  - 各端末に固定したポート範囲を割り当て
    - ポート数が多ければ（数百？）、これで十分
      - 端末は自己に割り当てられたポートのみをソースポートとして使う。返事が貰えないので、偽装は無理。
- ダイナミックNAT
  - 各端末は、ポート番号を随時NATGWに要求
    - NATGWのポート割当状態は、端末主導で更新
      - タイムアウト、複数GW間整合性等の問題は解消

固定ポートでのE2ENATとポートフォワーディングとの違い

- ポートフォワーディングでは
  - 一部のポートを特定の端末に固定割当
    - 旧来のNATが、トランスポート層で中継
    - スタティックNAT同様、端末はサーバ動作可能？
      - 実は透過性は無く、クライアントすらまともに動かない
- E2ENATでは
  - 端末の助けにより、完全なE2E透過性を実現

エンドツーエンドNATと ポート番号

- URLやDNS　SRVで、デフォールト以外のポート番号を指定可
- E2ENATはほぼIP層だけで動作するが
  - 受信者ポート番号はIPヘッダの外にある
    - 純トランスポートプロトコルでは、IPヘッダ直後16ビットが送信者ポート、次16ビットが受信者ポート
    - ICMPの仕様から、ICMP以外は、送信者ポート番号は、IPヘッダ直後の8バイトに含まれるはず
      - ポート番号は、IPヘッダ直後8バイト内の2バイト境界間の16ビットと決め打ちしてもよさそう（IPSECも対応可）

SRV RR（RFC2782）

- アプリ全般のMXのようなもの
  - Name TTL Class MX
    Priority Target
  - _Service._Proto.Name TTL Class SRV
    Priority Weight Port Target
    - Weightで重み、Portでポート番号が指定可
  - _http._tcp.www.example.com SRV
    0 1 9 server.example.com.

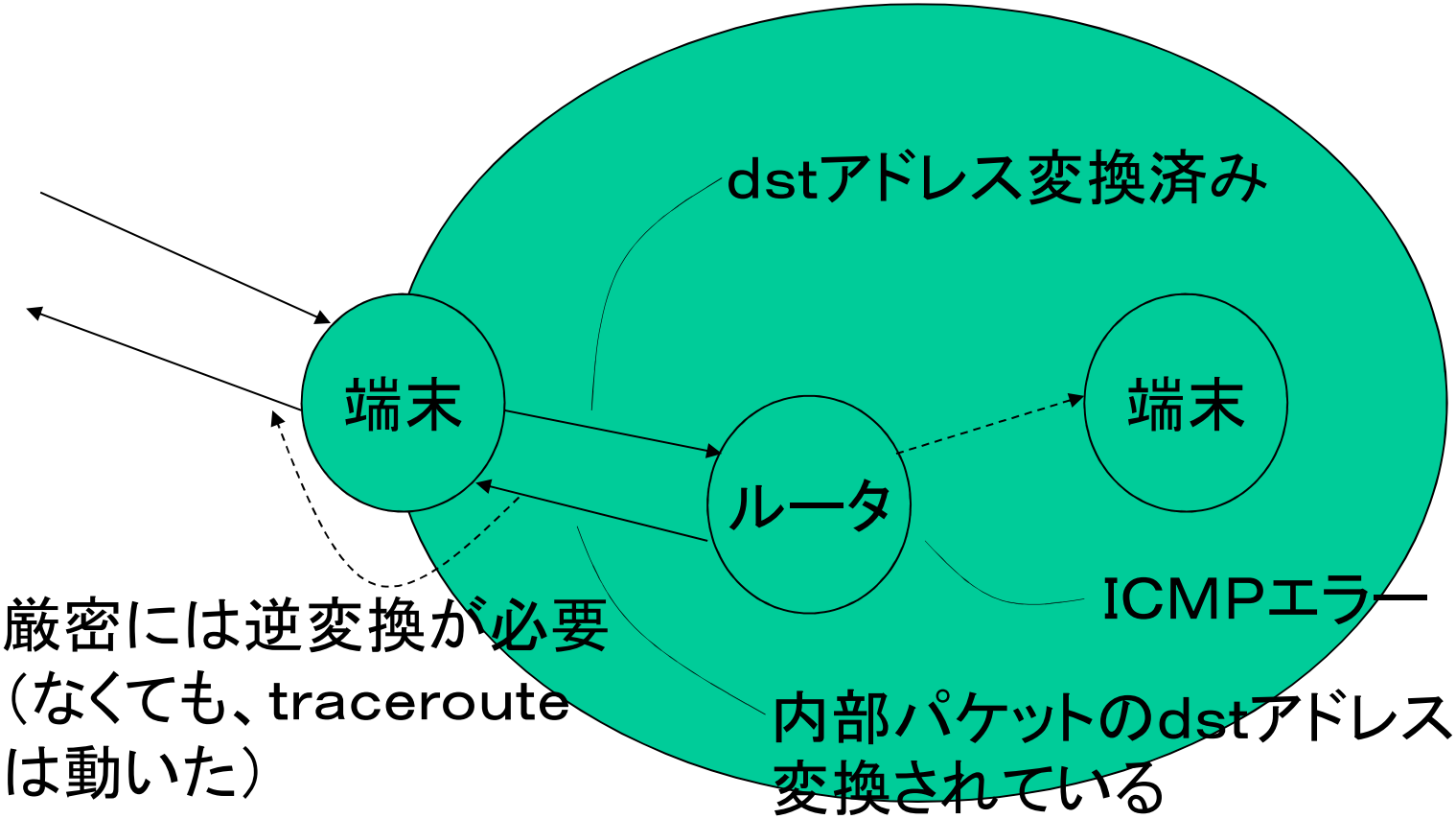
エンドツーエンドNATとICMP

- 基本的にそのまま動作
  - ICMP内部のアドレス変換は、厳密には必要
    - プライベート網からパブリック網へのICMPエラー
- ICMPパケット自体のポート番号は？
  - ICMPエラー
    - 内部パケットのsrcがdst、dstがsrc
  - ICMP　ECHO等
    - IDをsrc、seqnoをdstとみなす

エンドツーエンドNATと ICMPエラー

- ICMPエラーは、エラーを起こした内部パケットの送信者ポートによりアドレス変換
- ICMP　HOST　UNREACHは
  - パブリックアドレスを共有する他ホストに影響
  - ソフトエラーなので、気にしない
    - TCPは接続を切らない
    - 気を利かせたつもりでPORT　UNREACHに変換すると、ハードエラーなので悲惨

ICMP内部のアドレス変換



エンドツーエンドNATとping／traceroute

- pingやtracerouteを動かしたい
  - 自分に割り当てられたポート番号をソースポート（ID）に利用するよう拡張
  - ポート番号を指定できるよう拡張
    - ping ... host[:dstport[,incr[,count]]]
    - traceroute ... [-p port[.incr]] ... host ...
    - 各ホストに、port、port+incr、port+incr*2、、、のポートが割り当てられていると仮定

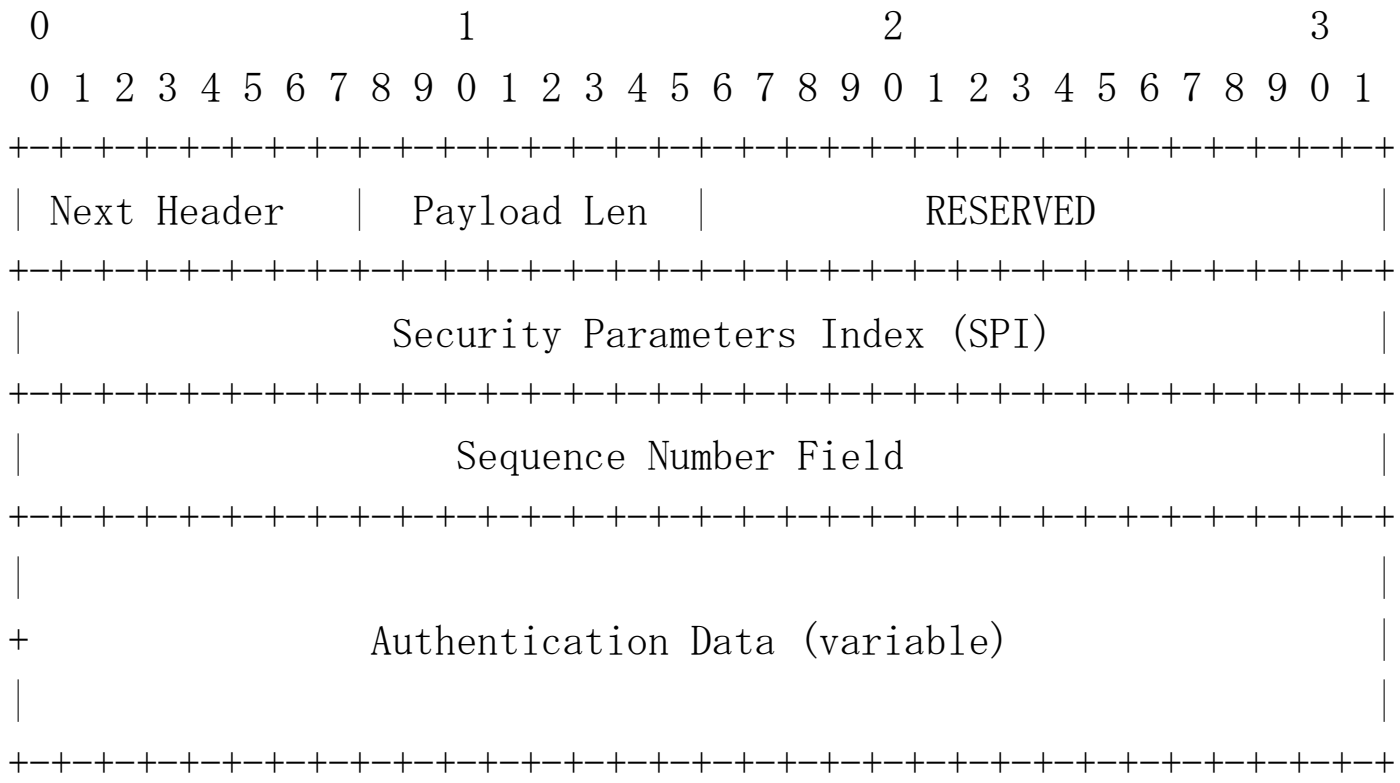
エンドツーエンドNATとICMPエコー

- ICMPエコーリクエストは
  - IDとSEQ＃をそれぞれ送信者と受信者のポート番号と看做し
    - IDを制約、SEQ＃でアドレス変換
- ICMPエコーリプライは、逆
  - IDとSEQ＃は、ICMPエコーリクエストからコピーされる
  - IDを受信者ポート番号と看做しアドレス変換すれば、エコーリクエストの送信者に届く

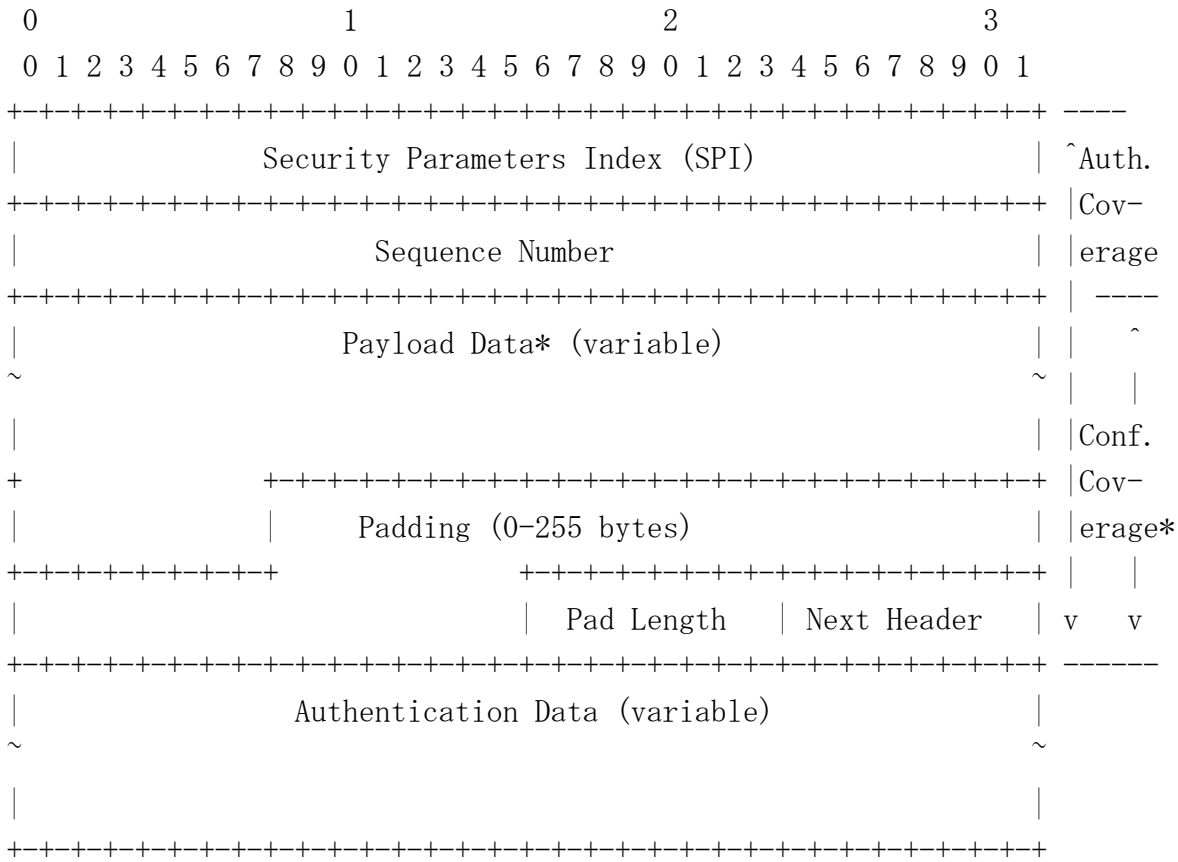
エンドツーエンドNATと IPSEC

- AHもESPも32ビットSPIをIPヘッダ直後の8バイト以内に持つ
- SPIの値を決める際、前半16ビットを送信者が指定し、後半16ビットを受信者が指定することとすれば
  - 送信者ポート番号、受信者ポート番号として利用可能

AH（Authentication Header）（RFC2402）



ESP（Encapsulating Security Payload）（RFC2406）



エンドツーエンドNATと アプリケーションリレー

- DNS、SMTP、HTTP等は、NATGWのデフォールトポートでリクエストを受け、リクエスト中の情報（ドメイン名等）で、リクエストを端末に振り分け可
  - HTTP：のURLでのポート指定は不要に
  - DNSやSMTPのポートはNSとMXに内包され（URLで指定不可）、プライベート網内にサーバを置くには（置けなくても、ほとんど困らないが）、アプリケーションリレーは必須

エンドツーエンドNATと エンドホストのアドレス使い分け

- エンドホストは、自分のパブリックアドレスとプライベート網のアドレスを知っている
  - プライベート網へのパケットのソースアドレスはプライベートアドレスで
  - それ以外のアドレスへのパケットのソースアドレスはパブリックアドレスで
  - 自分のパブリックアドレスへのパケットは、自分のポート番号じゃなければ出力する

エンドツーエンドNATと非対応端末

- NATGW背後のE2ENAT非対応端末は
  - DHCP等によるアドレス割り当ては受けても
    - NAT情報は理解できない
- 非対応端末が出したパケットに対して
  - NATGWは旧来のNATとして対応してもよい
    - UPnP機能等もあってもよい
- E2ENAT対応端末との区別は
  - ソースアドレスで判別可

エンドツーエンドNATの
ネスト

- E2ENATGWはネスト可能
- ISPからスタティックNATで多数（数百？）のポート番号を割り当てられた顧客は
  - 一部をサーバで固定的に利用
  - 一部はダイナミックNATGWの外側に割当
    - ダイナミックNAT背後にネストしたプライベートネットワーク内の多数の端末で、ポート番号をダイナミックに共有

エンドツーエンドNATと逆引き

- 共有アドレスは普通に逆引き可能

  www.example.com A 208.77.188.166

  166.188.77.208.in-addr.arpa PTR www.example.com

- ポート別の逆引きは以下のように可能

  p1.example.org CNAME www.example.com

  1.0. 166.188.77.208.in-addr.arpa PTR p1.example.org

  p2.example.org CNAME www.example.com

  2.0. 166.188.77.208.in-addr.arpa PTR p2.example.org

  – PTRがCNAMEを指すことは、PTRから先の自動参照の懸念（RFC1034）はないので、問題ではない

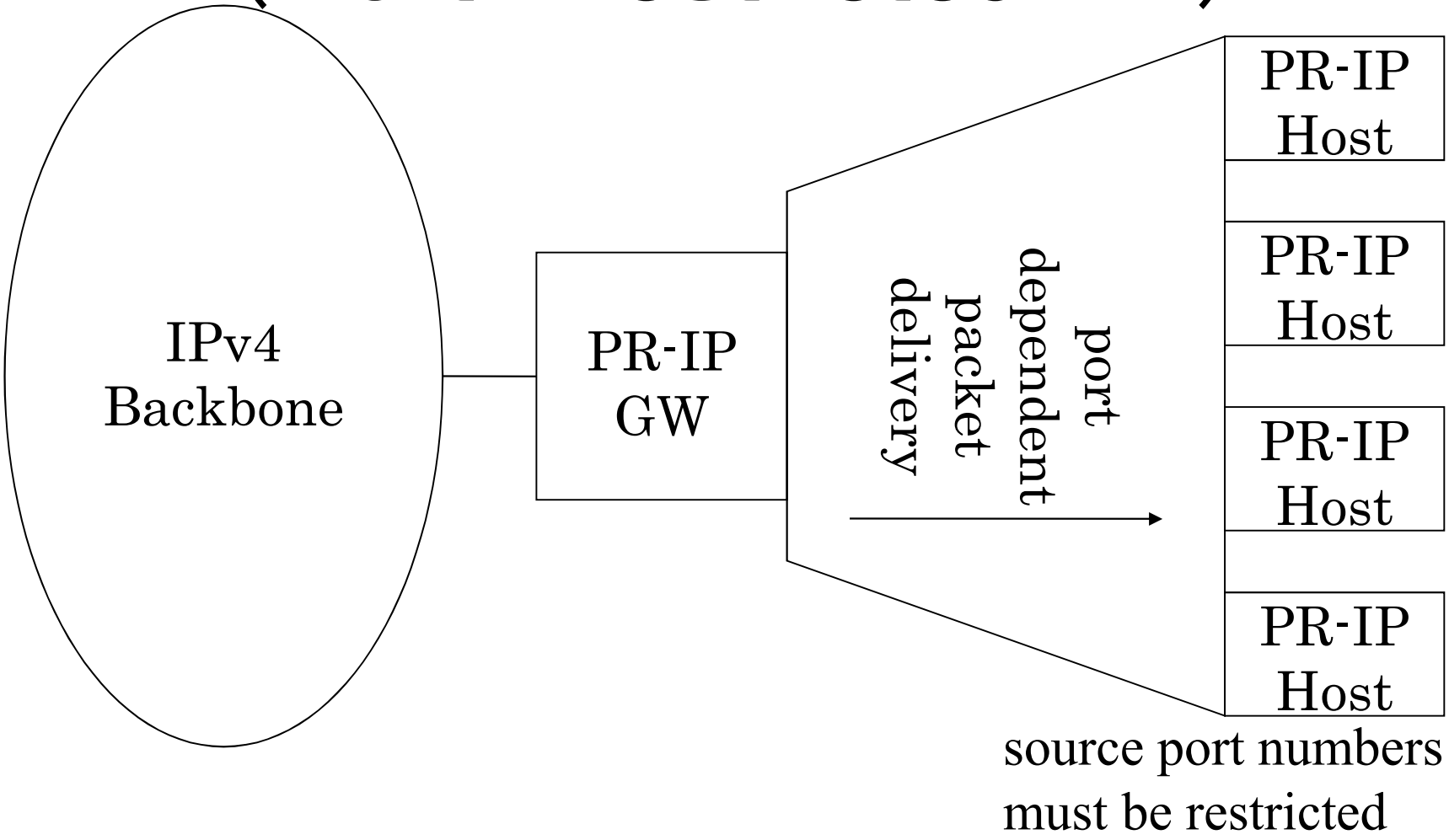
エンドツーエンドNATと マルチキャスト

- マルチキャストアドレスは、内外で共通
- プライベート網内の端末が送信する場合
  - マルチキャスト経路制御にはソースアドレスへの経路が影響するので
    - マルチキャストパケットはNATGWが出すべき
    - 端末は、送信すべきパケットを、IP　over　IPで、NATGWへ転送
      - PIMの仕組みでも使えばよい

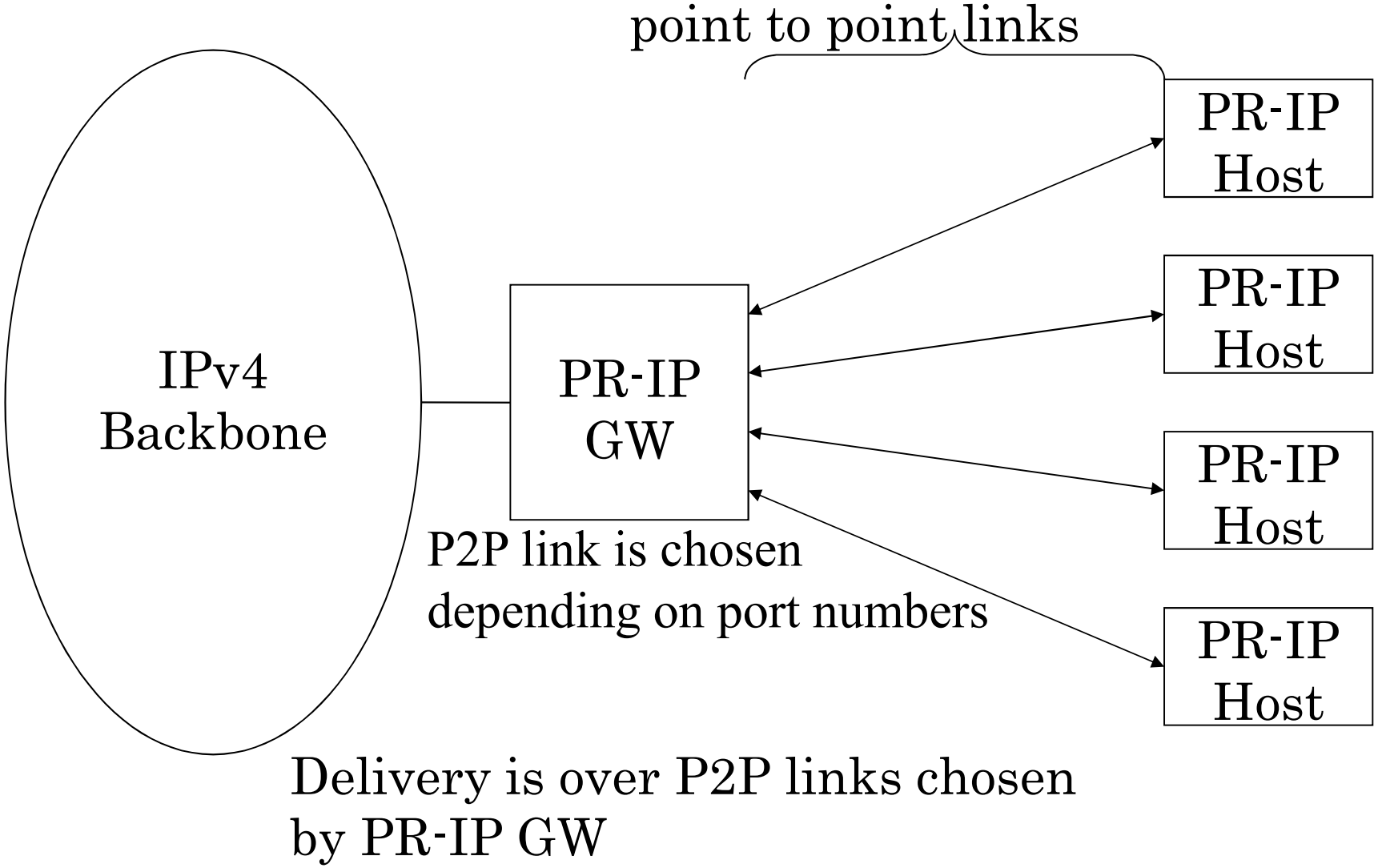
エンドツーエンドNATとモビリティ

- ホームアドレスがNAT背後にいる場合
  - NAT情報をMHに設定（静的設定で十分）
  - ホームNATGWとの通信は、HAが中継
- MHがNAT背後にいる場合
  - フォーリンアドレスとホームアドレスで、使えるポートは一般には一致しない
  - HA－＞MHのトンネルをIP over UDP over IPにすれば、解決
    - フォーリンポートは一個で十分（アドレス節約）

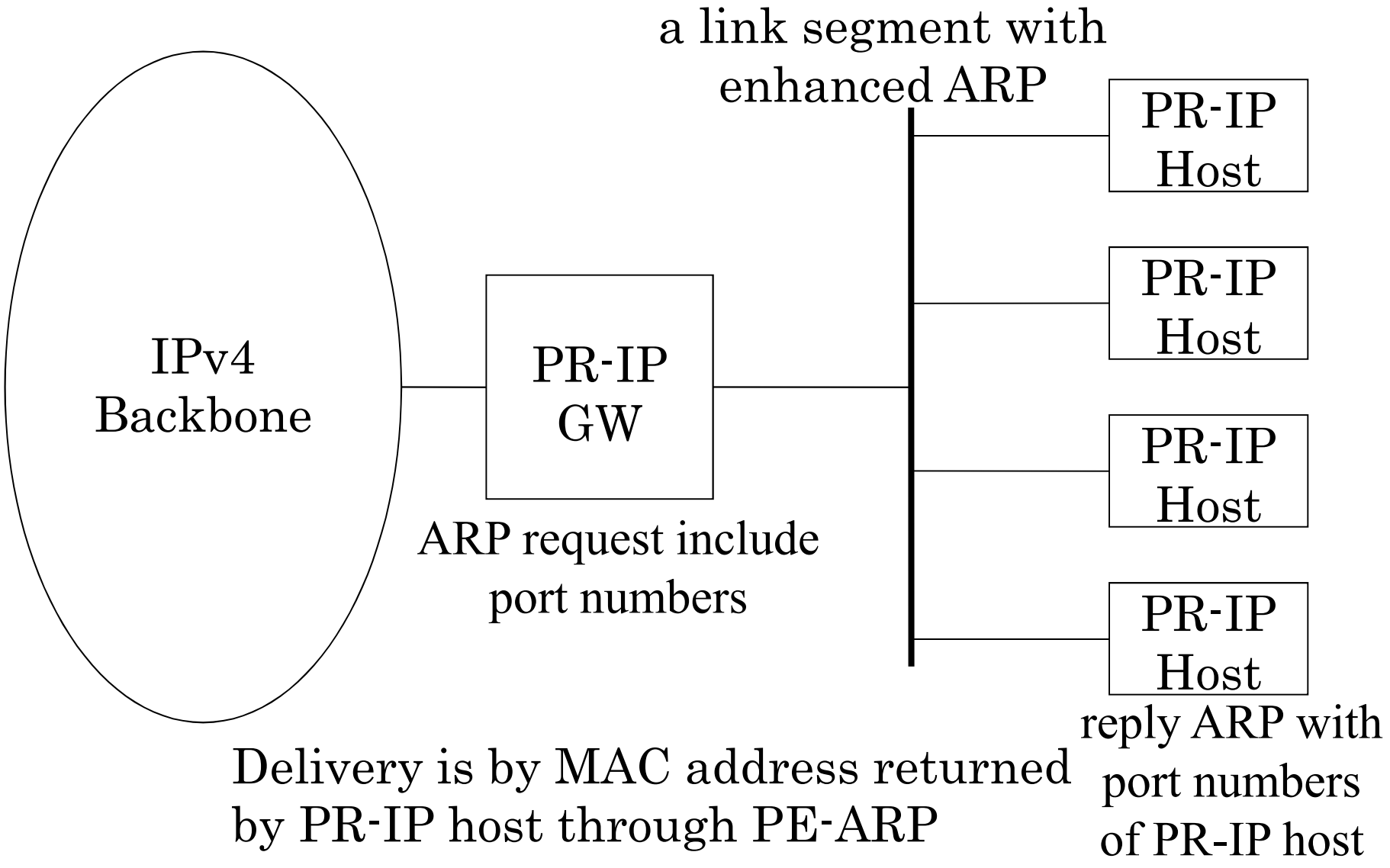
E2ENATとPR－IP
（Port Restricted IP）



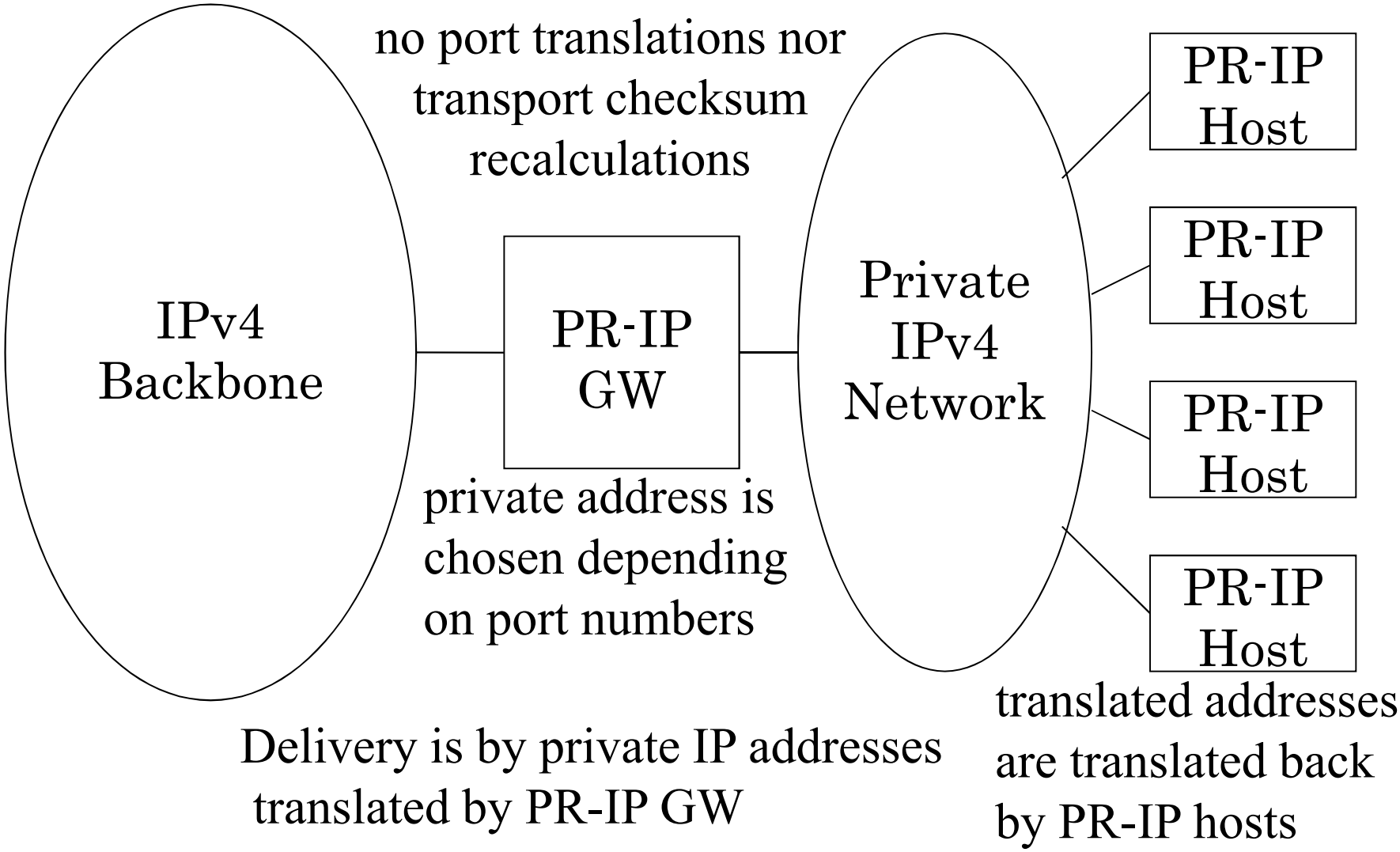
PR-IP with A+P (Address+Port)



PR-IP with Port Enhanced ARP



PR-IP with End to End NAT

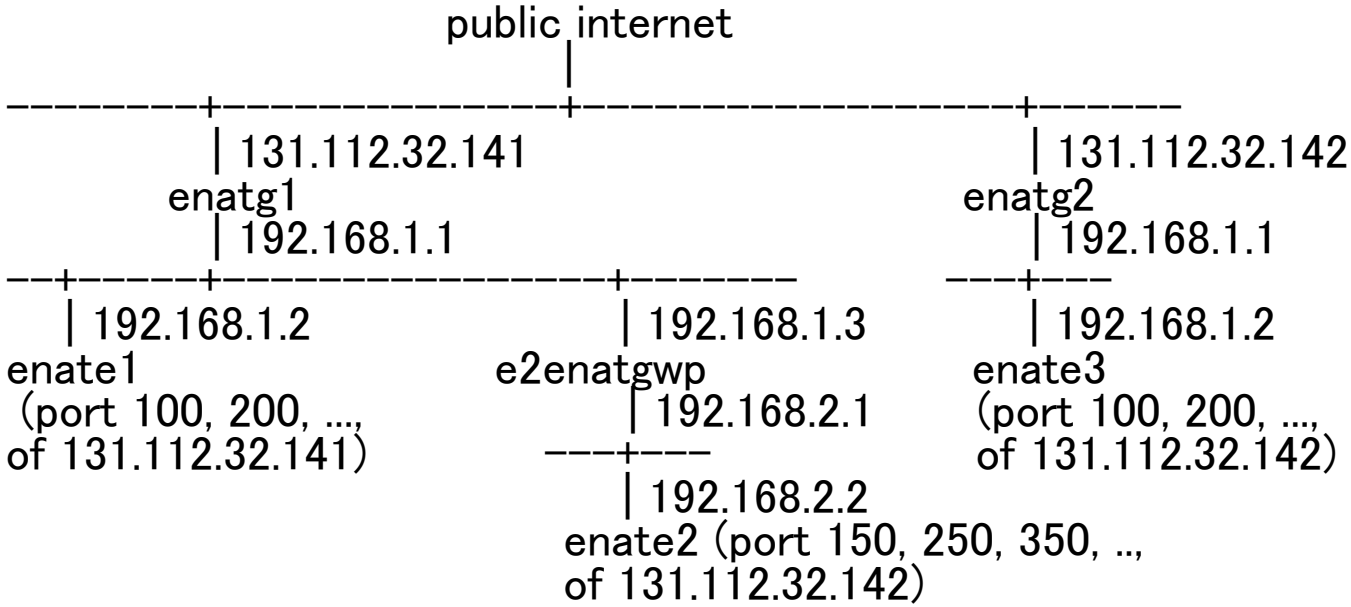


実装

- NetBSD5．1ベース、静的のみ
- 本質的改造は、
  - アドレス変換と逆変換のため
    - 端末のip__input．cへの数行の追加
    - GWのip__input．cの数十行の追加
  - ソースアドレスとソースポートの制限のため
    - 端末とGWのin__pcb．c等の数百行の追加
    - 端末とGWのip__output．cの数行の追加
      - NICにトランスポートチェックサム計算をやらせない

エンドツーエンドNATの
デモ環境

- sshでログイン可能（guest、guest）
  - e1、e2、e3のポート番号は100、150、100



エンドツーエンドNATと フラグメンテーション

- IPフラグメンテーションでは
  - IDはパケットの寿命の間は、dstアドレス毎にユニークなはず
    - 各エンドがばらばらに割り当てるIDは偶然衝突することもある
- 実際にはトランスポートチェックサムで救済
  - 16ビットのIDではどうせ不十分なので、気にしてもしょうがない
    - 1500B、寿命60秒で、13Mbpsで破綻

エンドツーエンドNATと アドレス分配ポリシー

- E2ENATは、現状のインターネット環境をエンドツーエンド透過性も含めほとんど全て保ちながら、アドレスを大幅に節約
- E2ENATをアドレス分配の前提にすべき
  - ISPの労力は？
    - どう考えても、IPv6とのデュアル運用より少ない
    - 特に、アドレスが暗記できるのは、非常に重要
- クラスEアドレスも利用すべき

エンドツーエンドNATと クラスEアドレス

- クラスEアドレスは、長持ちしない！？
- E2ENATによるアドレス節約前提なら
  - 移行期間の後、クラスEアドレスをユニキャストに使う意味はある
    - E2ENAT対応には端末の改造が必須なので
      - 同時にクラスE対応にすればよい
    - ISPやルータや既存の端末が対応しないと相互接続性が、、、
      - IPv6対応よりは、はるかに簡単（特にISPやルータ）

エンドツーエンドNATと プリフィックス

- 大域経路表に／24より長いのが増える？
  - 1600万あれば十分（というか多すぎ）
- どのみち、IPv6では、大域経路表プリフィックス数の抑制の試みは崩壊
- IPで時間を稼いで
  - エンドツーエンドマルチホーミングを実現

エンドツーエンドNATと エンドユーザー

- E2ENATの導入によりエンドユーザーは
  - サーバーもクライアントも今と同様に動作
  - IPv6対応は不要に
    - アドレス（とポート）は、普通の人でも暗記可能
  - httpのURLではポート指定不要
  - 既存ユーザーはそのままで、新規ユーザーは
    - （旧来のNAT環境が嫌なら）端末改造が必要
      - このまま旧来のNAT導入するより、遥かによい

TCP and UDP with Port Length Enhancement (TUPLE)

-- A Scribbled Slate Approach for Internet Addressing and Routing --

Masataka Ohta

Tokyo Institute of Technology

mohta@necom830.hpcl.titech.ac.jp

TUPLE

(TCP and UDP with Port Length Enhancement)

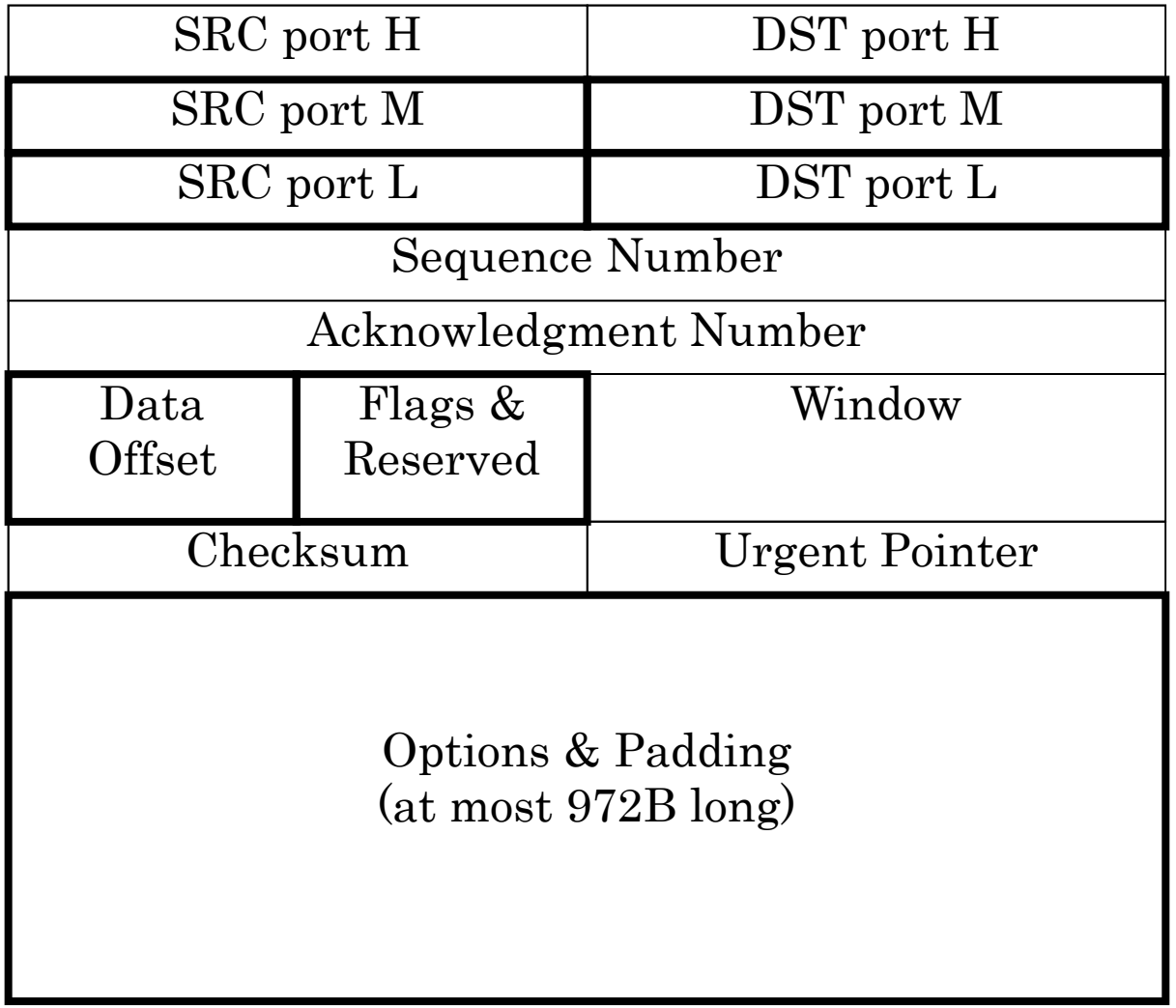
-- A Scribbled Slate Approach for Internet Addressing and Routing --

- TCP and UDP with 6B port numbers
  - and extended transport option field
    - length of “Data Offset” field of TCP is extended
    - unused UDP “Length” field is used as “Data Offset”
    - the option field may contain alternative source addresses for better aggregation
- Named after TUBA (was an IPng candidate)
  - “TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing” (RFC1347)
  - Port numbers of other protocols may also be enhanced

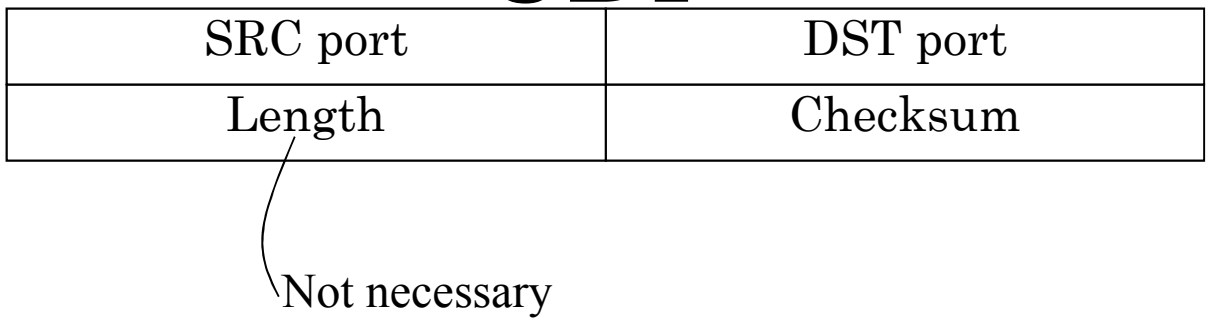
Header Format of the Current TCP

<table>
<tr><td colspan="2">SRC port</td><td>DST port</td></tr>
<tr><td colspan="3">Sequence Number</td></tr>
<tr><td colspan="3">Acknowledgment Number</td></tr>
<tr><td>Data Offset</td><td>Flags & Reserved bits</td><td>Window</td></tr>
<tr><td colspan="2">Checksum</td><td>Urgent Pointer</td></tr>
<tr><td colspan="3">Options & Padding
(at most 40B long)</td></tr>
</table>

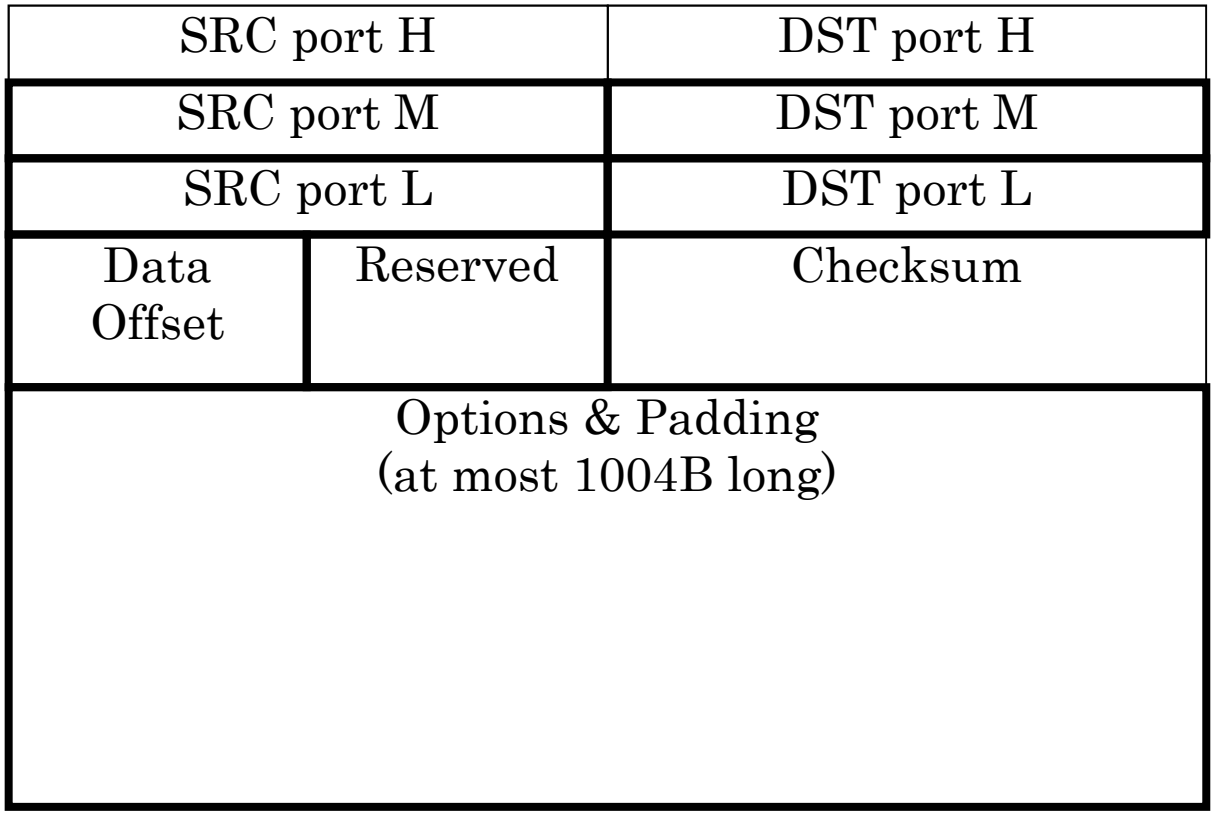
Header Format of TUPLE TCP



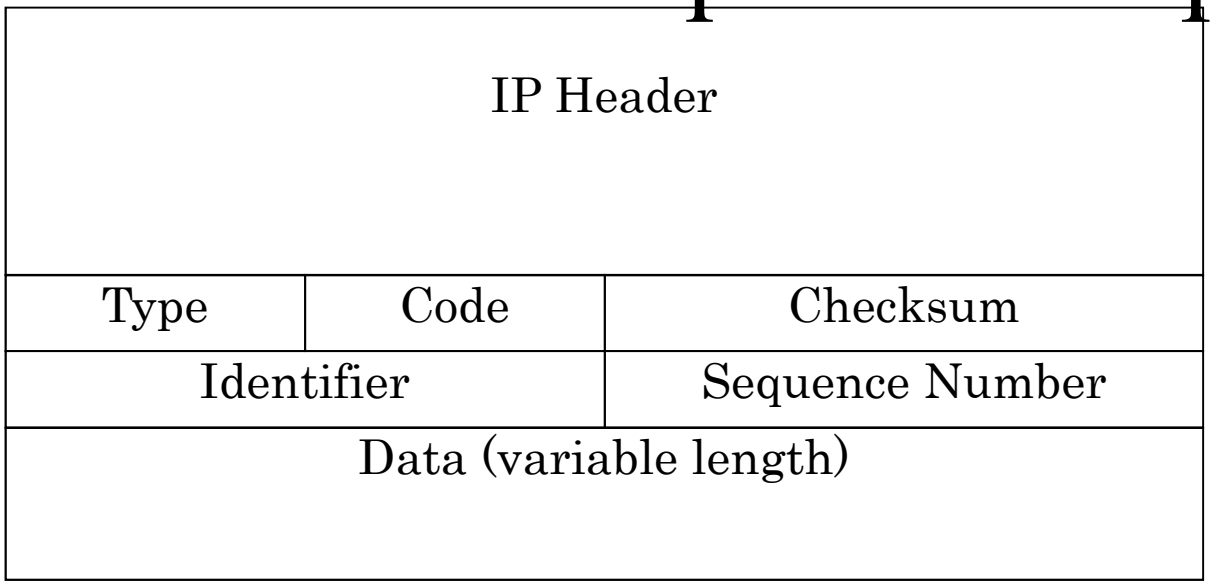
Header Format of the Current UDP



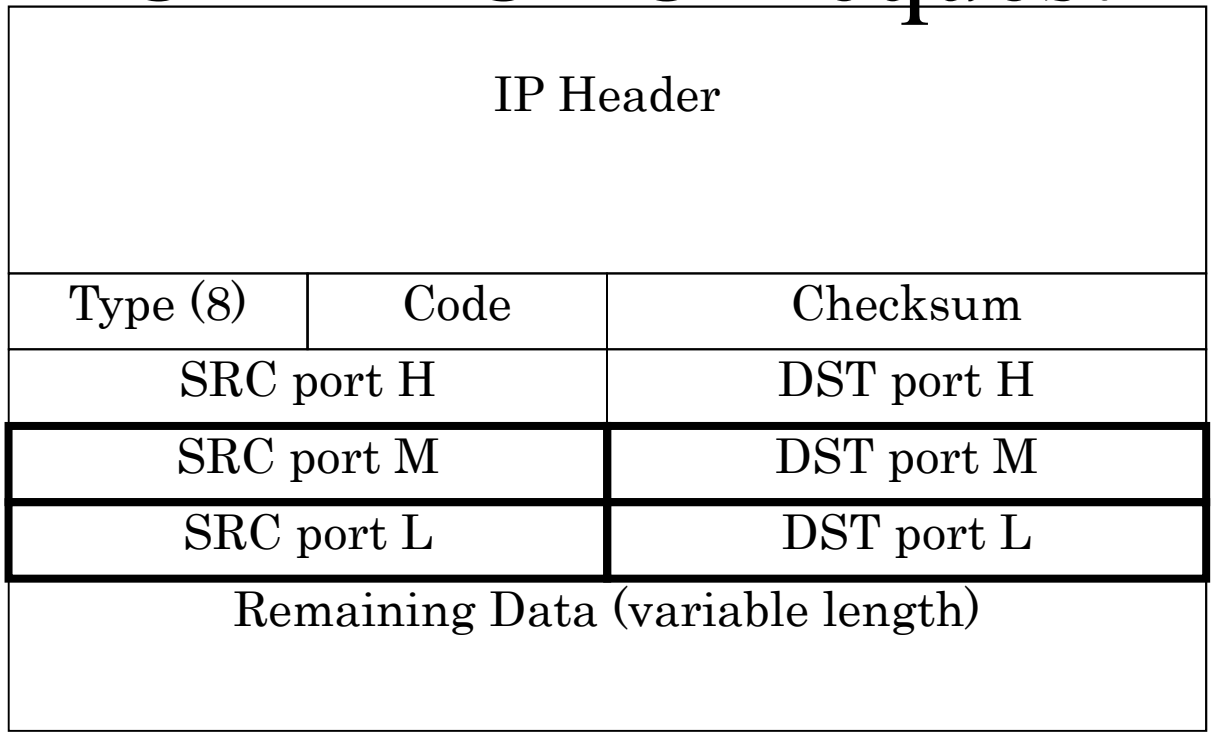
Header Format of TUPLE UDP



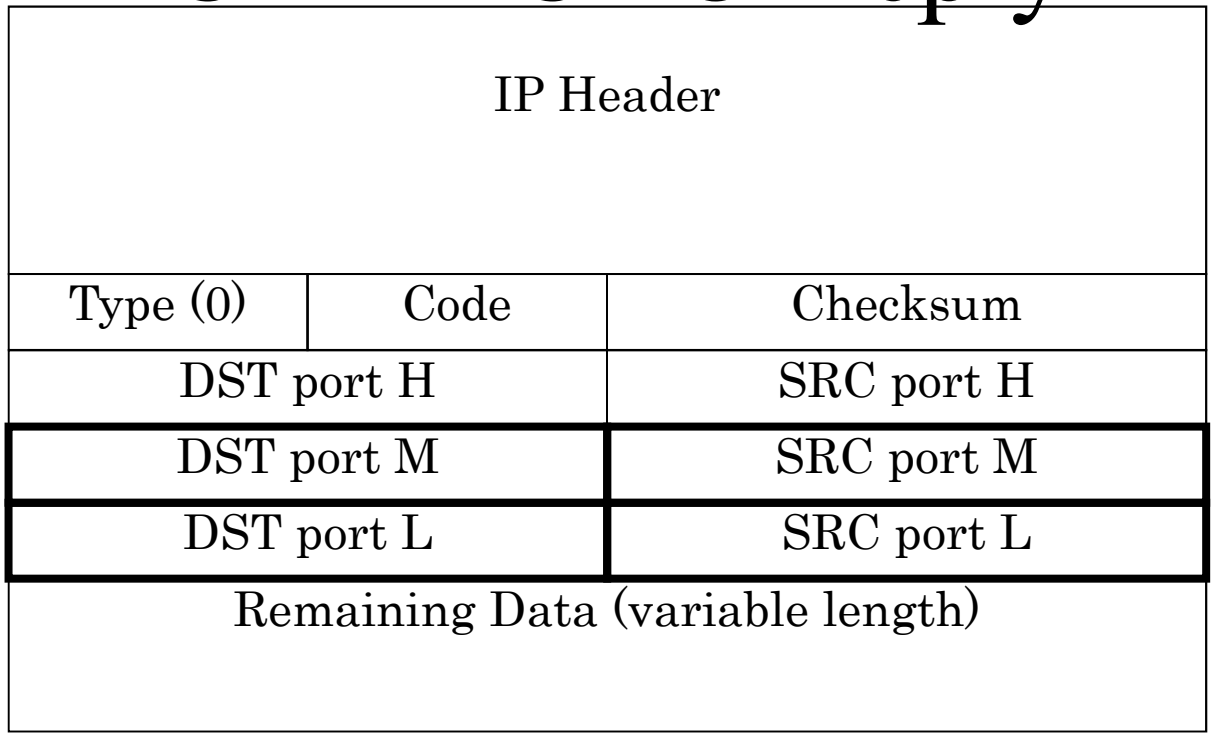
Packet Format of the Current ICMP ECHO Request & Reply



Packet Format of TUPLE ICMP ECHO Request



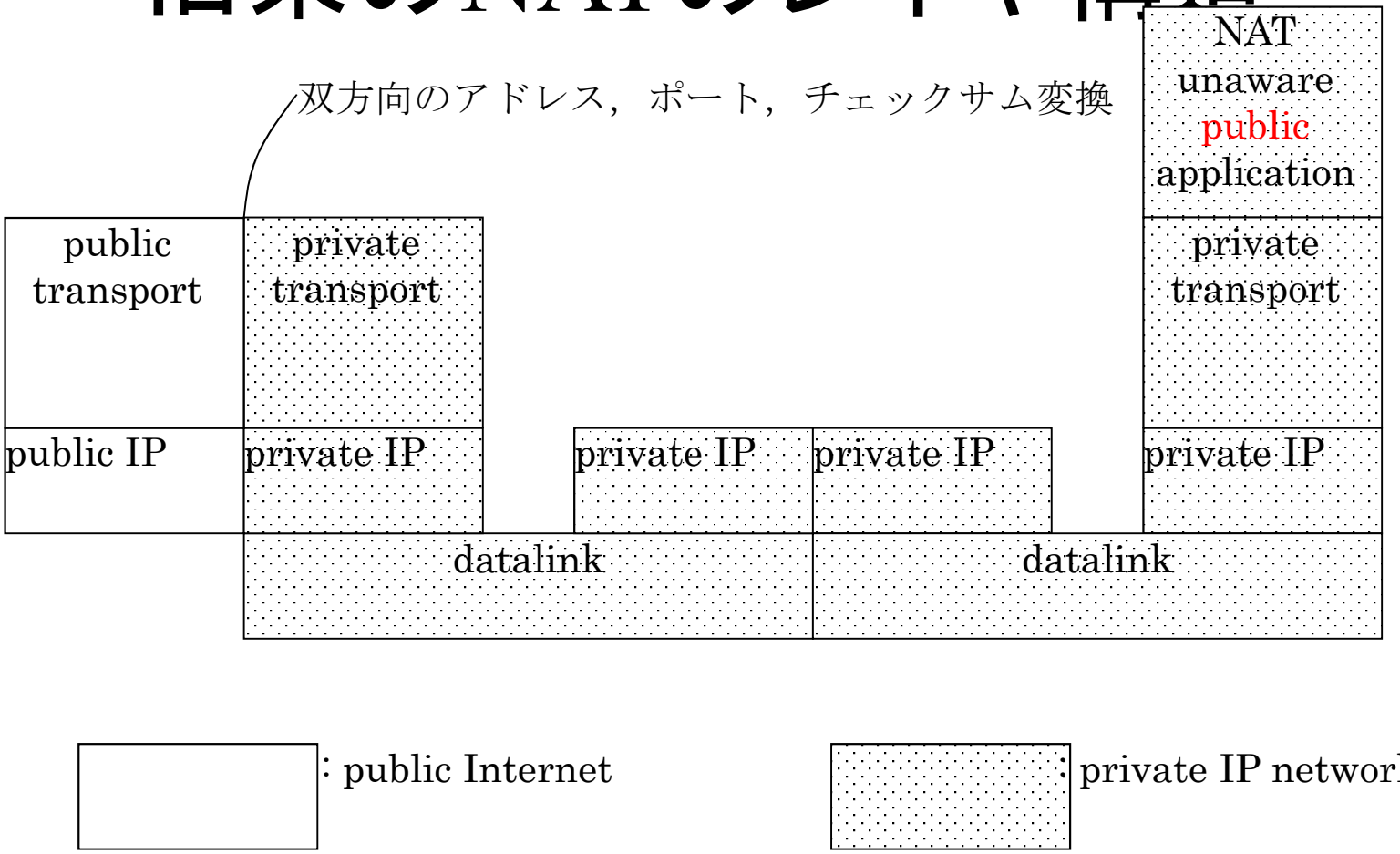
Packet Format of TUPLE ICMP ECHO Reply



Almost End to End NAT

太田昌孝
東京工業大学情報理工学研究科
mohta@necom830.hpcl.titech.ac.jp

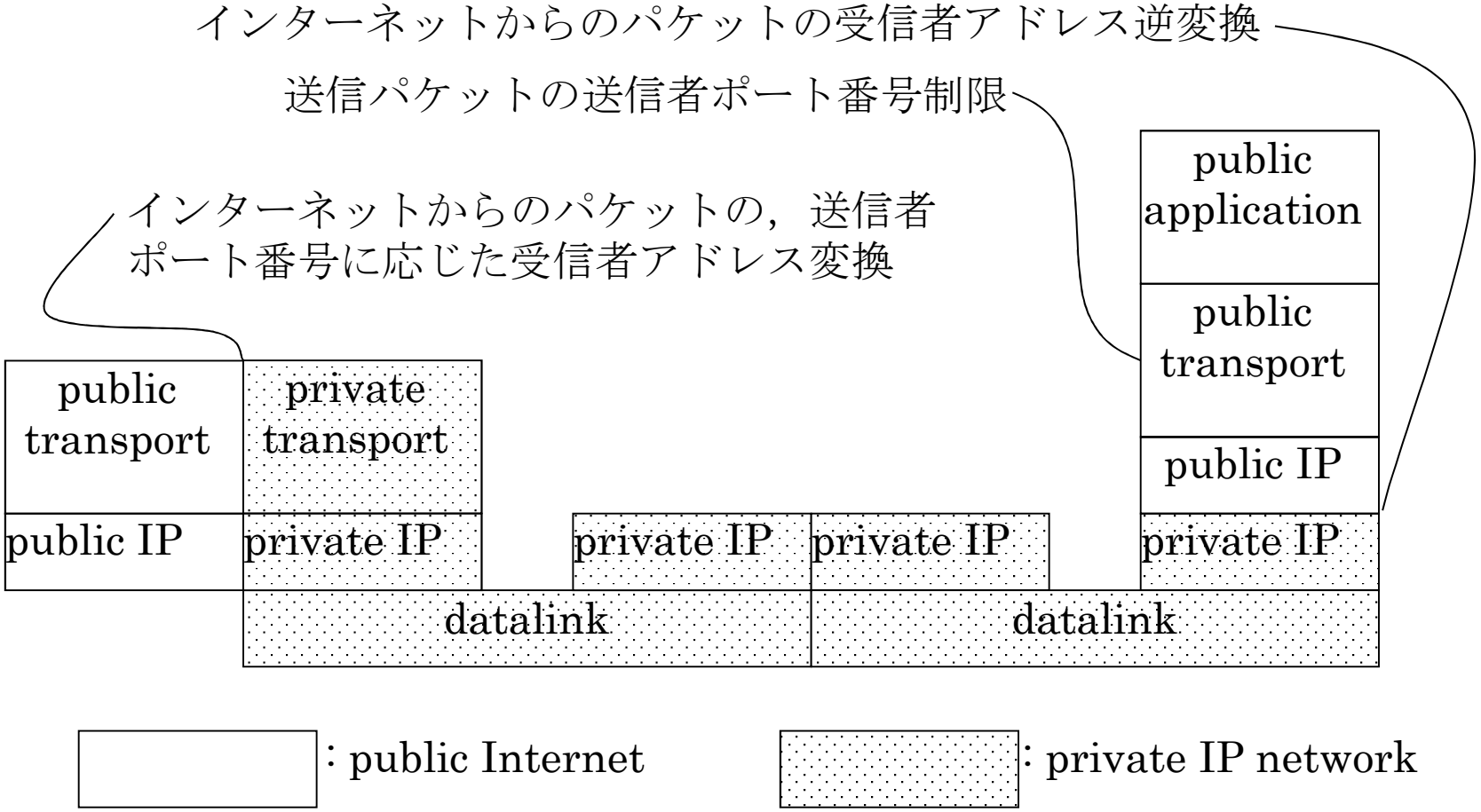
旧来のNATのレイヤ構造



End to End NAT

- NATの機能をなるべく端末に負わせる
  - NATゲートウェイではほとんど何もしない
    - ポート番号により受信者アドレスを変換するだけ
      - ポート番号やトランスポートチェックサムはそのまま
  - 端末では、受信者アドレスを元に戻す
    - トランスポートチェックサムは、自動的に正しくなる
  - 端末が送信するパケットの送信者アドレスは、グローバルアドレス、送信者ポート番号はその端末に割り当てられたポート番号に限定
    - ポート番号の衝突はなく、ポート変換の必要なし

エンドツーエンドNATの
レイヤ構造



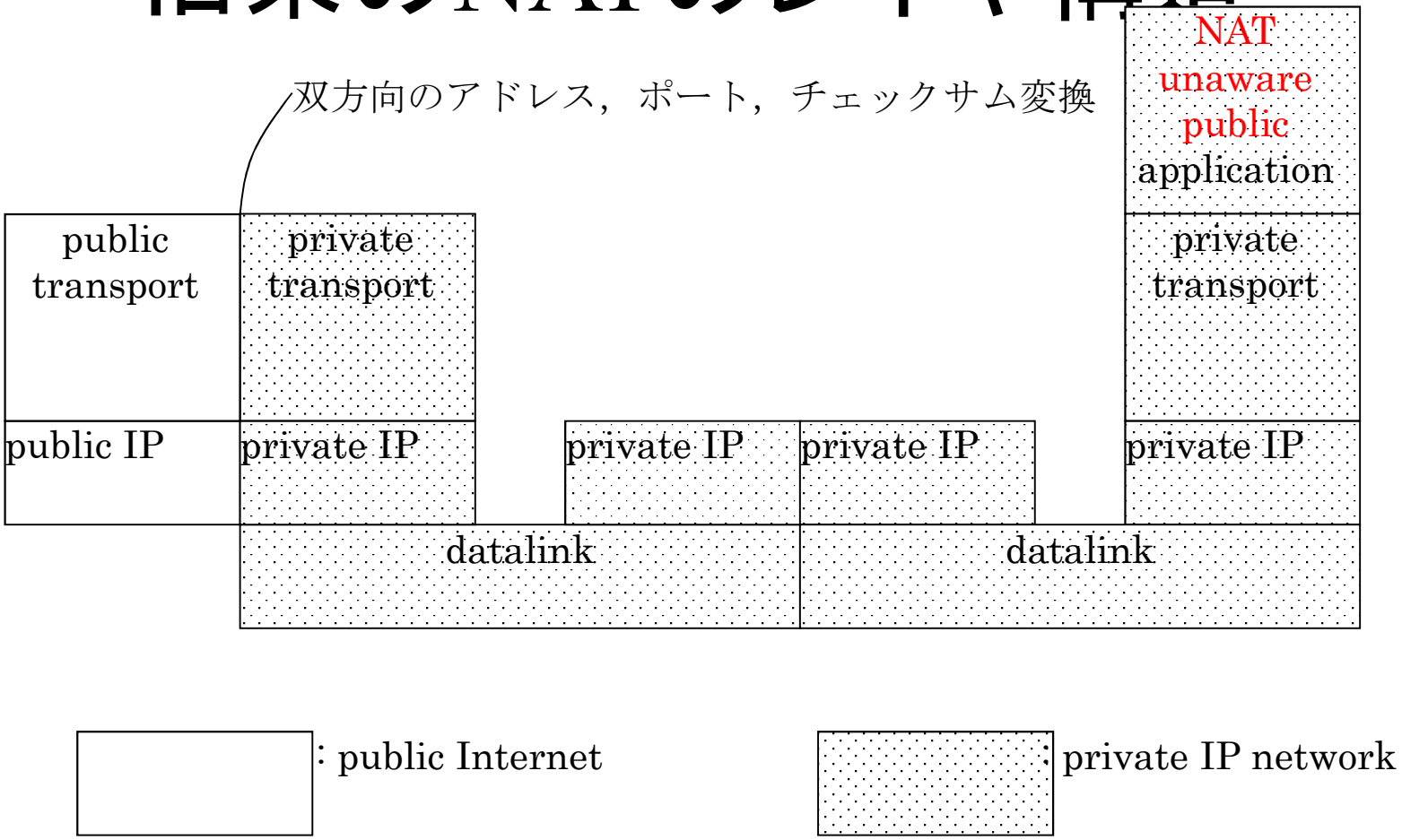
End to End NATの特長と問題点

- 全てのトランスポート層プロトコルが動作
  - ポート番号さえあれば（ICMP　EchoのIDやSeq. No、IPSECのSPI等もポート番号とみなせる）
- 普及させるのは容易ではない！！
  - NATゲートウェイの改造が必要
    - 旧来のNATゲートウェイとの両立も可能だが、、、
  - 端末の構成情報（グローバルアドレス、割り当てられたポート）をどう与えるか？
    - 新たなプロトコルが必要？IETFのPCP？

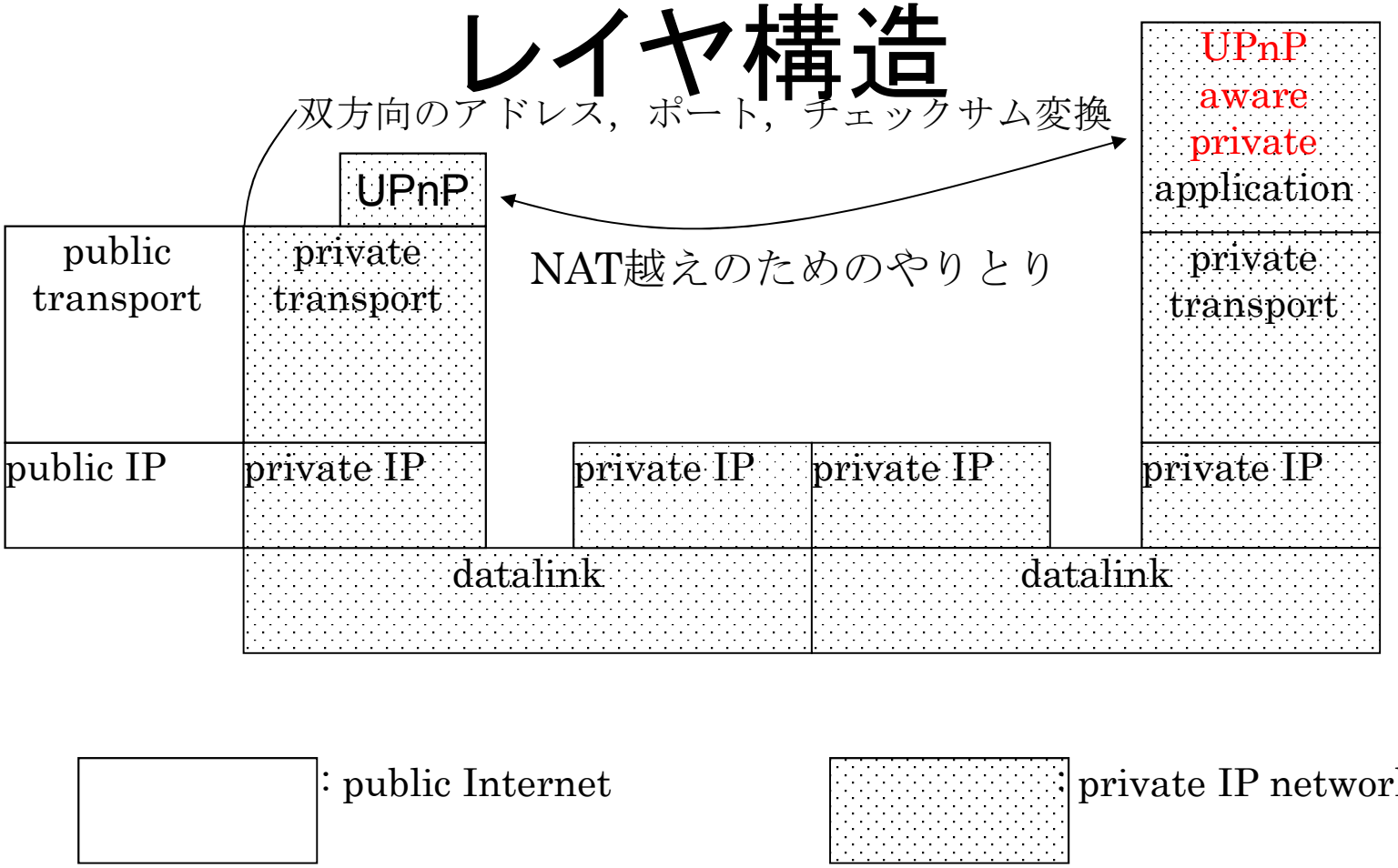
UPnP (Universal Plug and Play)

- 端末の自動構成のためのシステム
  - 端末がNAT越えするために、ポートマッピング情報を得るプロトコルを含む
    - WANIPConnectionサービスについての規定
  - 端末上のアプリケーションを、UPnPに対応したものに改造することを想定
- 多くのNATゲートウェイで実装されている
- トランスポート層としては、TCPとUDP（とICMP）にのみ対応

旧来のNATのレイヤ構造



UPnPの想定するレイヤ構造



UPnPと
End to End Argument

- UPnPは文字通り
  - with the knowledge and help of the application standing at the end points of the communication

ではあるが、アプリケーションで対応すると、

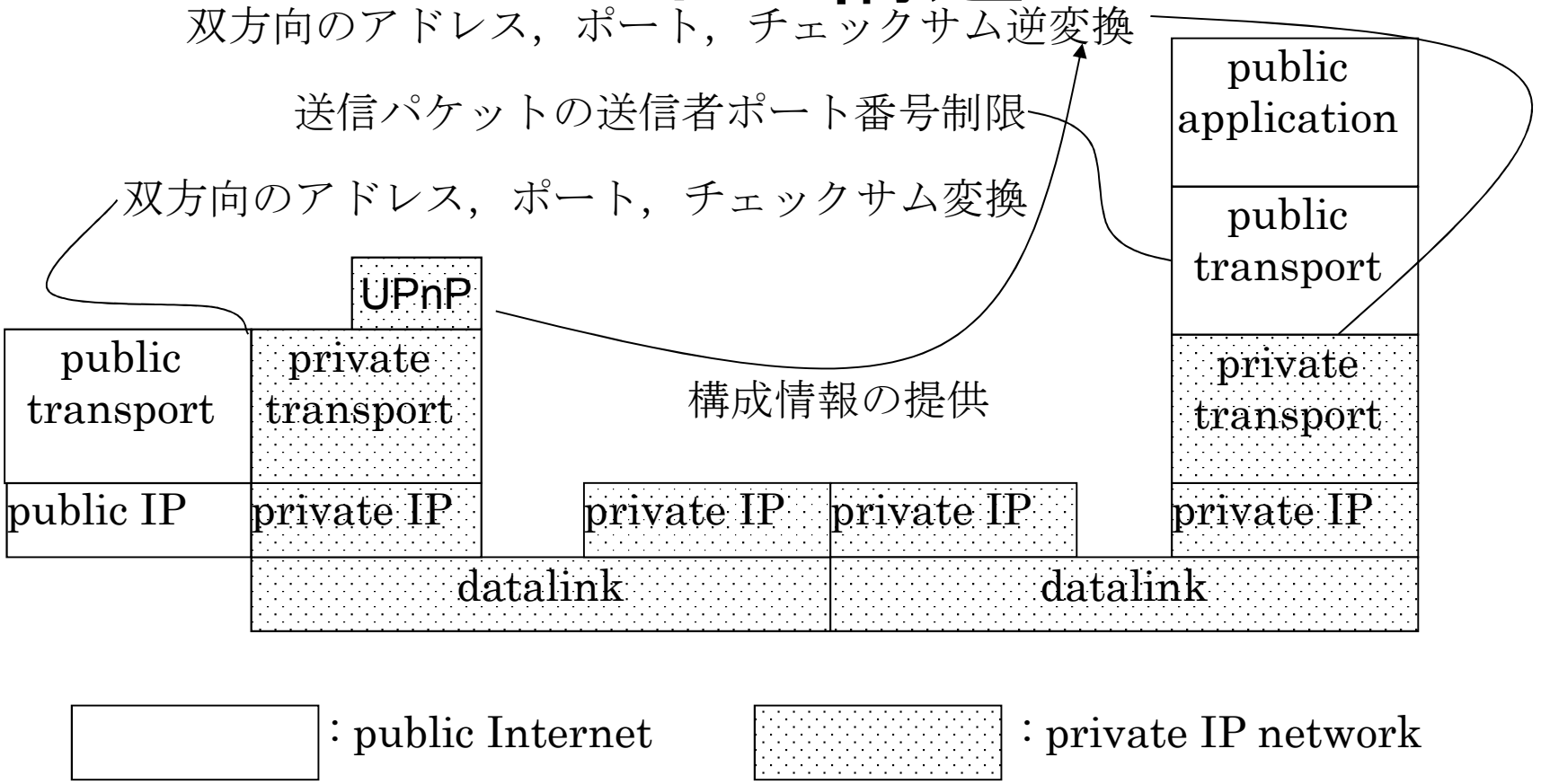
アプリケーションの変更が必要で、ださい

- End to End Argument当時のスタックは未分化で、今のアプリケーション層ではない
  - もっと下の層でなんとかしてもいい

Almost End to End NAT

- UPnP GWを利用したEnd to End NAT
- UPnP GWのポートマッピング（双方向）を、端末のトランスポート層で逆変換
  - アプリケーションに見える自分のアドレスはグローバルアドレス、送信者ポート番号はその端末に割り当てられたポート番号に限定
    - アプリケーションの改造は、一切不要
      - アプリケーションが使えるポート番号と数は制約されるが
  - TCP、UDPとICMP）しか使えないので、Almost

Almost End to End NATの レイヤ構造



Almost End to End NATの特長

- UPnP対応の既存NATゲートウェイがそのまま使える
- ゲートウェイ構成情報は、UPnPで取得可
  - グローバルアドレス：GetExternalIPAddress()
  - ポート変換情報：GetListOfPortMappings()
    - UPnP第一版では、GetGenericPortMappingEntry()
- 端末側の実装は、NetBSD5.1上のEnd to End NATの実装を手直しすれば、容易
  - ゲートウェイ上でポート番号を変換しなければ

NATによるポート番号不足？

- サーバが受けるポートは一つでいいが、クライアントがサーバに多数の要求を出すポートは多数必要？
  - 実際、Google Mapはポート番号を大量消費
  - 適切な実装により、回避可能
    - 実装が、ポート番号の一時的な不足（旧来のNATでは不可知だが、(Almost) End to End NATではEAGAINのエラー）に適切に対処すればよい
    - クライアントの送信者ポートは、setsockoptでREUSEPORTを設定すれば、多数のconnectで共有可
      - ソケットを共有ポートにbindしてから、connect

結論

- Almost End to End NATにより:
  - 既存のUPnP対応NATゲートウェイ背後の端末上で、UPnP非対応のTCP/UDP上のアプリケーションが、End to End透過性を保ちつつ動作可
  - IPv4アドレスの金銭取引で価格が高騰すれば、このような技術への追い風となる
- IPv4アドレス空間は、当分保つ
  - その間にクラスEを解放すれば、更に長く保つ
- 今後の課題は、URL全般へのSRVの導入

まとめ

- E2ENATは、現状のインターネット環境をエンドツーエンド透過性も含めほとんど全て保ちながら、アドレスを大幅に節約
- E2ENAT前提のアドレス管理により、IPアドレス空間は（クラスEも使えばなおさら）当分持つ
- IPv6？なにそれ？