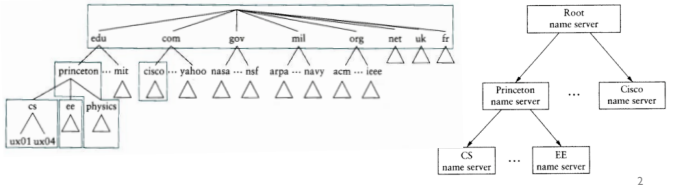
DNS Caching

- Once a (any) name server learns mapping, it caches mapping
  - to reduce latency in DNS translation
- Cache entries timeout (disappear) after some time (TTL)
  - TTL assigned by the authoritative server responsible for the host name
- Local name servers typically also cache
  - TLD name servers to reduce visits to root name servers
  - all other name server referrals
  - both positive and negative results

DNS Name Resolution Exercises

Show the DNS resolution paths, assuming the DNS hierarchy shown in the figures and assuming caching:

- `thumper.cisco.com` looks up `bas.cs.princeton.edu`
- `thumper.cisco.com` looks up `opt.cs.princeton.edu`
- `thumper.cisco.com` looks up `cat.ee.princeton.edu`
- `thumper.cisco.com` looks up `ket.physics.princeton.edu`
- `bas.cs.princeton.edu` looks up `dog.ee.princeton.edu`
- `opt.cs.princeton.edu` looks up `cat.ee.princeton.edu`



Peterson & Davie 2nd. ed., pp. 627, 628

DNS Design Points

DNS serves a core Internet function

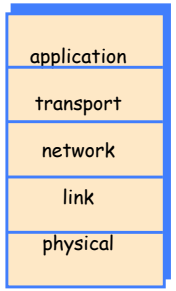
At which protocol layer does the DNS operate?

- host, routers, and name servers communicate to **resolve** names (name to address translation)
- complexity at network's "edge"

Why not centralize DNS?

- single point of failure
- traffic volume
- performance: distant centralized database
- maintenance
- ➡ doesn't scale!

DNS is "exploited" for server load balancing, how?

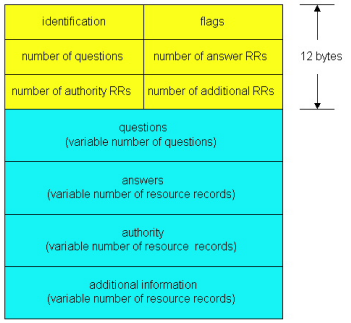


DNS protocol, messages

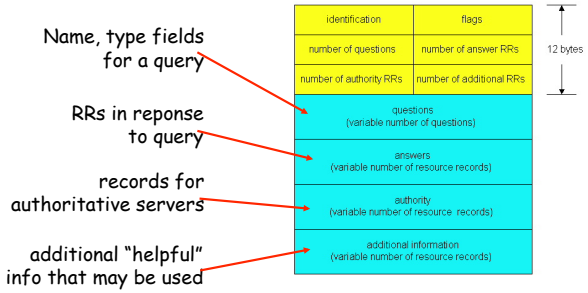
DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



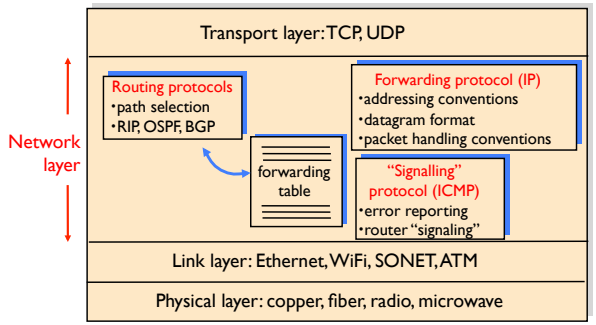
DNS protocol, messages



5

The Internet Network Layer

Host, router network layer functions:



6

Packet and Packet Header

Previously . . . the Internet is a **packet switched network**:

data is parceled into packets
each packet carries a destination address
each packet is routed independently
packets can arrive out of order
packets may not arrive at all

Just as with the postal system, the "content" you want to send must be put into an envelope and the envelope must be addressed

The "envelope" in this case is the **packet header**

Recall: **protocols** are rules ("syntax" and "grammar") governing communication between nodes

The **format** of a packet header is part of a protocol

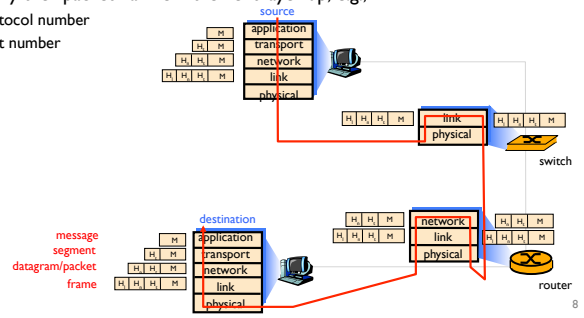
For packet forwarding on the Internet, the protocol is the **Internet Protocol (IP)**

7

Encapsulation

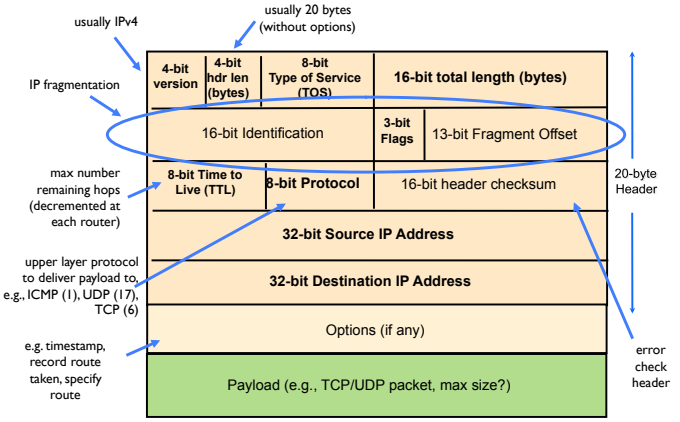
Each protocol has its own "envelope"

- each protocol attaches its header to the packet
- so we have a protocol wrapped inside another protocol
- each layer of header contains a protocol demultiplexing field to identify the "packet handler" the next layer up, e.g.,
  - protocol number
  - port number



8

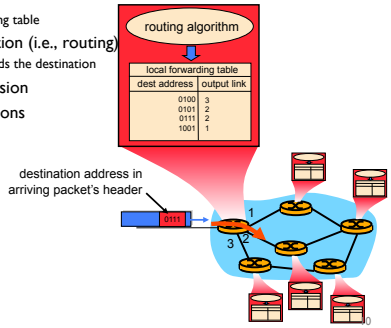
IPv4 Packet Header Format



Packet Forwarding

Goal: deliver packets through routers from source to destination

- source node puts destination address in packet header
- each router node on the Internet:
  - looks up destination address in its routing table
    - we'll study several path selection (i.e., routing)
  - sends the packet to the next hop towards the destination
    - routes may change during session
    - analogy: driving, asking directions

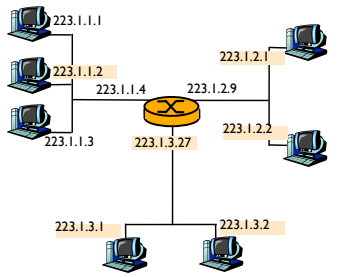


IP Addressing: Introduction

IP address: 32-bit identifier for host/router **interface**

interface: connection between host/router and physical link

- routers typically have multiple interfaces
- host may have multiple interfaces
- IP address associated with each interface



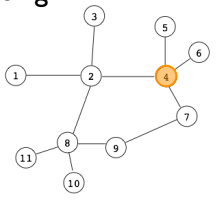
223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Flat vs. Hierarchical Addressing

Flat addressing:

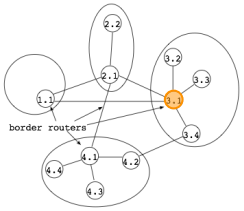
- each router needs 10 entries in its routing table



1	2
2	2
3	2
4	-
5	5
6	6
7	7
8	2
9	7
10	2
11	2

Hierarchical addressing:

- hosts only need to know the default router, usually its border router
- each border router keeps in its routing table:
  - next hop to other networks
  - all hosts within its own network



2.*	2.1
1.*	1.1
4.*	2.1
3.2	3.2
3.3	3.3
3.4	3.4

note that for routing table, we store the next hop address instead of the interface number

IPv4 Addressing

Independent of physical hardware address

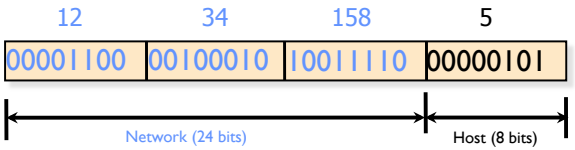
32-bit number represented as dotted decimal:

- for ease of reference
- each # is the decimal representation of an octet

Divided into two parts:

- network prefix, globally assigned
  - route to network first
- host ID, assigned locally

Example: 12.34.158.0/24 is a 24-bit network prefix with 2^8 host addresses



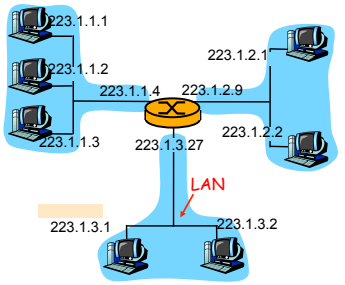
13

Subnets

A network can be further divided into subnets

What's a subnet ?

- device interfaces with same subnet part of IP address
- can physically reach each other without intervening router



a network consisting of 3 subnets

14

Classfull Addresses

For the example network prefix: 12.34.158.0/24

- how many hosts can the network have?

What is a good partition of the 32-bit address space between the network and host parts?

Historically . . . classfull addresses:

Class A: 0*, very large /8 blocks (e.g., MIT has 18.0.0.0/8)
Class B: 10*, large /16 blocks (e.g,. UM has 141.213.0.0/16)
Class C: 110*, small /24 blocks (e.g., AT&T Labs has 192.20.225.0/24)
Class D: 1110*, multicast groups
Class E: 11110*, reserved for future use

Problems:

1. the Goldilock problem: everybody wanted a Class B
2. address space usage became inefficient
3. routing table explosion
4. and then, address space became scarce…
   - by 1992, half of Class B has been allocated, would have been exhausted by 3/94

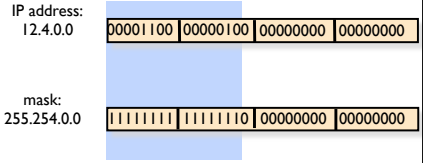
15

Classless InterDomain Routing (CIDR)

Network portion of address is of arbitrary length, determined by a prefix mask

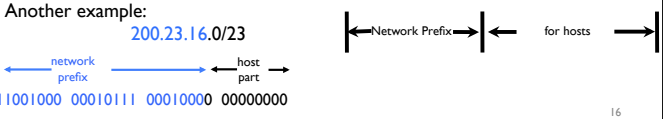
Uses two 32-bit numbers to represent a network address
network number = IP address + mask

Usually written as a.b.c.d/x, where x is number of bits in the network portion of address: 12.4.0.0/15



Another example:

200.23.16.0/23

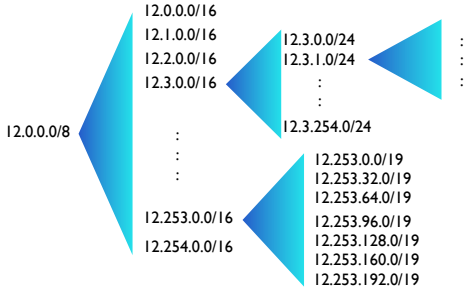


16

CIDR: Hierarchical Address Allocation

Prefixes are key to Internet routing scalability

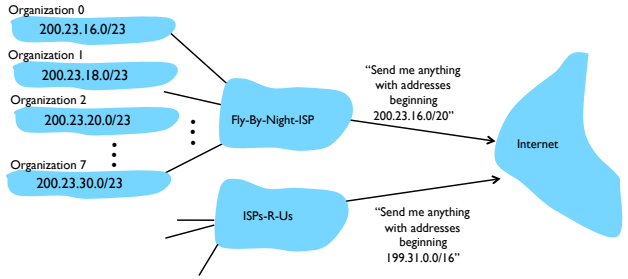
- address allocation by ICANN, ARIN/RIPE/APNIC and by ISPs
- routing protocols and packet forwarding based on prefixes
- today, routing tables contain ~150,000-200,000 prefixes



17

CIDR: Route Aggregation

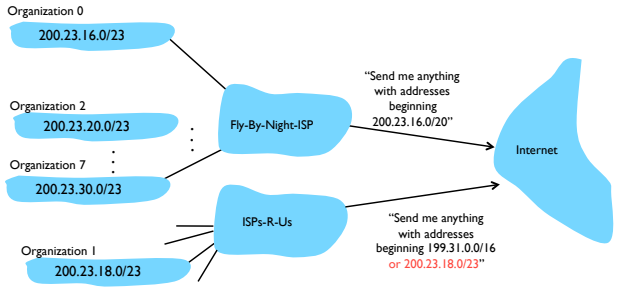
Hierarchical addressing allows efficient advertisement of routing information:



18

Longest Prefix Match: More specific routes

ISPs-R-Us has a more specific route to Organization 1

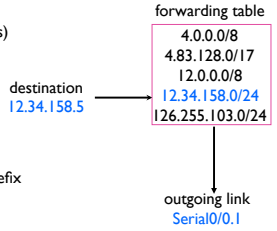


19

How are Packets Forwarded?

Routers have forwarding tables

- maps each IP prefix to next-hop link(s)
- entries can be statically configured
  - e.g., "map 12.34.158.0/24 to Serial0/0.1"



Destination-based forwarding

- Packet has a destination address
- Router identifies longest-matching prefix

But, this doesn't adapt

- to failures
- to new equipment
- to the need to balance load
- …

That is where routing protocols come in… [more on this in the next lectures]

20

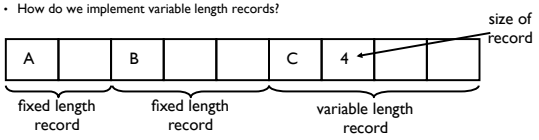
Special IPv4 Addresses

- network identification:
  - 0s on host part, e.g. ,`141.212.0.0` (cannot be used to send packets)
- directed broadcast:
  - `0xffff` on host part, e.g., `141.212.255.255`
  - Broadcast to all hosts on network (`141.212`) (Not implemented?)
- limited broadcast:
  - `0xffffffff`, received by all hosts on LAN, not forwarded beyond LAN
- this computer:
  - `0.0.0.0` to be used at startup to ask for one's own IP address (RARP, deprecated)
- loopback address:
  - `127.*.*.*` (usually `127.0.0.1`), named `localhost`
  - pkts sent to `localhost` traverse down the kernel networking code & back up to application without traversing the network, useful for testing networking code

PA1 Topics

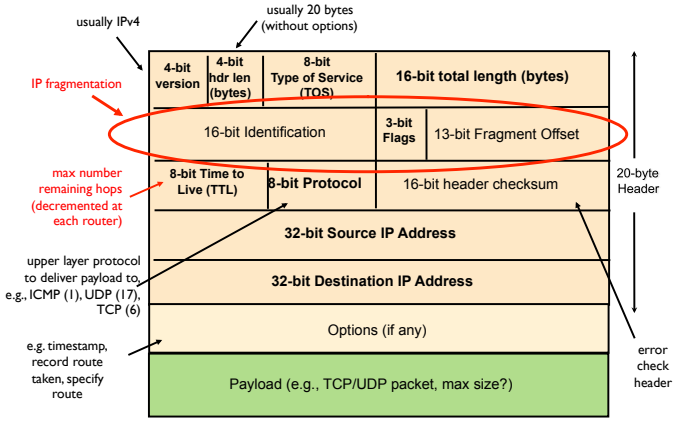
Replace calls to my "hidden" functions
- leaving them "as is" is considered cheating!

Stream socket data separation:
- Use records (data structures) to partition data stream
- Three ways to tell the end of a record
  - How do we implement variable length records?



How to populate and read a struct, e.g., when `vip_input()` calls `vif_pullup()`?

IPv4 Packet Header Format



ICMP: Internet Control Message Protocol

Used by hosts & routers to communicate network-level information
- error reporting: unreachable host, network, port, protocol
- echo request/reply (used by ping)

Network-layer "above" IP:
- ICMP msgs carried in IP datagrams

ICMP message:
- type
- code
- plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	4	frag needed but DF set
3	6	dest network unknown
3	7	dest host unknown
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

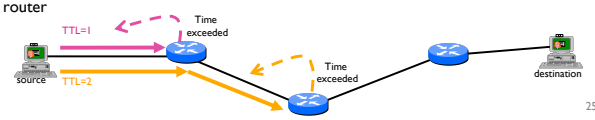
Source sends a series of UDP packets to dest
- first pkt has TTL =1
- second pkt has TTL=2, etc.
- sent to a (probably) unused port number

When *n*-th packet arrives at *n*-th router:
- router discards packet
- and sends to source an ICMP message (type 11, code 0)
- message includes IP address of router

When ICMP message arrives back at the source, source calculates RTT

Traceroute does this 3 times per TTL

Stopping criterion:
- UDP pkt eventually arrives at destination host
- Destination returns ICMP "destination port unreachable" message (type 3, code 3)
- When source gets this ICMP message, stops sending UDP packets

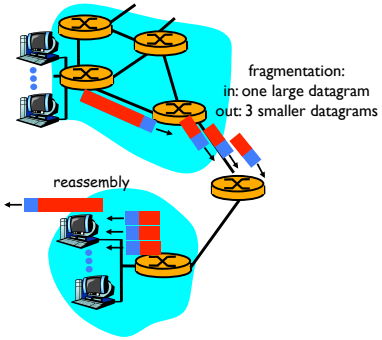


25

IP Fragmentation & Reassembly

Network links have MTU (max transfer size) limit - the largest possible link-level frame
- different link types, different MTUs

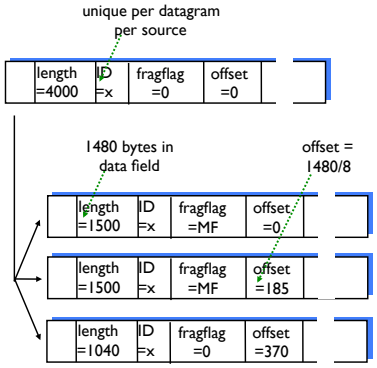
Large IP datagrams are split up ("fragmented") within the network
- one datagram becomes several datagrams
- each with its own IP header
- fragments are "reassembled" only at final destination (why?)
- IP header bits used to identify and order related fragments



26

IP Fragmentation and Reassembly

Example: 4000 byte datagram
MTU = 1500 bytes



One large datagram becomes several smaller datagrams
- all but the last fragments must be in multiple of 8 bytes
- offsets are specified in unit of 8-byte chunks

27

Fragmentation Considered Harmful

Reason 1: Lose 1 fragment, lose whole packet:
- kernel has limited buffer space
- but IP doesn't know number of fragments per packet

For example:
- Sender sends two pkts, *L* and *S*
- *L* is fragmented into 8 fragments
- *S* is fragmented into 2 fragments
- Receiver has 8 buffer slots
- Suppose fragments arrive in the following order:
  L1, L2, L3, L4, L5, L6, L7, S1, L8, S2
- Receiver's buffer fills up after *S1*, both packets thrown away when reassembly timer times out

Reason 2: Inefficient transmission

Example:
- 10 KB of data
- sent as 1024 byte TCP segments
- uses 10 IP packets, each 1064 bytes
- Suppose MTU is 1006 bytes
- each TCP packet is fragmented into 2 pkts of 1006 bytes & 78 bytes
- ends up sending 20 packets
- If TCP had sent 966-byte segments, only need to send 11 pkts

28

Fragmentation Considered Harmful

Analysis:

- IP doesn't have control over number of fragments
- TCP can do buffer management better because it has more information

Alternatives to fragmentation:

- send only small datagrams (why not?)
- do path MTU discovery and let TCP send the appropriate segment sizes
  - set DF flag
  - router returns ICMP error message (type 3, code 4) if fragmentation necessary
- IPv6 enforces min MTU of 576 bytes, no fragmentation at routers

29

IPv6

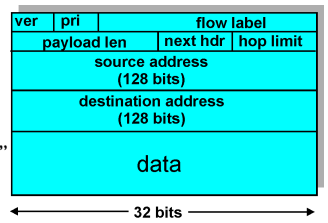
Initial motivation: 32-bit address space exhaustion

Additional motivation:

- header format helps speed processing/forwarding
  - fixed-length 40 byte header (0.06% overhead)
  - header checksum: removed entirely to reduce processing time at each hop
  - options: allowed, but outside of header, indicated by "next header" field
- header changes to facilitate QoS:
  - priority: identify priority among datagrams in flow (ToS bit)
  - flow label: identify datagrams in the same "flow" (concept of "flow" not well defined, originally these were "reserved" bits)

Next header identifies "upper layer" protocol or IPv6 options:

- hop-by-hop option, destination option, routing, fragmentation, authentication, encryption



30

IPv6 Addresses

What does an IPv6 address look like?

- 128 bits written as 8 16-bit integers seperated by ':'
- each 16 bit integer is represented by 4 hex digits

Example: `FEDC:BA98:7654:3210:FEDC:BA98:7654:3210`

Abbreviations:

- actual: `1080:0000:0000:0000:0008:0800:200C:417A`
- skip 0's: `1080:0:0:0:8:800:200C:417A`
- double '::': `1080::8:800:200C:417A` but `not ::BA98:7654::`

IPv4: `10.0.0.1`
can be written as `0:0:0:0:0:0:A00:1` or `::10.0.0.1`

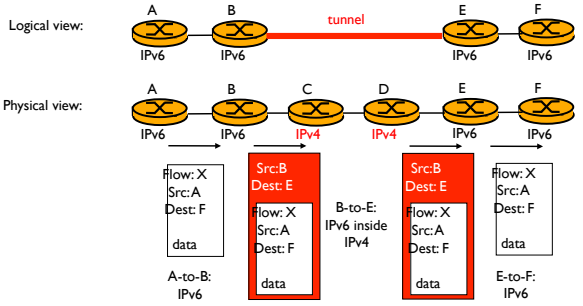
31

Tunneling

Not all routers can be upgraded simultaneous

- no "flag days"
- How will the network operate with mixed IPv4 and IPv6 routers?

Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers

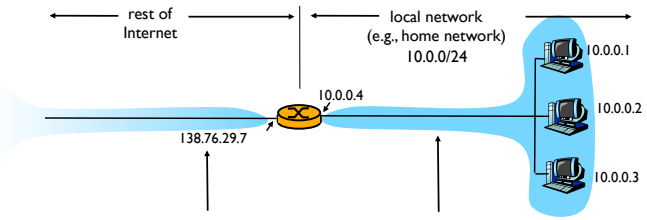


32

NAT: Network Address Translation

Motivation: a stop-gap measure to handle the IPv4 address exhaustion problem

- share a limited number (≥ 1) of global, static addresses by a number of local hosts
- local to global address binding done per connection, on-demand

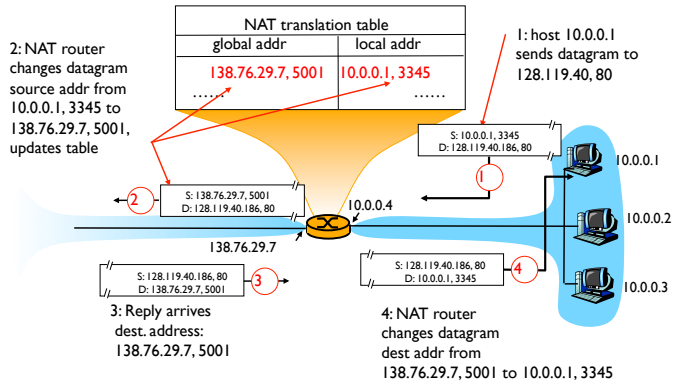


NAT: Network Address Translation

A NAT box functions:

- replaces `<source IP address, port #>` of every outgoing datagram to `<NAT IP address, new port #>`
  - remote hosts use `<NAT IP address, new port #>` as destination addr
- remember (in NAT translation table) every `<source IP address, port #>` to `<NAT IP address, new port #>` mapping
- replaces `<NAT IP address, new port #>` in dest field of every incoming datagram with corresponding `<source IP address, port #>` stored in the NAT table
  - forwards modified datagrams into the local network

NAT: Network Address Translation



NAT: Network Address Translation

Advantages:

- can change address of devices in local network without notifying outside world
- devices inside local net not explicitly addressable by or visible to the outside world (a security plus)

Disadvantage:

- devices inside local net not explicitly addressable by or visible to the outside world, making peer-to-peer networking that much harder
- routers should only process up to layer 3 (port#'s are app layer objects)
- address shortage should instead be solved by IPv6, instead NAT hinders the adoption of IPv6 (nothing wrong with that?)

Lesson:

Be careful what you propose as a "temporary" patch,
"temporary" solutions have a tendency to stay around beyond expiration date

"The evil that men do lives after them,
the good is oft interred with their bones."
– Shakespeare, *Julius Caesar*