Administrarea Bazelor de Date
Managementul în Tehnologia Informaţiei

Sisteme Informatice şi Standarde Deschise (SISD)

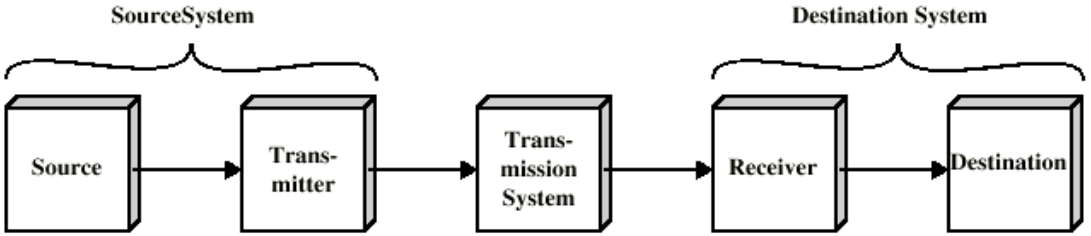
2009-2010

Curs 3

Standarde de comunicaţii

A Communications Model

- **Source**: generates data to be transmitted
- **Transmitter**: Converts data into transmittable signals
- **Transmission System**: Carries data
- **Receiver**: Converts received signal into data
- **Destination**: Takes incoming data



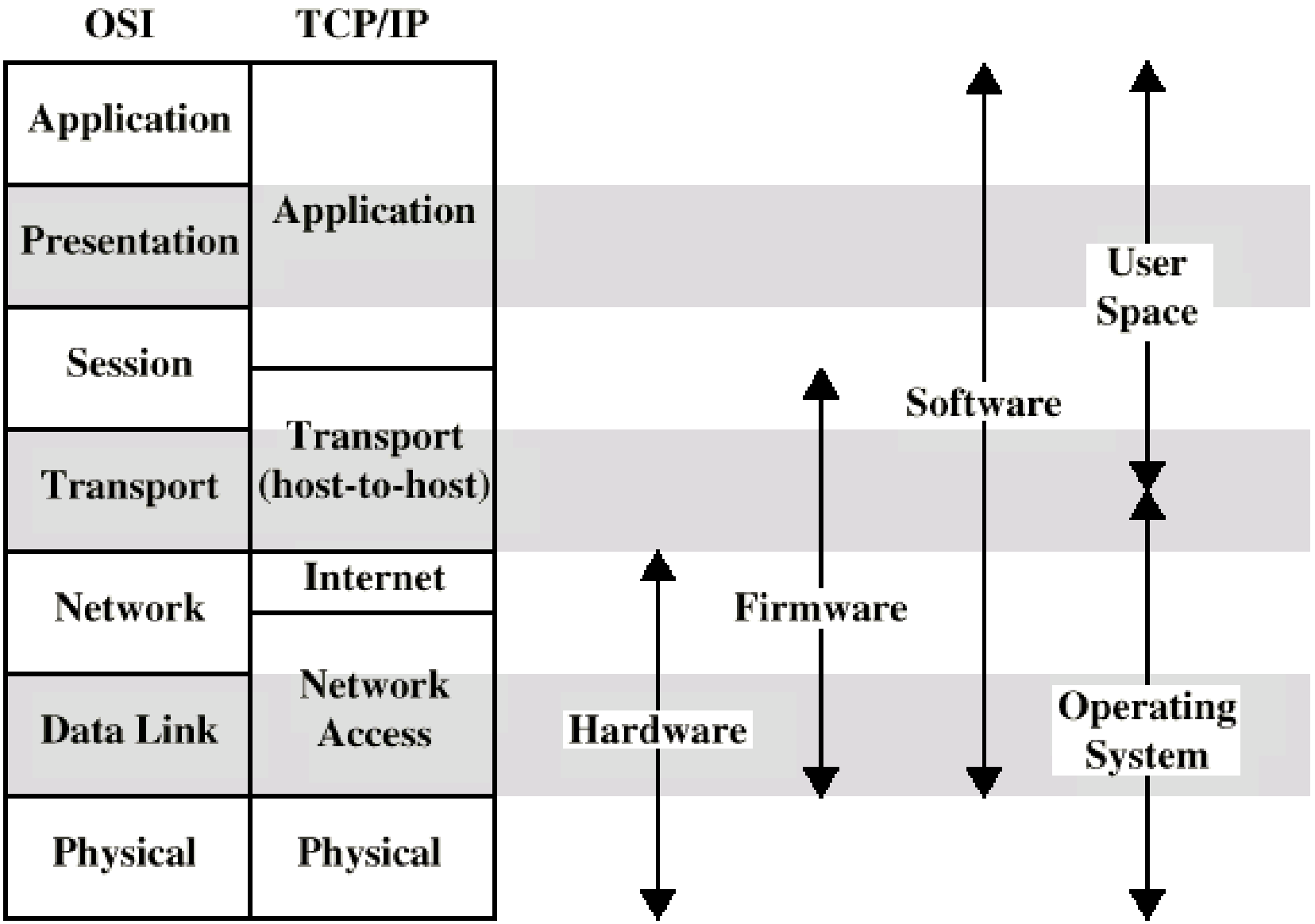
(a) General block diagram

(b) Example

Key Communications Tasks

- Transmission System Utilization
- Interfacing
- Signal Generation
- Synchronization
- Exchange Management
- Error detection and correction
- Addressing and routing
- Recovery
- Message formatting
- Security
- Network Management

OSI v TCP/IP


OSI
TCP/IP
Application
Presentation
Session
Transport
Network
Data Link
Physical
Application
Transport (host-to-host)
Internet
Network Access
Physical
Hardware
Firmware
Software
User Space
Operating System

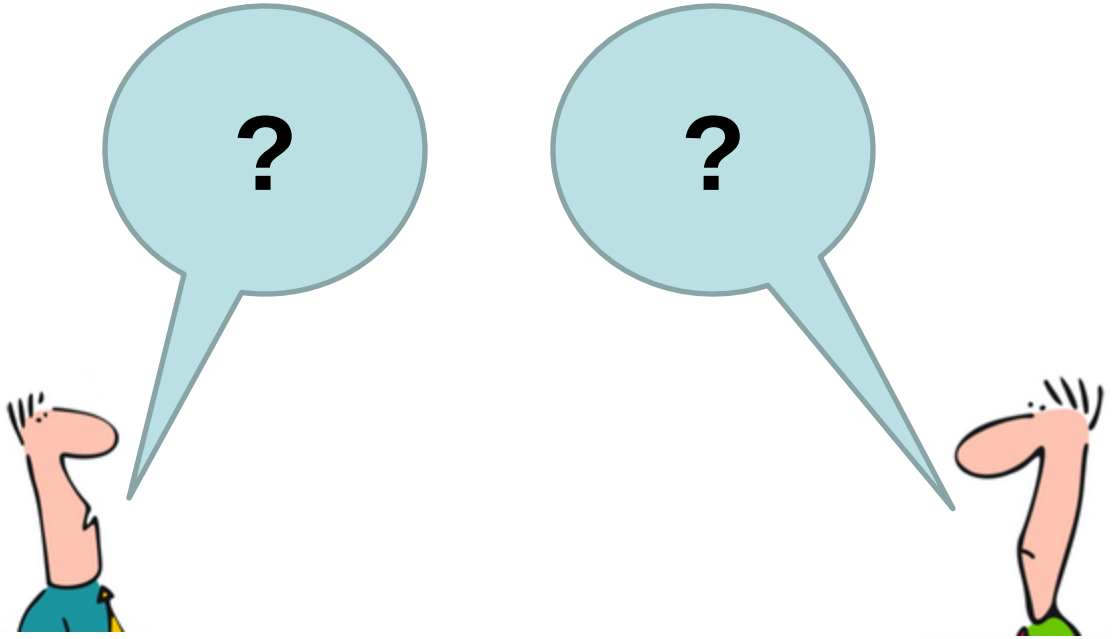
Standards Organizations

- Required to allow for interoperability between equipment
- Advantages
  - Ensures a large market for equipment and software
  - Allows products from different vendors to communicate
- Disadvantages
  - Freeze technology
  - May be multiple standards for the same thing

- Internet Society - http://www.isoc.org/
- ISO - http://www.iso.org
- ITU-T (formally CCITT) - http://www.itu.int
- ATM forum - http://www.broadband-forum.org/

Communication protocols

- Direct or indirect
- Monolithic or structured
- Symmetric or asymmetric
- Standard or nonstandard



Direct or Indirect

- Direct
  - Systems share a point to point link or
  - Systems share a multi-point link
  - Data can pass without intervening active agent
- Indirect
  - Switched networks or
  - Internetworks or internets
  - Data transfer depend on other entities

Monolithic or Structured

- Communications is a complex task
- To complex for single unit
- Structured design breaks down problem into smaller units
- Layered structure

Symmetric or Asymmetric

- Symmetric: Communication between peer entities
- Asymmetric: Client/server

Standard or Nonstandard

- Nonstandard protocols built for specific computers and tasks
- K sources and L receivers leads to K*L protocols and 2*K*L implementations
- If common protocol used, K + L implementations needed

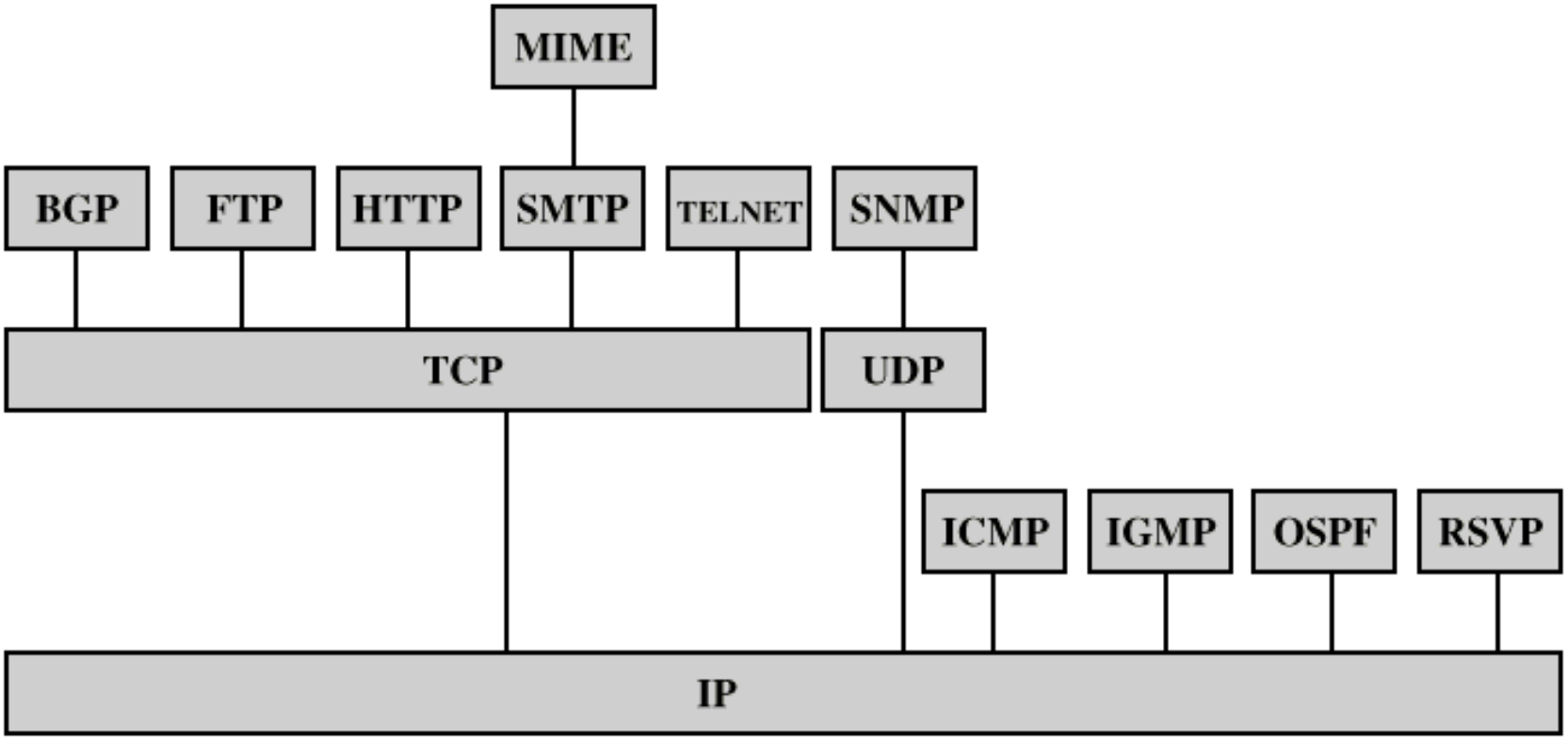
Communication Protocols - Functions

- Encapsulation
- Segmentation and reassembly
- Connection control
- Ordered delivery
- Flow control
- Error control
- Addressing
- Multiplexing
- Transmission services

Elements of Standardization

- Protocol specification
  - Operates between the same layer on two systems
  - May involve different operating system
  - Protocol specification must be precise
    - Format of data units
    - Semantics of all fields
    - allowable sequence of PCUs
- Service definition
  - Functional description of what is provided
- Addressing
  - Referenced by SAPs

Some Protocols in TCP/IP Suite

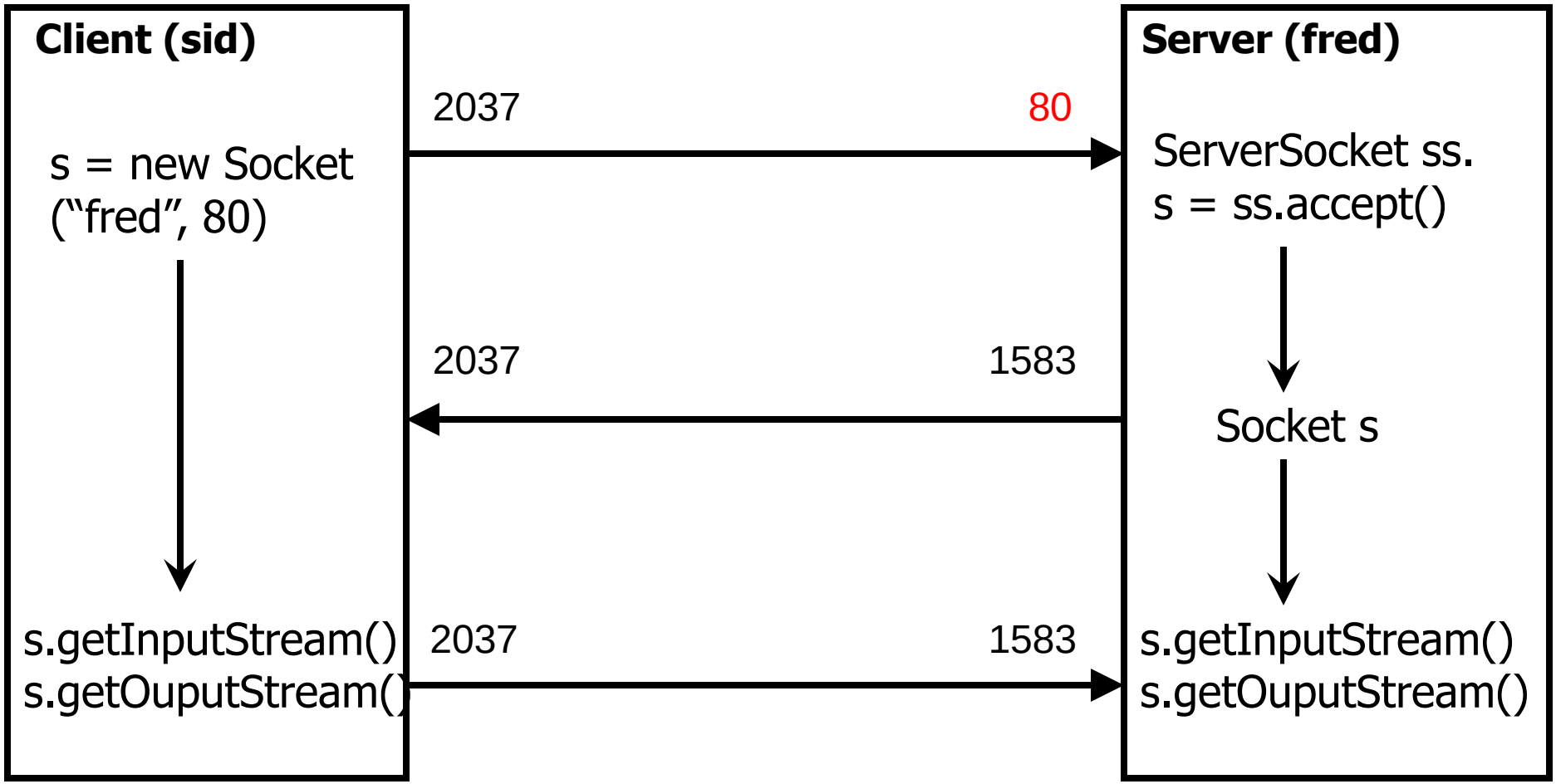


BGP = Border Gateway Protocol
FTP = File Transfer Protocol
HTTP = Hypertext Transfer Protocol
ICMP = Internet Control Message Protocol
IGMP = Internet Group Management Protocol
IP = Internet Protocol
MIME = Multi-Purpose Internet Mail Extension
OSPF = Open Shortest Path First
RSVP = Resource ReSerVation Protocol
SMTP = Simple Mail Transfer Protocol
SNMP = Simple Network Management Protocol
TCP = Transmission Control Protocol
UDP = User Datagram Protocol

Two types of TCP Socket

- `java.net.ServerSocket` is used by servers so that they can accept incoming TCP/IP connections
  - A server is a piece of software which *advertises* and then provides some service on request
- `java.net.Socket` is used by clients who wish to establish a connection to a (remote) server
  - A client is a piece of software (usually on a different machine) which makes use of some service

How it all fits together



A sample TCP server

```
public static void main(String[] args)
{
   try {
      ServerSocket agreedPort =
                  new ServerSocket(AGREED_PORT_NUMBER, 5);

      while (isStillServing()) {
         Socket session = agreedPort.accept();
         respond(session);
         session.close();
      }
      agreedPort.close();
   }
   catch (UnknownHostException uhe) {
      // Very unlikely to occur
   }
   catch (IOException ioe) {
      // May occur if the client misbehaves?
   }
 }
```

A sample TCP client

```
public static void main(String[] args)
{
   try {
      InetAddress server = InetAddress.getByName(args[0]);
      Socket connection =
                  new Socket(server, AGREED_PORT_NUMBER);
      makeRequestToServer(connection);
      getReplyFromServer(connection);
      connection.close();
   }
   catch (UnknownHostException uhe) {
      // arg[0] is not a valid server name or IP address
   }
   catch (IOException ioe) {
      // The connection to the server failed somehow:
      // the server might have crashed mid sentence?
   }
}
```

What are datagrams?

- Datagrams are discrete packets of data
- Each is like a parcel that can be addressed and sent to an recipient anywhere on the Internet
- This is abstracted as the User Datagram Protocol (UDP) in RFC768 (August 1980)
- Most networks cannot guarantee reliable delivery of datagrams
- Good for sending data that can naturally be divided into small chunks
- Poor for (lossless) stream based communications
- Makes economical use of network bandwidth (up to 3 times the efficiency of TCP/IP for small messages)
- Datagrams can be locally broadcast or multicast (one-to-many communication)

Application using datagrams

- UDP can be used for economical point-to-point communications over LANs
  - Unix NFS (Network File System)
  - NIS (e.g. Yellow Pages)
- Datagrams can be used for one-to-many communication:
  - Local network broadcasting;
  - Multicasting (MBONE)
- But there is no way to create one-to-many streams using TCP/IP

java.net.DatagramSocket (1)

- Used to represent a socket associated with a specific port on the local host
- Used to send *or* receive datagrams
- Note: there is no counterpart to `java.net.ServerSocket`! Just use a DatagramSocket with a *agreed* port number so others know which address and port to send their datagrams

java.net.DatagramSocket (2)

- Construction:
  - `DatagramSocket(int port)`
    - Uses a specified port (used for receiving datagrams)
  - `DatagramSocket()`
    - Allocate any available port number (for sending)
- Some useful methods:
  - `void send(DatagramPacket fullPacket)`
    - Sends the full datagram out onto the network
  - `void receive(DatagramPacket emptyPacket)`
    - Waits until a datagram and fills in emptyPacket with the message
- . . . and a few more in the Javadoc

Multicasting (1)

- Described in RFC1112 (August 1989)
- Multicasting allows distribution of a datagram to a group of listeners who are not within the local network
- Routers between networks need to pass multicast datagrams… but many do not!
- The MBONE is a way of tunneling datagrams across the Internet between *islands* of multicast activity
- Multicasts are also sent to a special address (known as a "*group*")
- Multicast groups need to be *agreed* in advance. They are not derived from a specific network/host address
  - Multicast groups identify a subject area (or stream of content) rather than a specific computer or network. They are more like a TV channel number than a telephone number.
- The IETF has set aside addresses from 224.0.0.1 to 239.255.255.255 specifically for multicasting

Multicasting (2)

- To send to (or receive from) a multicast group it is first necessary to *register interest* in the group
  - This results in an Internet Group Management Protocol (IGMP) message being sent to your router (RFCs 988/1112/2236)
- Then a datagram is created, addressed to the group (and the chosen port)
- Java has a specialised socket for multicasting: *java.net.MulticastSocket*

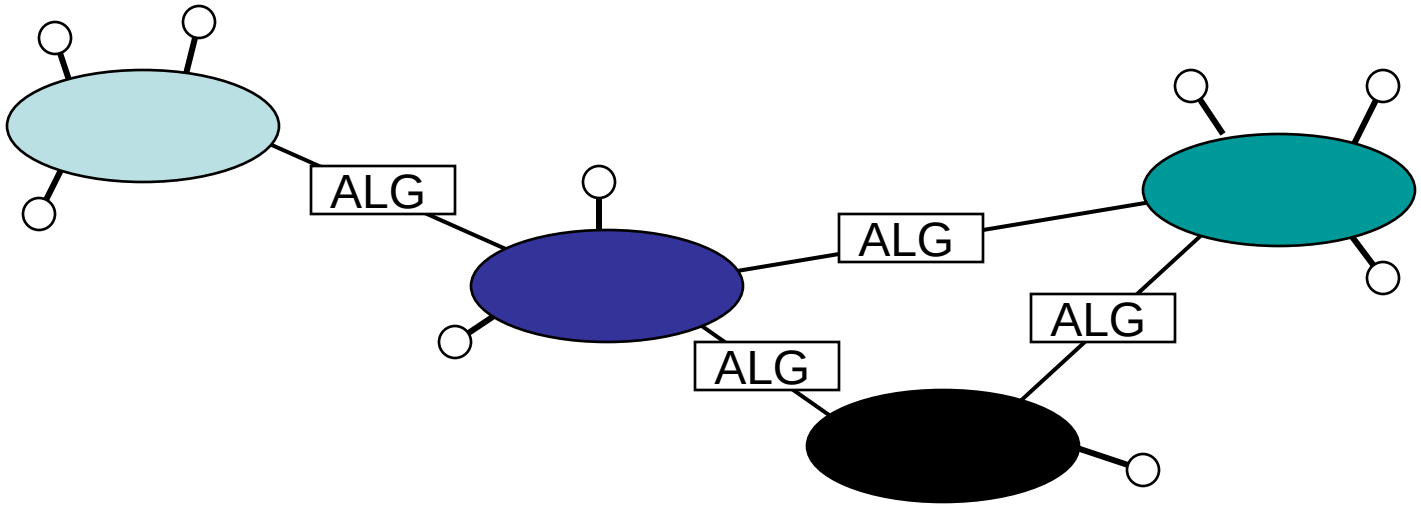
Some multicast groups

- 224.0.0.1 All hosts within local subnet
- 224.0.1.7 Audio news multicast
- 224.0.1.12 Video from IETF meetings
- 224.0.1.20 Expts. within local subnet
- 224.0.1.25 NBC Professional News
- There are 268 million multicast addresses (in IPv4) with 65 thousand ports in each!

java.net.MulticastSocket

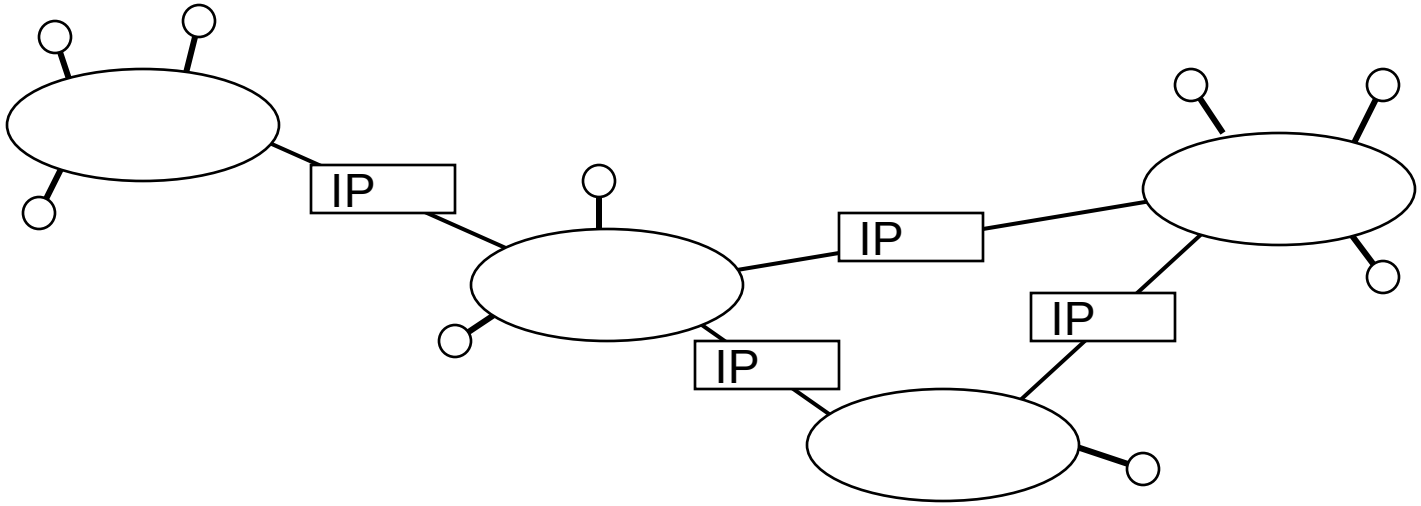
- Subclass of `java.net.DatagramSocket`
- Constructed the same way
- Adds some extra methods:
  - `void joinGroup(InetAddress mcastGroup)`
    - Enter the specifies group so that you can send or receive datagrams
  - `void leaveGroup(InetAddress mcastGroup)`
    - Leave a group that you previously joined
  - `void setTimeToLive(int ttl)`
    - Sets how far your datagrams will travel before routers ignore them
  - `int getTimeToLive()`

Life Before IP



- Application-layer gateways
  - inevitable loss of some semantics
  - difficult to deploy new internet-wide applications
  - hard to diagnose and remedy end-to-end problems
  - stateful gateways inhibited dynamic routing around failures
- No global addressability
  - ad-hoc, application-specific solutions

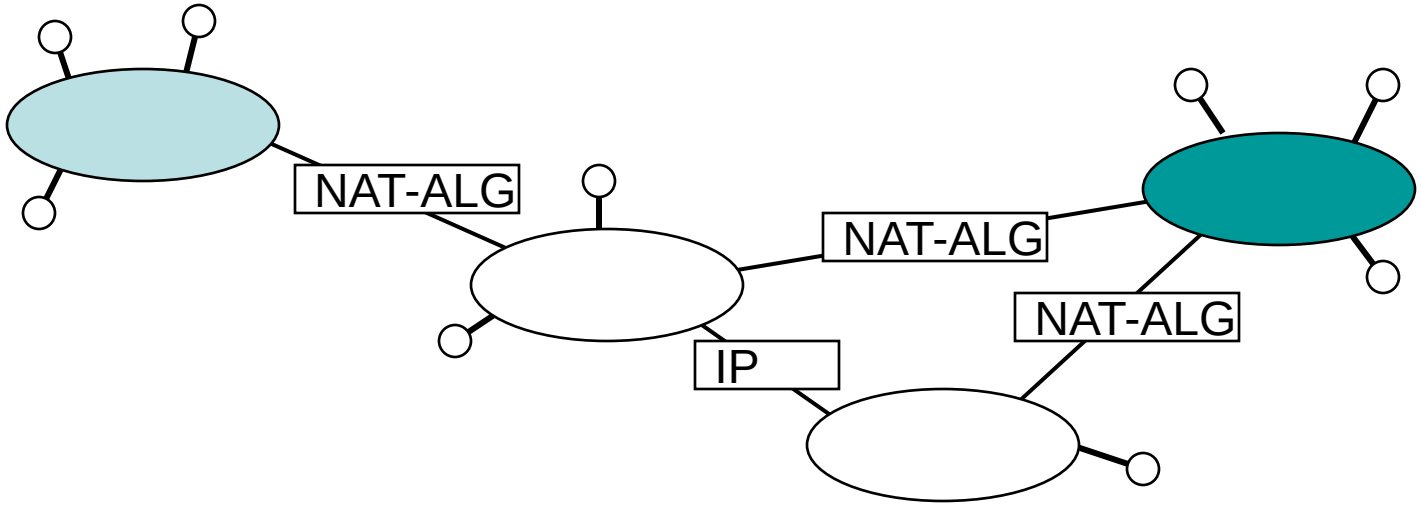
The IP Solution



internet-layer gateways & global addresses

- simple, application-independent, least-common-denominator network service: best-effort datagrams (i.e., packet switching)
- stateless gateways could easily route around failures
- with application-specific knowledge out of the gateways:
  - NSPs no longer had monopoly on providing new services
  - Internet became a platform for rapid, competitive innovation

The Internet Today



network address translators and app-layer gateways

- inevitable loss of some semantics
- difficult to deploy new internet-wide applications
- hard to diagnose and remedy end-to-end problems
- stateful gateways inhibit dynamic routing around failures

no global addressability

- ad-hoc, application-specific (or ignorant!) solutions

But Isn't There Still Lots of IPv4 Address Space Left?

- IPv4 addresses are effectively being rationed

  => consumption statistics tell us nothing about the real demand for addresses, or the hardship created by witholding them

  – the difficulty in obtaining addresses is why many (most?) of the NAT-ALGs exist

- New kinds of Internet devices will be much more numerous, and not adequately handled by NATs (e.g., mobile phones, cars, residential servers, ...)

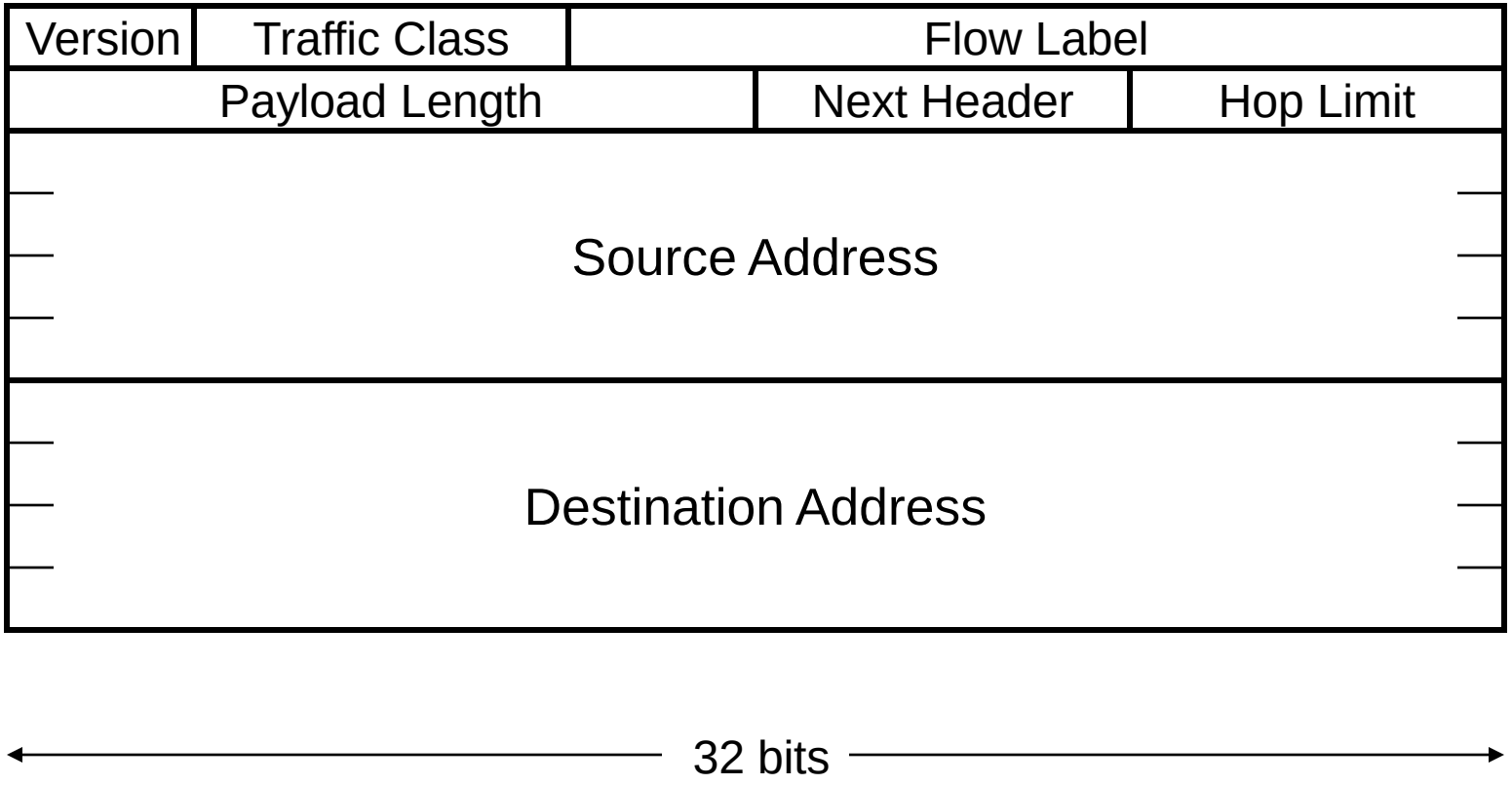
Why Are NATs Not Adequate?

- They won't work for large numbers of "servers", i.e., devices that are "called" by others (e.g., IP phones)
- They break most current IP multicast and IP mobility protocols
- They break many existing applications
- They limit the market for new applications and services
- They compromise the performance, robustness, security, and manageability of the Internet

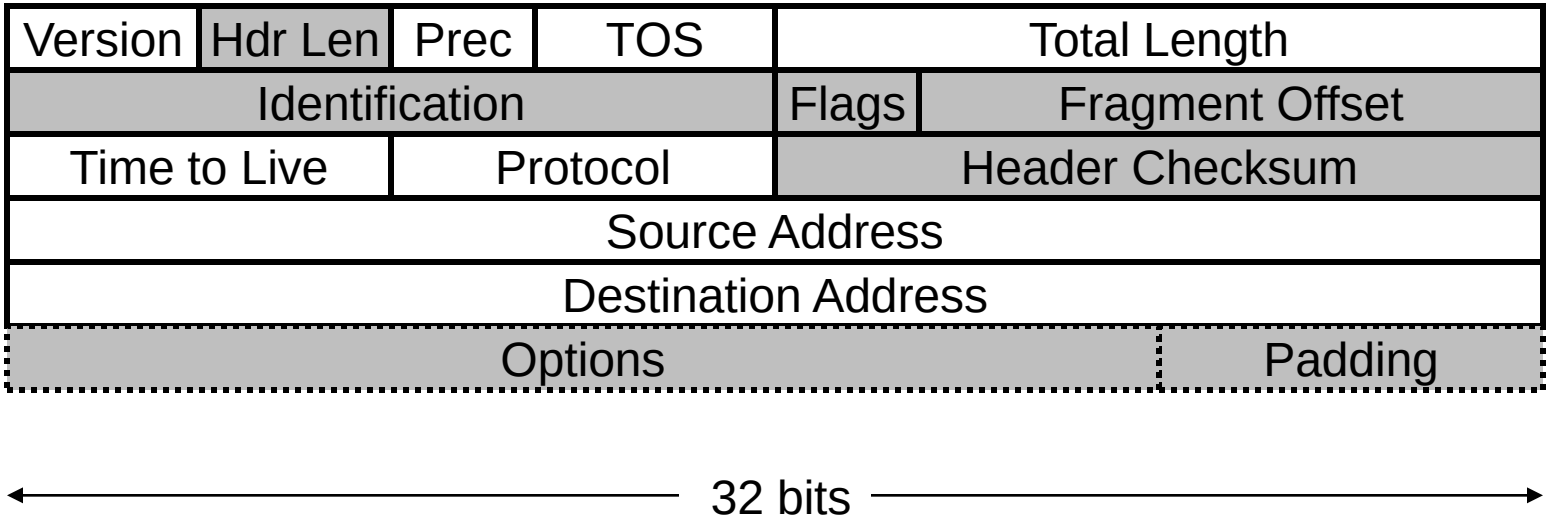
But Can't We Just Make the NATs Better?

- We could keep adding more protocols and features to try to alleviate some of their shortcomings
  - might improve their functionality, but will increase their complexity, fragility, obscurity, unmanagability,...
  - new problems will arise when we start needing inter-ISP NAT
- Doing one thing (moving to IPv6) will avoid the need to continue doing many other things to keep the Internet working and growing
- (no, IPv6 is not the only possible solution, but the most mature, feasible, and widely agreed-upon one)

The IPv6 Header

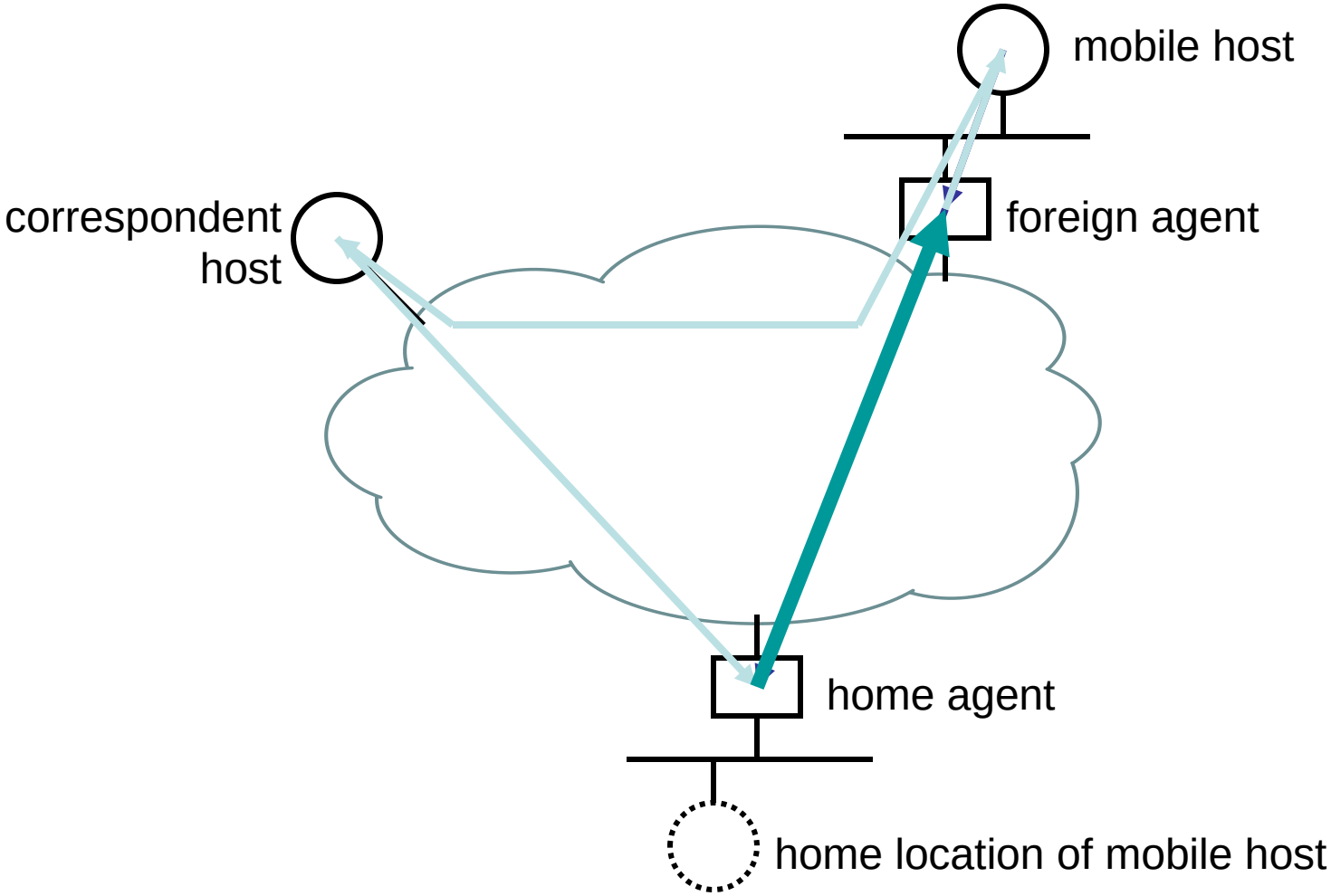


The IPv4 Header

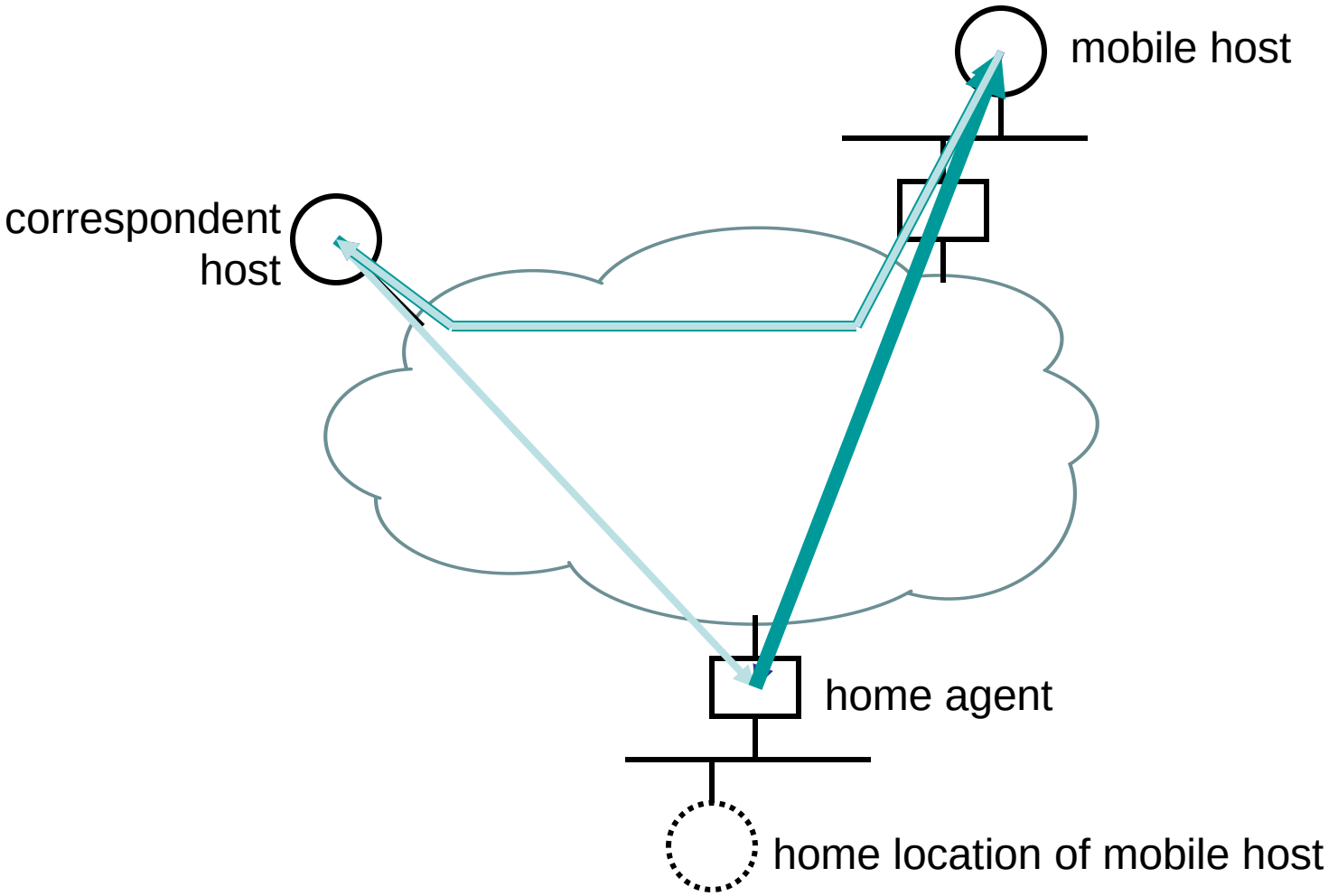


shaded fields are absent from IPv6 header

Mobile IP (v4 version)



Mobile IP (v6 version)



Other Features of IPv6

- Flow label for more efficient flow identification (avoids having to parse the transport-layer port numbers)
- Neighbor unreachability detection protocol for hosts to detect and recover from first-hop router failure
- More general header compression (handles more than just IP+TCP)
- Security (“IPsec”) & differentiated services (“diff-serv”) QoS features — same as IPv4

IPv4-IPv6 Co-Existence / Transition

• A wide range of techniques have been identified and implemented, basically falling into three categories:

(1) dual-stack techniques, to allow IPv4 and IPv6 to co-exist in the same devices and networks

(2) tunneling techniques, to avoid order dependencies when upgrading hosts, routers, or regions

(3) translation techniques, to allow IPv6-only devices to communicate with IPv4-only devices

expect all of these to be used, in combination

Dual-Stack Approach

- When adding IPv6 to a system, do not delete IPv4
  - this multi-protocol approach is familiar and well-understood (e.g., for AppleTalk, IPX, etc.)
  - note: in most cases, IPv6 will be bundled with new OS releases, not an extra-cost add-on
- Applications (or libraries) choose IP version to use
  - when initiating, based on DNS response:

    if (dest has AAAA or A6 record) use IPv6, else use IPv4
  - when responding, based on version of initiating packet
- This allows indefinite co-existence of IPv4 and IPv6, and gradual, app-by-app upgrades to IPv6 usage

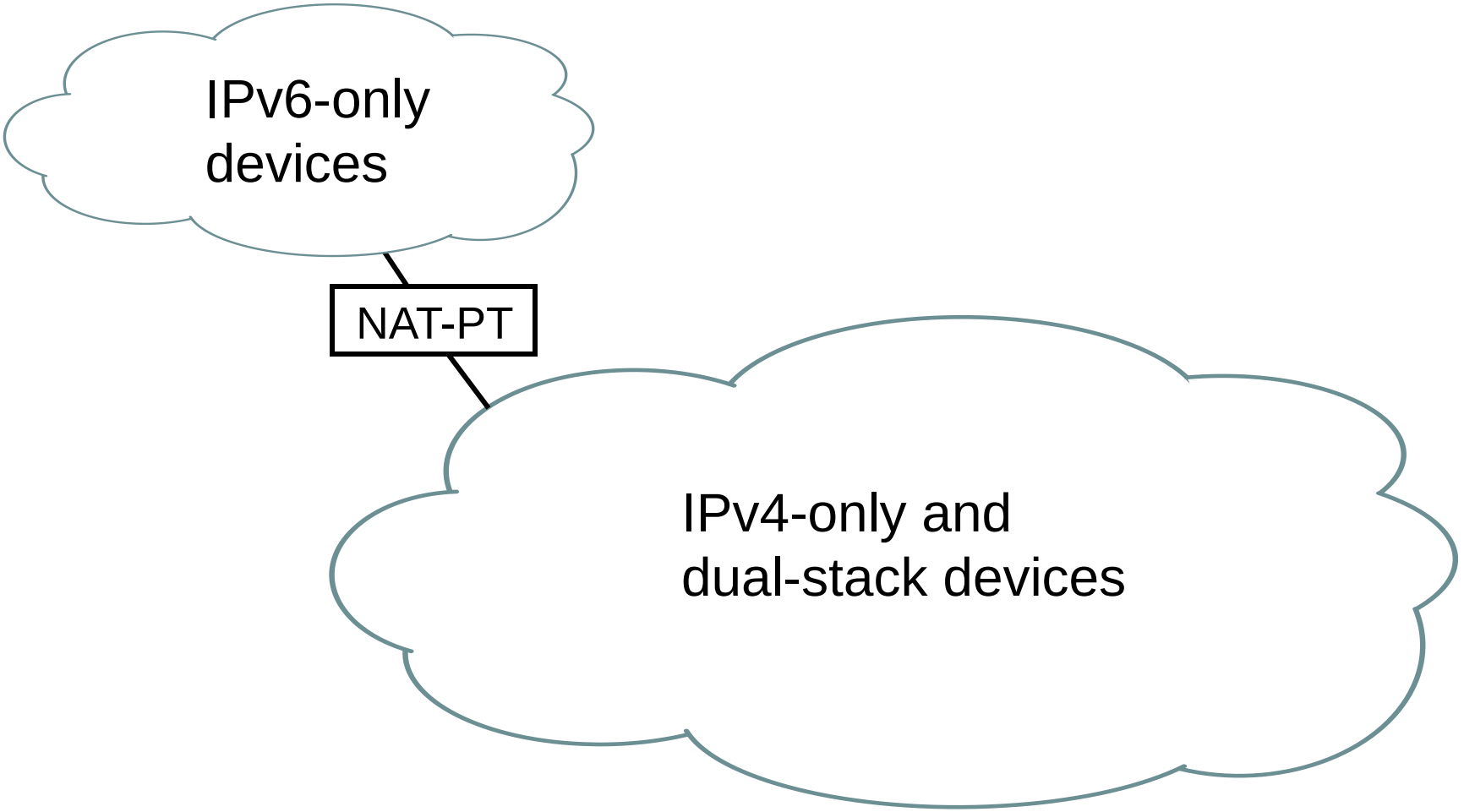
Tunnels to Get Through IPv6-Ignorant Routers / Switches

- Encapsulate IPv6 packets inside IPv4 packets (or MPLS frames)
- Many methods exist for establishing tunnels:
  - manual configuration
  - “tunnel brokers” (using web-based service to create a tunnel)
  - “6-over-4” (intra-domain, using IPv4 multicast as virtual LAN)
  - “6-to-4” (inter-domain, using IPv4 addr as IPv6 site prefix)
- Can view this as:
  - IPv6 using IPv4 as a virtual link-layer, or
  - an IPv6 VPN (virtual public network), over the IPv4 Internet (becoming “less virtual” over time, we hope)

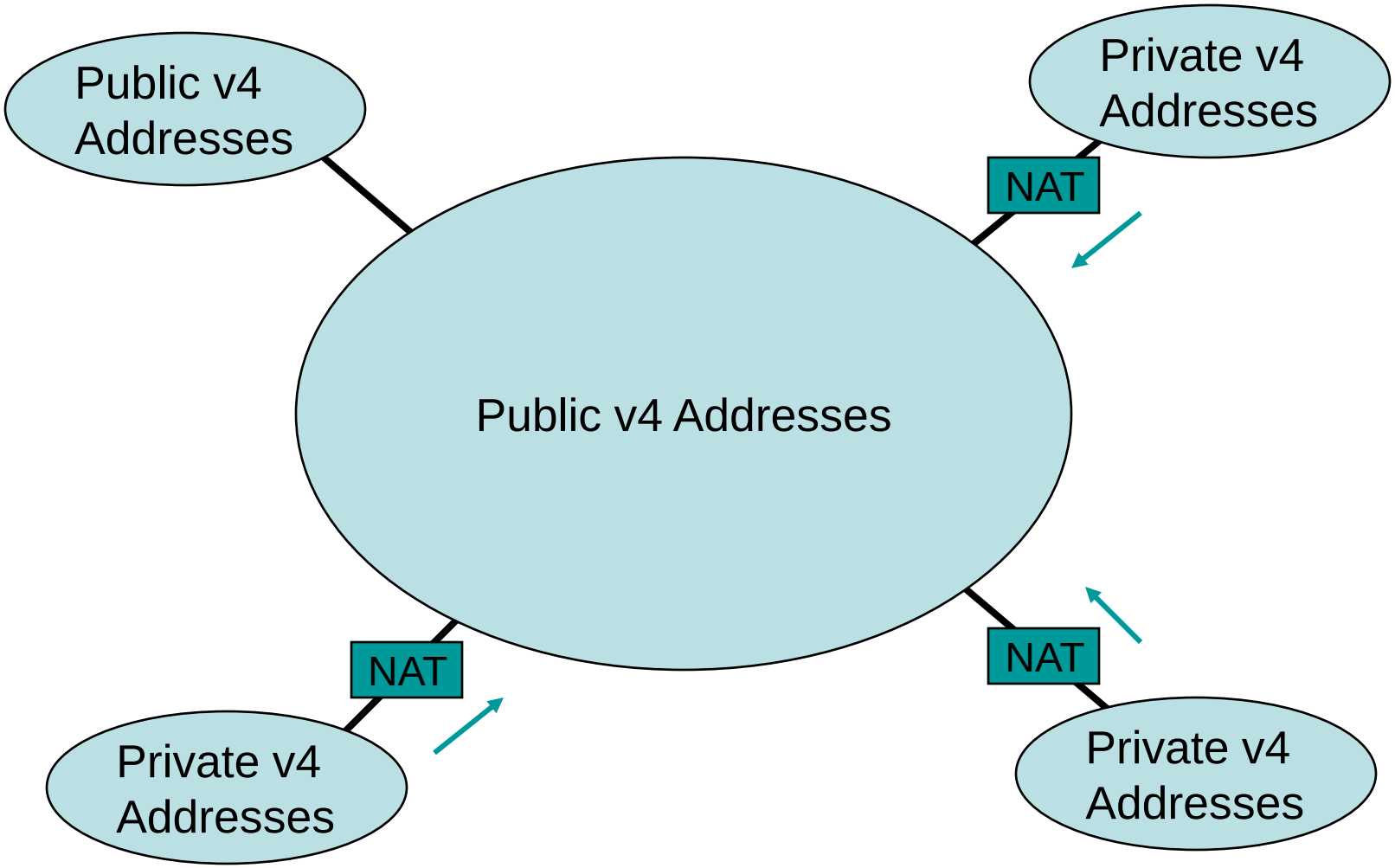
Translation

- May prefer to use IPv6-IPv4 protocol translation for:
  - new kinds of Internet devices (e.g., cell phones, cars, appliances)
  - benefits of shedding IPv4 stack (e.g., serverless autoconfig)
- This is a simple extension to NAT techniques, to translate header format as well as addresses
  - IPv6 nodes behind a translator get full IPv6 functionality when talking to other IPv6 nodes located anywhere
  - they get the normal NAT functionality when talking to IPv4 devices
  - methods used to improve NAT functionality (e.g, ALGs, RSIP) can be used equally to improve IPv6-IPv4 functionality
- Alternative: transport-layer relay or app-layer gateways

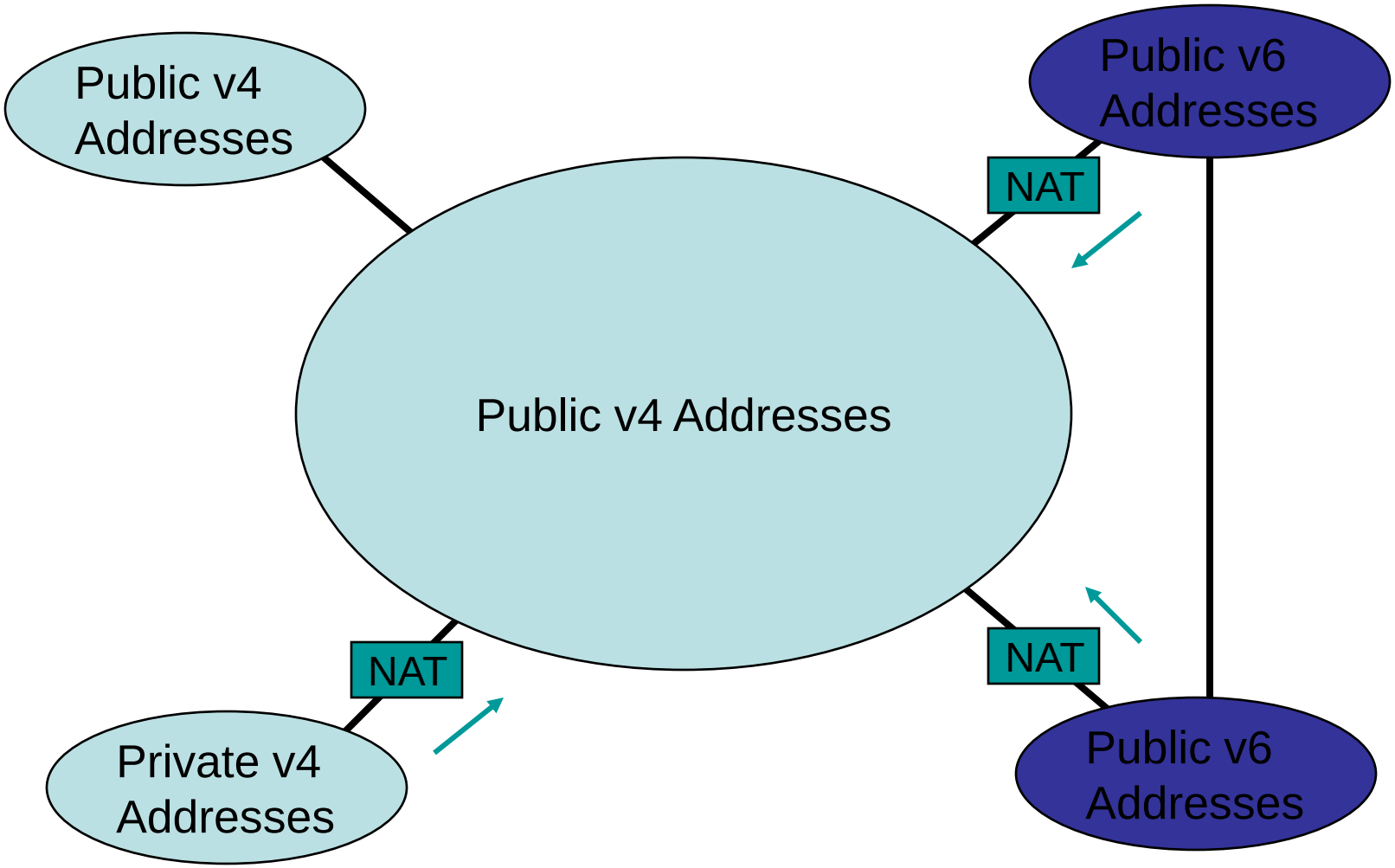
Network Address Translation and Protocol Translation (NAT-PT)



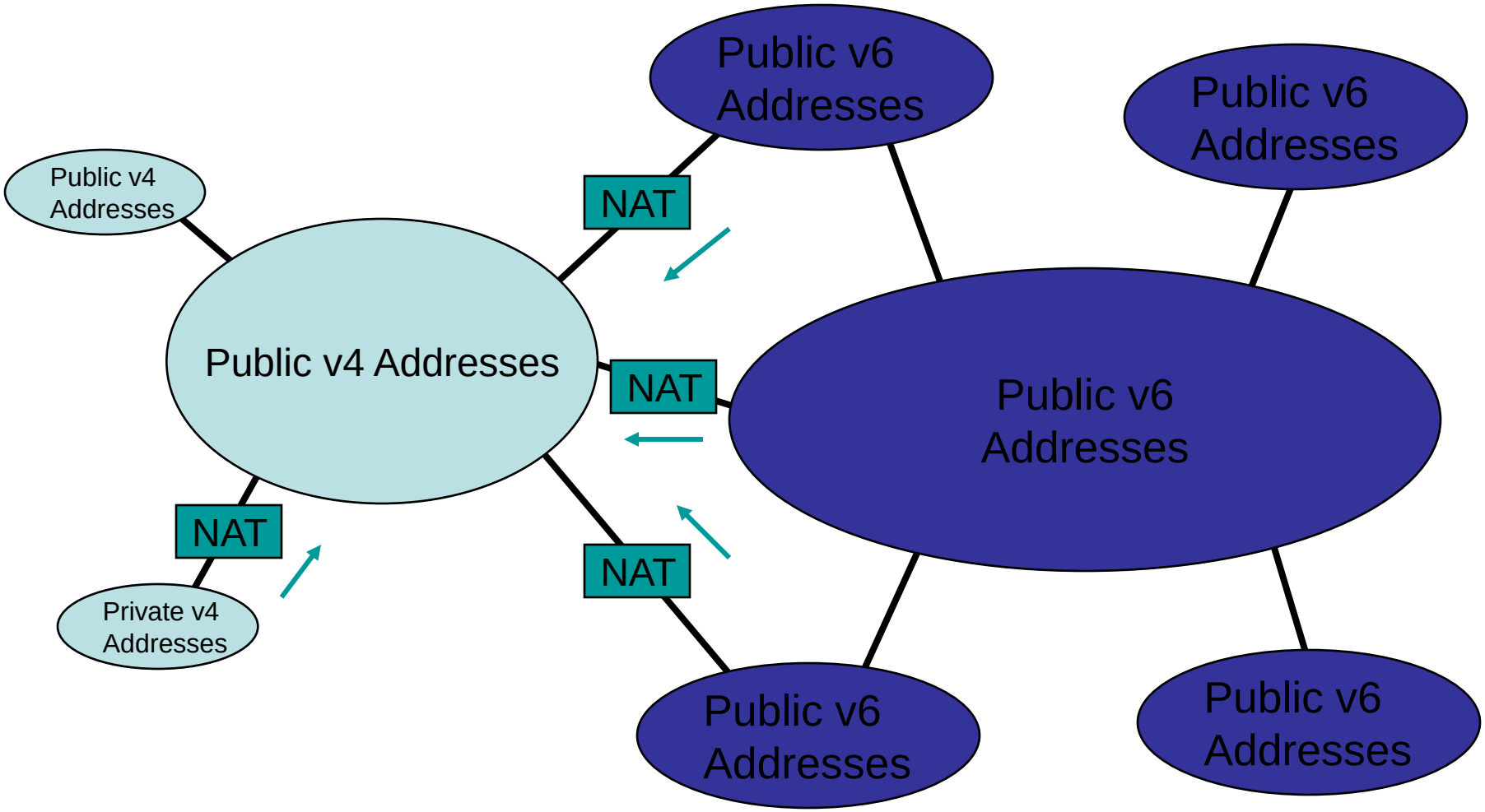
The IPv4 Internet Today



Introducing IPv6 (Simplified View)



Expanding IPv6 (Simplified View)



Exam's quizzes

- **1.** Descrieţi elementele specifice ce trebuie luate în considerare în procesul de proiectare a protocoalelor de comunicaţie.
- **2.** Care sunt caracteristicile protocoalelor de comunicaţie.
- **3.** Descrieţi modul de implementare a protocoalelor de nivel transport în Java. Descrieţi modul de implementare a unei aplicaţii folosind socketi în Java.
- **4.** Care sunt modificările aduse la nivelul reţea o dată cu apariţia IPv6?
- **5.** Care sunt variantele de coexistare IPv4 cu IPv6?