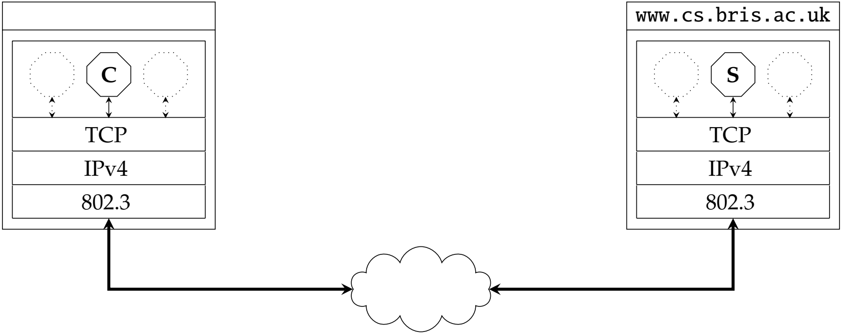
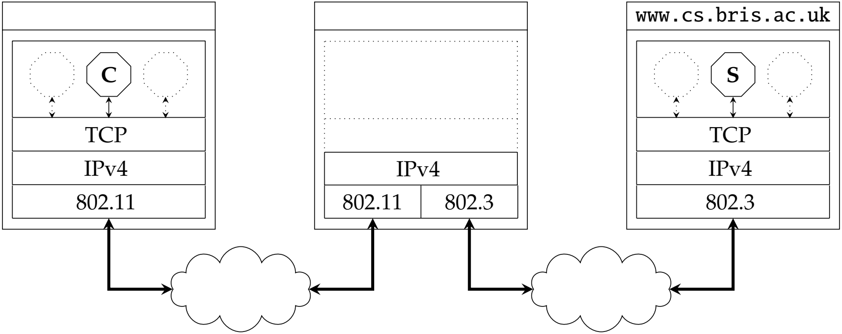


- Goal: investigate the **network layer** (or *inter*-networking more generally) e.g.,
  1. addressing,
  2. forwarding,
  3. routing

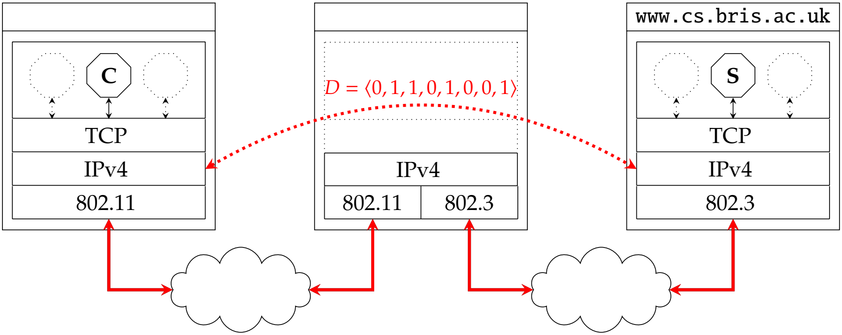
  st. we can transmit (structured) **packets** through an inter-network of multiple connected LANs.



- Goal: investigate the **network layer** (or *inter*-networking more generally) e.g.,
  1. addressing,
  2. forwarding,
  3. routing

  st. we can transmit (structured) **packets** through an inter-network of multiple connected LANs.

COMS20001 lecture: week #21



$F = H_{802.11} \parallel H_{IPv4} \parallel \langle 0,1,1,0,1,0,0,1 \rangle \parallel T_{802.11}$

- Goal: investigate the **network layer** (or *inter*-networking more generally) e.g.,

  1. addressing,
  2. forwarding,
  3. routing

  st. we can transmit (structured) **packets** through an inter-network of multiple connected LANs.

An Aside: "This Jen, is *the Internet*"

http://www.youtube.com/watch?v=iDbyYGrswtg

An Aside: "This Jen, is *the Internet*"

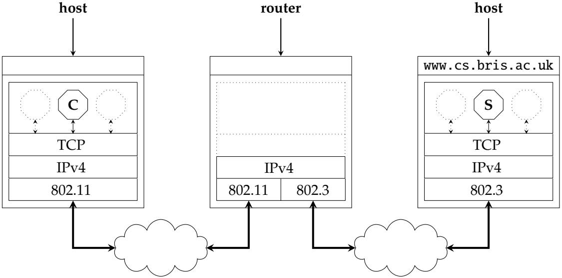
http://www.youtube.com/watch?v=iDbyYGrswtg

Concepts (1)

Definition (**host and router**)

An inter-network is formed from nodes termed

1. **End Systems (ES)** (or **hosts**), and
2. **Intermediate Systems (IS)** (or **routers**).

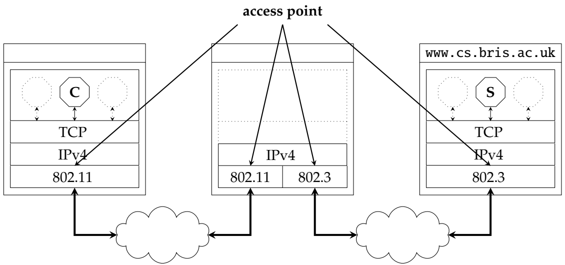


Concepts (1)

Definition (**address**)

Each network layer **access point** is identified by an **address**:

- unlike link layer addresses, network layer addresses need to be *globally* unique,
- one address per host may be insufficient: we need an address per point of access, so per NIC for example.



Concepts (1)

Definition (**routing vs. forwarding**)

Transmission of packets through an inter-network is based on two tasks, namely

1. **routing**, i.e., looking at the destination address in a packet and deciding on the next **hop** (toward said destination), and
2. **forwarding**, i.e., actually transmitting the packet to the next hop.

IPv4 (1)

- **Internet Protocol (IP)** [11] is an inter-networking lingua franca ...
- ... rather than services per se, it deals with *abstraction*: it offers
  - unicast (one-to-one),
  - broadcast (one-to-all),
  - multicast (one-to-some), and
  - anycast (one-to-any)

  packet delivery models, each of which is

  - unreliable,
  - connection-less (i.e., stateless), and
  - “best effort” (i.e., no QoS guarantees).

  and, crucially, allows hererogenaity wrt. lower and higher layers.

IPv4 (2) – Addressing

Definition (**IP address**)

Each access point is assigned a unique identifier called an **IP address**. Note that

- a given address x is simply an n bit integer,
- for IPv4 we have $n = 32$,
- IP addresses are often written in dotted-decimal notation, i.e., 4 decimal integers $0 \leq \hat{x}_i < 256$ each representing 8 bits of the address x.

Definition (**IP prefix**)

IP addresses are hierarchical: a **prefix** l defines a block of addresses, called a **sub-network** (or **sub-net**), by dividing the address into

1. an l-bit **network identifier**, and
2. an $(n - l)$-bit **host identifier**

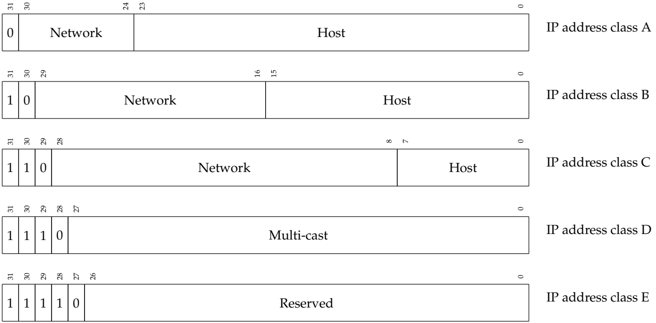
All host identifiers within a block share the same prefix, i.e., have the same network identifier. Note that smaller l means more host identifiers, so is less specific, while larger l means fewer host identifiers, so is more specific.

IPv4 (4) – Addressing

- Question: how do we know the prefix l?

IPv4 (4) – Addressing

- Question: how do we know the prefix l?
- Answer #1: pre-1993, using a class-based hierarchy, i.e.,



IPv4 (4) – Addressing

- Question: how do we know the prefix l?

- Answer #2: post-1993, using **Classless Inter-Domain Routing (CIDR)** [13], st.

$$137.222.103.3/24$$

  explicitly denotes the fact $l = 24$.

IPv4 (4) – Addressing

- Question: how do we obtain an address x (or block thereof)?

IPv4 (4) – Addressing

- Question: how do we obtain an address x (or block thereof)?
- Answer #1: use an assigned **public address block**
  1. network identifiers are assigned by **Internet Assigned Numbers Authority (IANA)**, e.g.,

     $$137.222.0.0/16$$

     was hierarchically assigned via IANA $\rightarrow$ RIPE $\rightarrow$ UoB, and
  2. host identifiers are assigned within the (sub-)network, e.g.,

     $$137.222.103.3$$

     was statically assigned to `snowy.cs.bris.ac.uk`

  which *is* globally unique.

IPv4 (4) – Addressing

- Question: how do we obtain an address x (or block thereof)?

- Answer #2: use *any* **private address block** [9] e.g.,

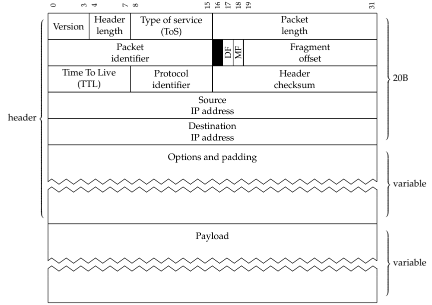
$$192.168.0.0/16$$

  and *any* address in it, e.g.,

$$192.168.1.123$$

  which *isn't* globally unique!

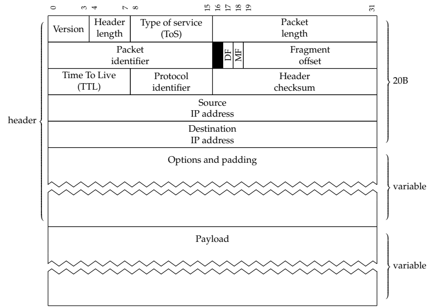
Data Structure (IPv4 packet [11, Section 3.1])



The data structure includes:

- A packet version number (allowing protocol evolution).
- A header length (measured in 32-bit words), formally termed **Internet Header Length (IHL)**.
- A packet length (measured in 8-bit octets), which includes the header.
- A packet identifier.

Data Structure (IPv4 packet [11, Section 3.1])



The data structure includes:

- A set of service options, originally defined as
  - 2-bit reserved,
  - 3-bit priority level,
  - 1-bit reliability (normal or high),
  - 1-bit latency (normal or low),
  - 1-bit throughput (normal or high)

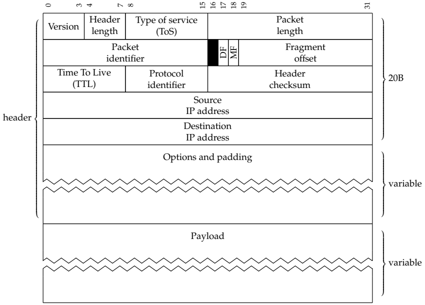
  but now depreciated in favour of

  1. **Differentiated Services (DS)** [10] and
  2. **Explicit Congestion Notification (ECN)** [12]

  fields.

IPv4 (6) – Packets

Data Structure (IPv4 packet [11, Section 3.1])

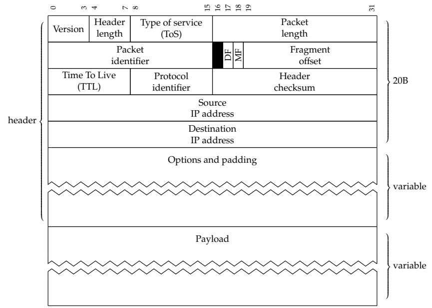


The data structure includes:

- A set of flags, including
  - 1-bit **More Fragments (MF)** flag, which marks final or non-final fragment(s),
  - 1-bit **Don't Fragment (DF)** flag, which prohibits fragmentation by the IP layer.
- A fragment offset (measured in 64-bit words), specifying where this packet payload stems from in the original, unfragmented packet.

IPv4 (6) – Packets

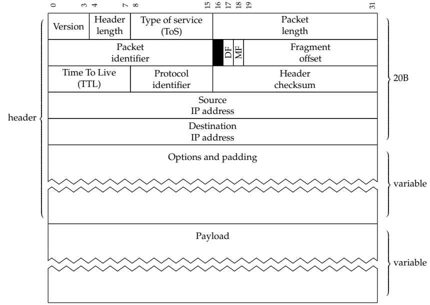
Data Structure (IPv4 packet [11, Section 3.1])



The data structure includes:

- A **Time To Live (TTL)** field specifying the point at which the packet is discarded (if not yet delivered to the destination).
- A 16-bit checksum (on header only) used to detect errors.

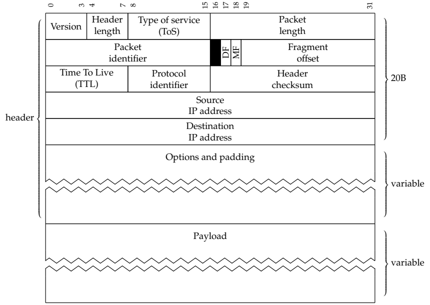
Data Structure (**IPv4 packet** [11, Section 3.1])



The data structure includes:

- A 32-bit source IP address.
- A 32-bit destination IP address.
- The protocol identifier, specifying which higher layer will receive the packet once it reaches the destination.

Data Structure (IPv4 packet [11, Section 3.1])



The data structure includes:

- A set of options (allowing protocol extensibility).
- Any padding required to ensure the header is a multiple of 32 bits.
- The payload.

IPv4 (10) – Forwarding

- Goal: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.

IPv4 (10) – Forwarding

- Goal: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.
- Observation:
  - all hosts on the same (sub-)network share a prefix, so
  - maintain a **forwarding table** that maps prefixes to next hop

  st. the table remains compact iff. sensible prefixing is used.

IPv4 (10) – Forwarding

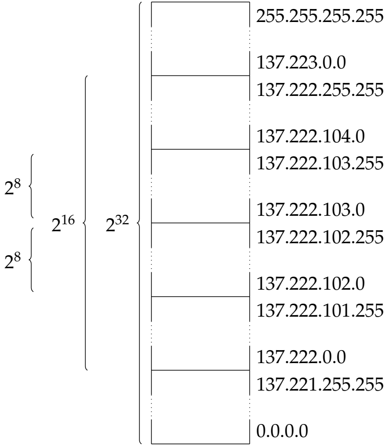
- Goal: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.
- Problem: prefixes in the table might overlap.
- Solution: select the *longest* matching prefix that applies; this allows
  - special-case behaviours (via more-specific prefix, e.g., 137.222.103.3/32), and
  - default behaviours (via less-specific prefix, e.g., 0.0.0.0/0).

IPv4 (10) – Forwarding

- Goal: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.
- Observation: we only need to know what the next hop is, so
  - have the routers make (global) routing decisions, and
  - have the hosts follow a simple rule:
    1. communicate locally with destination if it is on the same sub-network, otherwise
    2. forward to nearest router as next hop toward destination

i.e., improve scalability by leveraging hierarchy within topology and addressing.

IPv4 (11) – Forwarding



- Consider a router with 3 NICs, and the forwarding table

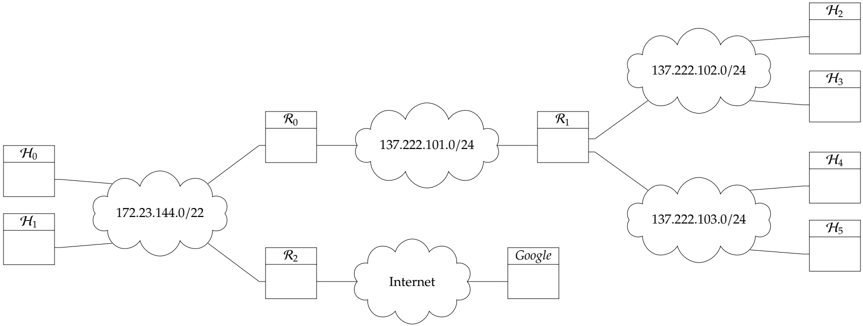
Prefix	Next hop
137.222.102.0/24	...
137.222.0.0/16	...
0.0.0.0/0	...

- A given address 137.222.102.13
  - is within the prefixes in *all* rows, *but*
  - the prefix in the first row is longer, so this is the selected match.

- Example: forward a packet

$$P = H_{IPv4}[\text{src} = \mathcal{H}_0, \text{dst} = \mathcal{H}_2] \parallel D$$

through some (purely hypothetical) inter-network.

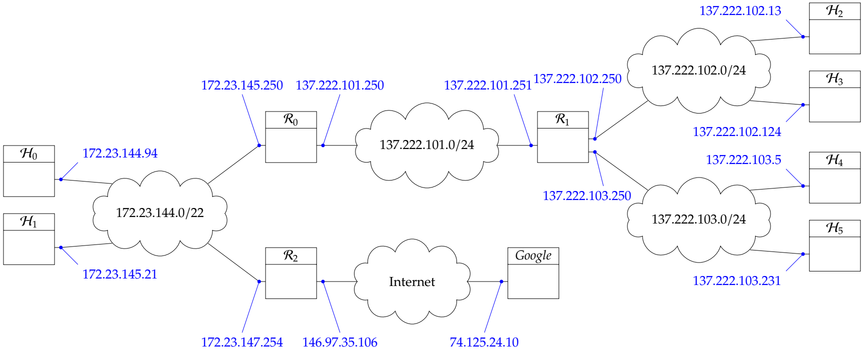


IPv4 (12) – Forwarding

- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13] \;\|\; D$$

through some (purely hypothetical) inter-network.

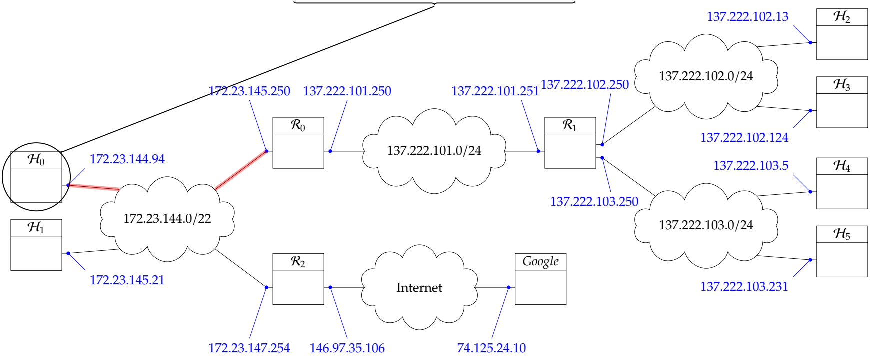


- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13, \text{TTL} = 64] \parallel D$$

through some (purely hypothetical) inter-network.

Address	Next hop
172.23.144.0/22	NIC via broadcast
0.0.0.0/0	NIC via 172.23.145.250



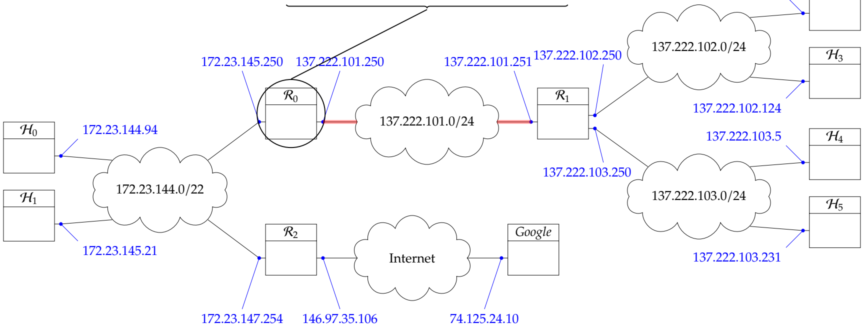
IPv4 (12) – Forwarding

- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13, \text{TTL} = 63] \parallel D$$

through some (purely hypothetical) inter-network.

Address	Next hop
137.222.101.0/24	NIC #1 via broadcast
137.222.102.0/23	NIC #1 via 137.222.101.251
172.23.144.0/22	NIC #2 via broadcast
0.0.0.0/0	NIC #2 via 172.23.147.254

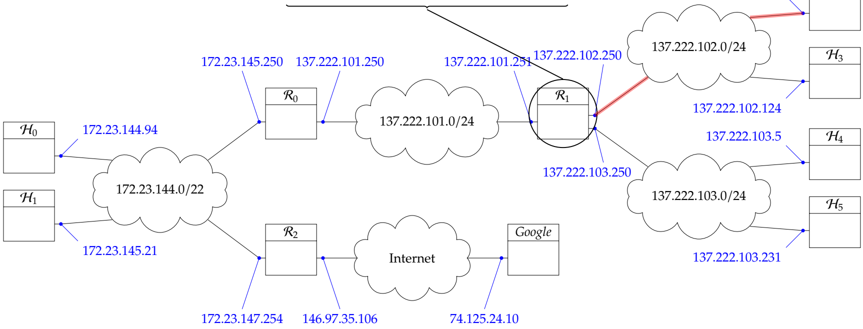


- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13, \text{TTL} = 62] \;\|\; D$$

through some (purely hypothetical) inter-network.

Address	Next hop
137.222.103.0/24	NIC #3 via broadcast
137.222.102.0/24	NIC #2 via broadcast
137.222.101.0/24	NIC #1 via broadcast
0.0.0.0/0	NIC #1 via 137.222.101.250



- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

IPv4 (13) – Fragmentation

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?
- "Solution" #1: produce an error, and drop the packet!

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

- Solution #2:
  1. allow $\mathcal{H}_i$ to transmit packets of any length,
  2. have each router **fragment** any outgoing packet that is larger than the associated MTU,
  3. reassemble packet fragments at destination.

IPv4 (13) – Fragmentation

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

- Solution #3:
  1. force $\mathcal{H}_i$ to discover the **path MTU** between $\mathcal{H}_i$ and $\mathcal{H}_j$, then
  2. limit the size of packets transmitted by $\mathcal{H}_i$ so fragmentation is avoided.

IPv4 (13) – Fragmentation

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

noting that IPv4 hosts and routers

- must support reassembly [8, Section 3.3.2], and
- may support fragmentation [8, Section 3.3.3]

but modern implementations typically use path MTU discovery.

IPv4 (14) – Fragmentation

Algorithm (**fragment**)

At the source or an intermediate router, imagine there is a need to fragment some packet P:

1. If the DF flag in P is **true**, drop P.
2. Otherwise divide the payload into n fragments, F_i for $0 \leq i < n$, each of whose length (including header) is less than the MTU.
3. Copy the header from P into each F_i, and update both the offset and MF flags based on i.
4. Transmit each F_i.

Algorithm (**reassemble**)

At the destination *only*:

1. Buffer fragments with same identifier until they're all received, noting
   - they may be received out-of-order,
   - a reassembly time-out prevents indefinite buffering, and
   - a fragment F_i whose MF flag is **false** is the last one, so yields the total length.
2. Reconstruct the original packet P (at least the payload).
3. Process P as per normal, i.e., provide it to whatever transport layer protocol P indicates.

Continued in next lecture ...

References

[1] Wikipedia: Internet protocol (IP).
http://en.wikipedia.org/wiki/Internet_Protocol.

[2] Wikipedia: IP address.
http://en.wikipedia.org/wiki/IP_address.

[3] Wikipedia: IP fragmentation.
http://en.wikipedia.org/wiki/IP_fragmentation.

[4] Wikipedia: Network layer.
http://en.wikipedia.org/wiki/Network_layer.

[5] Wikipedia: Packet forwarding.
http://en.wikipedia.org/wiki/Packet_forwarding.

[6] Wikipedia: Routing.
http://en.wikipedia.org/wiki/Routing.

[7] Wikipedia: Routing table.
http://en.wikipedia.org/wiki/Routing_table.

[8] R. Braden.
Requirements for Internet hosts – communication layers.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1122, 1989.
http://tools.ietf.org/html/rfc1122.

References

[9] IANA.
Special-use IPv4 addresses.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 3330, 2002.
http://tools.ietf.org/html/rfc3330.

[10] K. Nichols, S. Blake, F. Baker, and D. Black.
Definition of the Differentiated Services field (or DS field) in the IPv4 and IPv6 headers.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 2474, 1998.
http://tools.ietf.org/html/rfc2474.

[11] J. Postel.
Internet Protocol.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 791, 1981.
http://tools.ietf.org/html/rfc791.

[12] K. Ramakrishnan, S. Floyd, and D. Black.
The addition of Explicit Congestion Notification (ECN) to IP.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 3168, 2001.
http://tools.ietf.org/html/rfc3168.

[13] Y. Rekhter and T. Li.
An architecture for IP address alloation with CIDR.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1518, 1989.
http://tools.ietf.org/html/rfc1518.

References

[14] W. Stallings.
Chapter 19: Internet protocols.
In *Data and Computer Communications* [16].

[15] W. Stallings.
Chapter 20: Internetwork operation.
In *Data and Computer Communications* [16].

[16] W. Stallings.
Data and Computer Communications.
Pearson, 9th edition, 2010.