Reflection

• The events in last few days have left me sad!

• Such events must be condemned
  • They have no place in our society

• Please do not lose faith
  • If you are feeling overwhelmed
    • Please express your thoughts
    • Talk to friends, family, me, professionals

• My office door is always open
Announcements

• Please fill out the feedback
  • I emailed a link

• If you have a conflict with prelims/finals
  • Let me know by 03/15
  • I’ll do everything I can to accommodate!
Quiz 1 distribution

Quiz 2 Grades

Grade (out of 20)

<table>
<thead>
<tr>
<th>Grade (out of 20)</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>0</td>
</tr>
<tr>
<td>4-7</td>
<td>0</td>
</tr>
<tr>
<td>8-11</td>
<td>0</td>
</tr>
<tr>
<td>12-15</td>
<td>0</td>
</tr>
<tr>
<td>16-19</td>
<td>25</td>
</tr>
<tr>
<td>20-20</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>17.875</td>
</tr>
<tr>
<td>Median</td>
<td>19</td>
</tr>
<tr>
<td>Std. deviation</td>
<td>3.344</td>
</tr>
</tbody>
</table>
Goals for Today’s Lecture

• Acknowledge:
  • We have studied a lot of protocols in last few lectures
  • If you do not understand how they fit together
    • Don’t worry; you’ll soon!
    • In couple of lectures, we’ll come back to bigger picture

• The Internet Protocol
Network Layer

• THE functionality: *delivering the data*

• **THE protocol: Internet Protocol (IP)**
  • To achieve its functionality, IP protocol has **three** responsibilities
    • **Addressing hosts**
    • **Forwarding packets (actually datagrams)**
    • **Routing** (link-state, distance vector; several more lectures)
Routing versus Forwarding

- **Routing:** “control plane”
  - Computing paths the packets will traverse
  - Distributed protocol leads to state at each router
  - Can be done slowly (tens of milliseconds)
  - But must scale to size of network!

- **Forwarding:** “data plane”
  - Directing a packet toward the destination
  - Individual switch/router use their routing tables
  - Must be done quickly (nano seconds)

- Different goals, different constraints, different mechanisms
  - So far we have learnt about the first one
Internet Protocol

• THE functionality: delivering the data

• THE protocol: Internet Protocol (IP)
  • To achieve its functionality, IP protocol has three responsibilities

• Unifying protocol
What is “designing” a protocol?

• Specifying the syntax of its messages
  • Format

• Specifying their semantics
  • Meaning
  • Responses
What is Designing IP?

• Syntax: format of packet
  • Nontrivial part: packet “header”
  • Rest is opaque payload (why opaque?)

• Semantics: meaning of header fields
  • Required processing
Packet Header as Interface

• Think of packet header as interface
  • Only way of passing information from packet to switch

• Designing interfaces:
  • What task are you trying to perform?
  • What information do you need to accomplish it?

• Header reflects information needed for basic tasks
What Tasks Do We Need to Do?

- Read packet correctly
- Get the packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with the packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path
Reading Packet Correctly

• Where does the header end?

• Where the the packet end?

• What version of IP?
  • Why is this so important?
Getting to the Destination

• Provide destination address

• Should this be location or identifier (name)?
  • And what’s the difference?

• If a host moves should its address change?
  • If not, how can you build scalable Internet?
  • If so, then what good is an address for identification?
Getting Response Back to Source

• Source address

• Necessary for routers to respond to source
  • When would they need to respond back?
    • Failures!
  • Do they really need to respond back?
    • How would the source know if the packet has reached the destination?
Carry Data

- Payload!
Questions?
List of Tasks

• Read packet correctly
• Get the packet to the destination
• Get responses to the packet back to source
• Carry data
• Tell host what to do with packet once arrived
• Specify any special network handling of the packet
• Deal with problems that arise along the path
Telling Destination How to Process Packet

• Indicate which protocols should handle packet
• What layers should this protocol be in?
• What are some options for this today?
• How does the source know what to enter here?
Special Handling

• Type of service, priority, etc.

• Options: discuss later
Dealing With Problems

• Is packet caught in loop?
  • TTL

• Header corrupted:
  • Detect with Checksum
  • What about payload checksum?

• Packet too large?
  • Deal with fragmentation
  • Split packet apart
  • Keep track of how to put together
Are We Missing Anything?

• Read packet correctly
• Get the packet to the destination
• Get responses to the packet back to source
• Carry data
• Tell host what to do with packet once arrived
• Specify any special network handling of the packet
• Deal with problems that arise along the path
From Semantics to Syntax

- The past few slides discussed the kinds of information the header must provide
- Will now show the syntax (layout) of IPv4 header, and discuss the semantics in more detail
IP Packet Structure

- 4-bit Version
- 4-bit Header Length
- 8-bit Type of Service (TOS)
- 16-bit Total Length (Bytes)
- 16-bit Identification
- 3-bit Flags
- 13-bit Fragment Offset
- 8-bit Time to Live (TTL)
- 8-bit Protocol
- 16-bit Header Checksum
- 32-bit Source IP Address
- 32-bit Destination IP Address
- Options (if any)
- Payload
20 Bytes of Standard Header, then Options

- 4-bit Version
- 4-bit Header Length
- 8-bit Type of Service (TOS)
- 16-bit Total Length (Bytes)
- 16-bit Identification
- 3-bit Flags
- 13-bit Fragment Offset
- 8-bit Time to Live (TTL)
- 8-bit Protocol
- 16-bit Header Checksum
- 32-bit Source IP Address
- 32-bit Destination IP Address
- Options (if any)
- Payload
Next Set of Slides

• Mapping between tasks and header fields
• Each of these fields is devoted to a task
• Let’s find out which ones and why...
Go Through Tasks One-by-One

• Read packet correctly
• Get the packet to the destination
• Get responses to the packet back to source
• Carry data
• Tell host what to do with packet once arrived
• Specify any special network handling of the packet
• Deal with problems that arise along the path
Read Packet Correctly

• **Version number** (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically “4” (for IPv4), and sometimes “6” (for IPv6)

• **Header length** (4 bits)
  - Number of 32-bit words in the header
  - Typically “5” (for a 20-byte IPv4 header)
  - Can be more when IP options are used

• **Total length** (16 bits)
  - Number of bytes in the packet
  - Maximum size is 65,535 bytes (2^16 -1)
  - ... though underlying links may impose smaller limits
Fields for Reading Packet Correctly

- 4-bit Version
- 4-bit Header Length
- 8-bit Type of Service (TOS)
- 16-bit Total Length (Bytes)
- 16-bit Identification
- 3-bit Flags
- 13-bit Fragment Offset
- 8-bit Time to Live (TTL)
- 8-bit Protocol
- 16-bit Header Checksum
- 32-bit Source IP Address
- 32-bit Destination IP Address
- Options (if any)
- Payload
Getting Packet to Destination and Back

• Two IP addresses
  • Source IP address (32 bits)
  • Destination IP address (32 bits)

• Destination Address
  • Unique locator for the receiving host
  • Allows each node to make forwarding decisions

• Source Address
  • Unique locator for the sending host
  • Recipient can decide whether to accept packet
  • Enables recipient to send a reply back to the source
### Fields for Reading Packet Correctly

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td></td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td></td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td></td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td></td>
</tr>
<tr>
<td>16-bit Identification</td>
<td></td>
</tr>
<tr>
<td>3-bit Flags</td>
<td></td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td></td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td></td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td></td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td></td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td></td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>

The diagram illustrates the structure of an IP packet, highlighting the key fields for reading packet correctly.
Questions?
List of Tasks

• Read packet correctly
• Get the packet to the destination
• Get responses to the packet back to source
• Carry data
• Tell host what to do with packet once arrived
• Specify any special network handling of the packet
• Deal with problems that arise along the path
Telling Host How to Handle Packet

- **Protocol (8 bits)**
  - Identifies the higher level protocol
  - Important for demultiplexing at receiving host

- **Most common examples**
  - E.g., “6” for the Transmission Control Protocol (TCP)
  - E.g., “17” for the User Datagram Protocol

![Diagram showing IP and TCP headers with Protocol values 6 and 17]
## Fields for Reading Packet Correctly

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>4-bit Header Length</td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td>16-bit Total Length (Bytes)</td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>3-bit Flags</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>13-bit Fragment Offset</td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td>16-bit Header Checksum</td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td></td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>
Special Handling

• **Type-of-Service (8-bits)**
  • Allow packets to be treated differently based on needs
  • E.g., low delay for audio, high bandwidth for bulk transfer
  • Has been redefined several times, no general use

• **Options**
  • Ability to specify other functionality
  • Extensible format (later)
### Fields for Reading Packet Correctly

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>Version of the Internet Protocol Protocol</td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td>Length of the header</td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td>Type of Service (TOS)</td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td>Total length in bytes</td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>Identification</td>
</tr>
<tr>
<td>3-bit Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>Time to Live (TTL)</td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td>Protocol</td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td>Header checksum</td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td>Source IP Address</td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td>Destination IP Address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>Payload</td>
<td>Payload</td>
</tr>
</tbody>
</table>
# Option Field Layout

<table>
<thead>
<tr>
<th>Field</th>
<th>Size (bits)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copied</td>
<td>1</td>
<td>Set if field copied to all fragments</td>
</tr>
<tr>
<td>Class</td>
<td>2</td>
<td>0 = control, 2 = debugging/measurement</td>
</tr>
<tr>
<td>Number</td>
<td>5</td>
<td>Specified option</td>
</tr>
<tr>
<td>Length</td>
<td>8</td>
<td>Size of entire option</td>
</tr>
<tr>
<td>Data</td>
<td>Variable</td>
<td>Option-specific data</td>
</tr>
</tbody>
</table>
Examples of Options

- Record Route
- Strict Source Route
- Loose Source Route
- Timestamp
- Traceroute
- Router Alert
- ...

Potential Problems

- Header Corrupted: Checksum
- Loop: TTL
- Packet too large: Fragmentation
Preventing Loops

- Forwarding loops cause packets to cycle forever
  - As these accumulate, eventually consume all capacity

- Time-to-live (TTL) Field (8-bits)
  - Decremented at each hop, packet discarded if reaches 0
  - ... and “time exceeded” message is sent to the source
    - Using “ICMP” control message; basis for traceroute
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>Version of the IP protocol</td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td>Length of the IP header</td>
</tr>
<tr>
<td>8-bit Type of Service</td>
<td>Type of Service (TOS)</td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td></td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>Identification of the packet</td>
</tr>
<tr>
<td>3-bit Flags</td>
<td>Flags of the packet</td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td>Fragment offset of the packet</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>Time to Live</td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td>Protocol of the packet</td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td>Header checksum of the packet</td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td>Source IP address</td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Options if any</td>
</tr>
<tr>
<td>Payload</td>
<td>Payload</td>
</tr>
</tbody>
</table>
Header Corruption

- Checksum (16 bits)
  - Particular form of checksum over packet header

- If not correct, router discards packets
  - So it doesn’t act in bogus information

- Checksum recalculated at every router
  - Why?
  - Why include TTL?
  - Why only header?
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>Number of IP version</td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td>Length of IP header</td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td>Type of service options</td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td>Total length of IP header (in bytes)</td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>Identification number</td>
</tr>
<tr>
<td>3-bit Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>Time to live (in hops)</td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td>Protocol number</td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td>Header checksum</td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td>Source IP address</td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Options if any</td>
</tr>
<tr>
<td>Payload</td>
<td>Payload</td>
</tr>
</tbody>
</table>
Addressing
Addressing so far

• Each node has a “name”
  • We have so far worked only with names
  • Assumed that forwarding/routing etc. done on names

• Today:
  • Why do we need addresses?
  • Why do we assign addresses the way we assign addresses?
  • And why thats the wrong way!
Current Addressing

• Reflects series of necessary hacks
  • Necessary to survive, but not pretty

• No one would design such a system from scratch

• Let me walk you through the same that is today’s addressing
Why addresses?

• Used by routers to forward packets to destination
  • A “Locator”, so to say …

• Sometimes also used as identifiers
  • Must let destination know packet was for them
    • Destinations know what their address is

• Typically, addresses are mostly locators
  • Contains information about how to reach the destination host
Two requirements for addressing

• Scalable routing
  • How much state must be exchanged to create paths?
  • How must state must be stored to forward packets?
  • How much state needs to be updated upon host arrival/departure?

• Efficient forwarding
  • How quickly can one locate items in routing table?
  • Some addressing schemes make this easier than others

• Host must be able to recognize packet is for them
Recognizing packets are for the destination

- Address can contain something intrinsic to the destination

- I know my current address
  - Can recognize that incoming packet is using that address

- Different layers handle this differently
  - L2 addresses are intrinsic
  - L3 addresses are assigned (and ephemeral)
Layer 2: “Flat” Addressing

• Typically uses MAC address
  • “Names”, remember? Used as identifier

• Unique identifiers hardcoded in the hardware
  • No location information

• Local area networks route on these “flat” addresses
  • Each switch stores a separate routing entry **for each host**

• Works nicely with Spanning Tree Protocol
  • Routes set up “on demand”
How does this meet our requirements?

• Scalable routing
  • Can scale to size of L2 networks

• Efficient forwarding
  • Exact match lookup on MAC addresses
  • Only need table for “active” hosts

• Host must be able to recognize the packet is for them
  • MAC address does this perfectly
  • Conclusion: scaling limited by table size (#hosts)
How would you scale L2

• Suppose we want to design a much larger L2 network

• Must use MAC address as part of the address
  • Only way host knows that the packet is for them

• But how do you avoid having separate routing entry for each host?
How would you scale L2

• Lets try to design an addressing scheme to achieve our requirements:
  
  • Scalable routing on large networks
    • Small routing tables (less than #hosts)
    • Small number of updates upon each host arrival/departure
  
  • Efficient forwarding
    • Fast to look up from the table
  
  • Destination must be able to identify the packet is for it
Solution

- Addresses are of form — Switch:Host

- All internal forwarding done on switch addresses
  - Fewer in number (than hosts)
  - Very stable

- Mapping between hosts and switches
  - A mechanism we’ll study soon ...
  - Information only kept at the end-hosts

- Hosts know that packet is for them
  - Using MAC addresses
  - Don’t need to know which switch they are at
How do we extend this to the entire Internet?

• Routing tables cannot have entry for each switch in the Internet

• Cannot flood when you don’t know where someone is
One solution

• Use addresses of the form — Network:Host

• Routers know how to reach all networks in the world
  • Routers ignore host part of the address
  • Hosts can recognize when packets are from them (host)

• Each network knows how to reach local hosts
  • E.g., using L2

• A lookup mechanism allows hosts to know where every host is
  • That is, which network to send to

• This was the original IP addressing scheme
What do I mean by “network”

• In the original IP addressing scheme ...
  • Network meant an L2 network
  • Often referred to as a “subnet”
  • There are too many of them now to scale
Two key aspects of the solution

• Aggregation

• Mapping between identifier and locator
Aggregation

• Aggregation: single forwarding entry used for many individual hosts

• Example:
  • In our scalable L2 solution: aggregate was switch
  • L3: aggregate was network

• Advantages:
  • Fewer entries and more stable
  • Change of hosts do not change tables
    • Don’t need to keep state on individual hosts
Name/Identifier to Location Mapping

• Uses Domain Name System
  • Remember?
  • We are going to discuss this in a few lectures ....

• Use “name” as an identifier

• Returns the IP address as a locator
Where are we?

- Have a sensible addressing scheme for scaling L2
  - Use MAC addresses for host addressing
  - But forward based on destination switch addresses

- We have a sensible addressing scheme for L3
  - Use Network:Host addressing
  - Routers forward on network fields
How do these fit together?

- When sending a packet from A to B ...
  
  - A sends over L2 network to “edge” router
    - Using MAC address of router and L2 forwarding

- Series of routers carry packets to B’s L2 network
  - Looking at network portion of IP address

- B’s L2 network delivers packets to B
  - Using B’s MAC address and L2 forwarding
  - But, but, but ....
  - How do you find out what B’s MAC address is?
    - ARP (discussed later)
Hierarchical Structure

- The Internet is an “inter-network”
  - Used to connect networks together, not hosts

- Forms a natural two-way hierarchy
  - Wide area network (WAN) delivers to the right LAN
  - LAN delivers to the right host
Hierarchical Addressing

• Can you think of an example?

• Addressing in the US mail
  • Country
  • City, Zip code
  • Street
  • House Number
  • Occupant “Name”
Quick review

• Original IP addressing — Network:Host

• Elegant, but perhaps not sufficiently scalable

• How would you make it more scalable?
Extending the L3 solution

• If too many networks, then could add another layer

• ISP:Network:Host

• Network might be one of the many L2 networks within an ISP

• Can add additional levels of hierarchy (e.g., region)

• And can do flat routing at each level
  • Address can be both locator (prefix) and identifier (suffix)

• Simple, elegant, easy to implement, as scalable as one wants...

• But that’s not what happened :(
IP addresses

- Unique 32 bit numbers associated with an “interface” (link)

- Use dotted-quad notation, e.g., 12.34.128.5
Original Internet Addresses

• First eight bits: network address (/8)
  • Slash notation indicates network address

• Last 24 bits: host address

• Assumed 256 networks were more than enough!!!
  • Now we have millions!
Suppose we want to accommodate more networks

• We can allocate more bits to network address

• Problem?
  • Fewer bits for host names
  • What if some networks need more hosts?
Today’s Addressing: CIDR

- Classless Inter-domain Routing
- Flexible division between network and host addresses
- Must specify both address and mask
  - Clarifies the boundary between network and host
- Mask carried in routing algorithms
  - Not implicitly carried in address