CS4450

Computer Networks: Architecture and Protocols

Lecture 9
Spanning Tree Protocol
Internet Protocol

Spring 2018
Rachit Agarwal
Life Lessons

• Life is full of important people and events
  • **YOU**, my PhD students, colleagues, deadlines, family, friends, me ... 

• Sharing life across people is like sharing networks between users
  • Delays mostly due to just transmission and propagation;
    • My meetings, sleep (rare, but happens)
  • When `#incoming-packets > link load`, queueing delay is inevitable;
    • If `#emails > what I can handle`, queueing delay (current status)
  • Sometimes failures happen — requires retransmitting packets

• Last week was one of those for me
  • Queueing delay at my inbox (too many emails to handle)
  • I am reducing the queue sizes ...
  • Help me!
Announcements

• Please give your TAs more work to do
  • I am happy to receive emails
  • Please cc the two TAs on emails: Justin (jmm825@), Burcu (bc633@)

• Problem Set 2 is out (and on the webpage now)

• Quiz 2 solutions will be out soon (on Piazza)
  • Already graded! Available after class

• We will release the code for socket programming soon

• Thanks for notifying me before the class about absence
  • Please cc the TAs in future
Quiz 1 distribution

Quiz 1 Grades

<table>
<thead>
<tr>
<th>Grade (out of 30)</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-4</td>
<td>2</td>
</tr>
<tr>
<td>5-9</td>
<td>4</td>
</tr>
<tr>
<td>10-14</td>
<td>7</td>
</tr>
<tr>
<td>15-19</td>
<td>12</td>
</tr>
<tr>
<td>20-24</td>
<td>18</td>
</tr>
<tr>
<td>25-29</td>
<td>4</td>
</tr>
<tr>
<td>30-30</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Mean**: 17.22
- **Median**: 20
- **Std. deviation**: 6.214827562583942
Goals for Today’s Lecture

• Wrap up Switched Ethernet (and link layer)

• Start on IP (the Internet Protocol)
  • Packet Header as a network “interface”
Recap: Link Layer

• Originally a broadcast channel
  • MAC addresses (really, names)
  • CSMA/CD
    • Remember: Exponential back-off (more in problem set 2)
  • Why does Ethernet use frames?
  • How Link Layer builds on top of Physical Layer (that uses bits)
  • Bounds on network length and/or minimum frame size
    • Due to propagation delays

• More recently: switched Ethernet
  • Broadcast storm!
Switched Ethernet

- Enables concurrent communication
  - Host A can talk to C, while B talks to D
  - No collisions -> no need for CSMA, CD
  - No constraints on link lengths or frame size
Routing in “Extended LANs”

Bridges relay broadcasts from one LAN to the other

Local-Area Network (LAN)
Naïvely Routing in “Extended LANs”: Broadcast storm

Local-Area Network (LAN)

Bridges relay broadcasts from one LAN to the other
How to avoid the Broadcast Storm Problem?

Get rid of the loops!
Let's get back to the graph representation!
Easiest Way to Avoid Loops

• Use a network topology (graph) where loop is impossible!

• Take arbitrary topology (graph)

• Build spanning tree
  - Subgraph that includes all vertices but contains no cycles
  - Links not in the spanning tree are not used in forwarding frames

• Only one path to destinations on spanning trees
  - So don't have to worry about loops!
Consider Graph
A Spanning Tree
Another Spanning Tree
Yet Another Spanning Tree
Spanning Tree Protocol

- Protocol by which bridges construct a spanning tree

- Nice properties
  - Zero configuration (by operators or users)
  - Self healing

- Still used today

- Constraints for backwards compatibility
  - No changes to end-hosts
  - Maintain plug-n-play aspect

- Earlier Ethernet achieved plug-n-play by leveraging a broadcast medium
  - Can we do the same for a switched topology?
Algorithm has Two Aspects...

• Pick a root:
  • Destination to which the shortest paths go
  • Pick the one with the smallest identifier (MAC address)

• Compute the shortest paths to the root
  • No shortest path can have a cycle
  • Only keep the links on the shortest path
  • Break ties in some way
    • so we only keep one shortest path from each node

• Ethernet’s spanning tree construction does both with a single algorithm
Breaking Ties

• When there are multiple shortest paths to the root,
  • Choose the path that uses the neighbor switch with the lower ID

• One could use any tie breaking system
  • This is just an easy one to remember and implement
Constructing a Spanning Tree

• Messages (Y,d,X)
  • From node X
  • Proposing Y as the root
  • And advertising a distance d to Y

• Switches elect the node with smallest identifier (MAC address) as root
  • Y in messages

• Each switch determines if a link is on its shortest path to the root
  • If not, excludes it from the tree
  • d to Y in the message is used to determine this
Steps in Spanning Tree Protocol

- **Messages** \((Y,d,X)\)
  - For root \(Y\); From node \(X\); advertising a distance \(d\) to \(Y\)

- Initially each switch proposes itself as the root
  - that is, switch \(X\) announces \((X,0,X)\) to its neighbors

- Switches update their view
  - Upon receiving message \((Y,d,Y)\) from \(Z\), check \(Y\)'s id
  - If \(Y\)'s id < current root: set root = \(Y\)

- Switches compute their distance from the root
  - Add 1 to the shortest distance received from a neighbor

- If root or shortest distance to it *changed*, send neighbors updated message \((Y,d+1,X)\)
Group Exercise:
Run the Spanning Tree Protocol on this example
Example
(root, dist, from)
Example
(root, dist, from)
Example
(root, dist, from)
Links on Spanning Tree

• 3-1
• 5-1
• 6-1
• 2-3
• 4-2
• 7-2
Robust Spanning Tree Algorithm

- Algorithm must react to **failures**
  - Failure of the root node
  - Failure of switches and links

- Root node sends periodic announcement messages
  - Other switches continue forwarding messages

- Detecting failures through timeout (**soft state**)  
  - If no word from root, time out and claim to be the root!
Self-resilient upon link/node failures (suppose node 1 fails)

- 2 is new root
- 3-2
- 6-2
- 4-2
- 7-2
- 5-6
The end of Link Layer ....

And the beginning of network layer :-D

- Built on top of reliable delivery
- Built on top of best-effort forwarding
- Built on top of best-effort routing
- Built on top of physical bit transfer
Network Layer

- THE functionality: **delivering the data**

- **THE protocol: Internet Protocol (IP)**
  - To achieve its functionality (delivering the data), IP protocol has **three** responsibilities

- Addressing (next lecture)
- Encapsulating data into packets (actually datagrams; next lecture)
- Routing (using a variety of protocols; several lectures)
Internet Protocol

• THE functionality: delivering the data

• THE protocol: Internet Protocol (IP)
  • To achieve its functionality (delivering the data), 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?)*

![Header Opaque Payload Diagram]

- 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 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

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

- **32-bit Source IP Address**
- **32-bit Destination IP Address**
- **Options (if any)**
- **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 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
# Fields for Reading Packet Correctly

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>Version</td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td>Header Length</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 (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>
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?
### Checksum Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>Version</td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td>Length</td>
</tr>
<tr>
<td>8-bit Type of Service (TOS)</td>
<td>Type of Service</td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td>Total Length</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</td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td>Protocol</td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td>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</td>
</tr>
<tr>
<td>Payload</td>
<td>Payload</td>
</tr>
</tbody>
</table>
Thats it for today