Data Communications

LECTURE: THE NETWORK LAYER

Macquarie University

The Network Layer

WHAT HAPPENS IN THIS LAYER?

We’ve got some concepts to cover first

The Network Layer

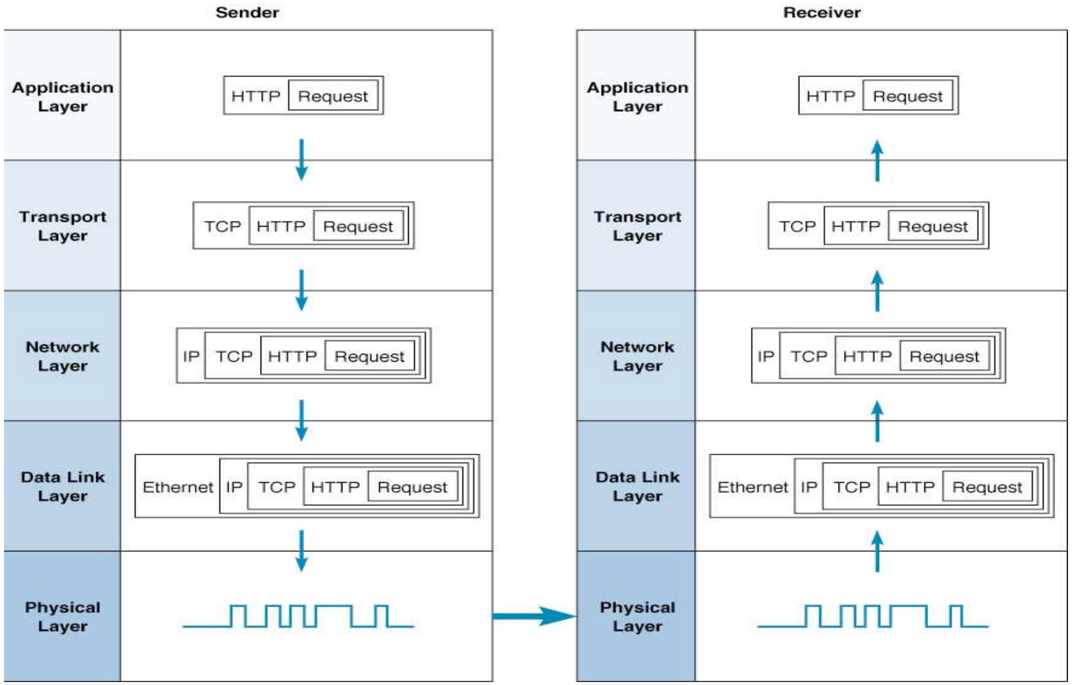
WHAT ARE WE LOOKING AT?

	protocol data unit	layer
Host Layers	data	application
		Network process to application
	data	presentation
		Data Representation and Encryption
	data	session
		Internet Communication
	segment	transport
		End-to-end Connections and Reliability
Media Layers	datagram	network
		Path determination and Logical Addressing
	frame	data-link
		Physical Addressing and Media Access
	bit	physical
		Media, Signal, and Binary Tranmission

LAYER OF OPERATION

- End-to-end delivery of packets
- Nodes still have layer-2 connectivity, but layer-3 provides communication between networks

TCP/IP 5-Layer Network Model



TCP/IP Protocol Family

TCP/IP NETWORK LAYER

- ARP, RARP, DHCP, BOOTP, IP, ICMP
- IP Routing (eg. RIP, OSPF)
- IP Multicasting: IGMP, DVMRP
- IP Error messages: ICMP

TCP/IP TRANSPORT LAYER

- TCP and UDP (separate lectures)

Network Layer

TWO MAJOR (AND RELATED) FUNCTIONS

ROUTING

- Routing is a distributed application that runs on multiple *routers*
  - Hence it is actually a layer-5 activity
  - However, it effects the way routers forward traffic
- Collects network topology into routing tables

FORWARDING

- Forwarding is the process of handling incoming packets
  - This task is a layer-3 activity
  - Forwards packets to next hop

Forwarding

NETWORK LAYER

- Concerned with end-to-end delivery

DATA-LINK LAYER

- Concerned with each hop (link)

Looking more closely

NETWORK CONNECTIONS

Network Devices

HOST

- An end system
  - Traffic either starts or finishes at a host (mostly)

ROUTER

- An intermediate system
  - Passes messages to another network

Internet Protocol

RFC 791

IP PROVIDES

- Universal addressing
  - IP addresses and only IP addresses used everywhere
- Routing of packets between hosts

IP DELIVERS

- Packets from source host to destination host
- End-to-end, not link-to-link

IP IS AN UNRELIABLE CONNECTIONLESS PACKET DELIVERY SERVICE

Unreliable?

WHY WOULD WE WANT IT TO BE UNRELIABLE?

UNRELIABLE MEANS

- Delivery is not guaranteed.
- Makes a best effort attempt to deliver a packet.
- Along the way a packet might be: lost, corrupted, delayed, duplicated
- Acknowledgment not required when data is received.
- Sender/receiver not informed when a packet is lost, etc
- Acknowledgment of packets is responsibility of higher layer transport protocol such as TCP.

Unreliable?

WHY WOULD WE WANT IT TO BE UNRELIABLE?

SOMETIMES RELIABILITY IS TOO COSTLY

- Real time traffic (video, audio)
- Some packet loss is tolerable
- If a packet is lost or corrupted, trying to recover can make things worse

SOMETIMES RELIABILITY IS NECESSARY

- Files must be delivered without corruption
- Delegate this problem to the transport layer
  - Separation of concerns

Connectionless

ORDER IS UNIMPORTANT TO IP

- Every packet is treated as a separate message
- No ordering relationship between consecutive packets
- Packets are just transmitted individually
- If order is important this becomes the transport layer’s responsibility

Failure Model

CLASSIFYING THINGS THAT CAN GO WRONG ALONG THE WAY

PACKETS CAN BE

- Lost
- Corrupted
- Delayed — which can cause out of sequence delivery
- Duplicated

SIMPLE FAILURE MODEL

- Treat all errors as packet loss

Network Layer Functions

ADDRESSING (STATIC)

- Internet Addresses
- Assignment of addresses
- Address resolution

ROUTING (STATIC AND DYNAMIC)

- Routing protocols
- Information gathering about network topology

FORWARDING (DYNAMIC)

- Process of deciding what path a packet must take to reach destination

Addressing at the Network Layer

Who manages these addresses?

NETWORK LAYER ADDRESSING

MANAGED BY ICANN

- Internet Corporation for Assigned Names and Numbers
- Manages the assignment of both IP addresses and domain names
  - Both assigned at the same time and in groups
  - Manages some domains directly (e.g., .com, .org, .net) and
  - Authorises private companies to become domain name registrars as well
- Example: Macquarie University
  - URLs that end in .mq.edu.au
  - IP addresses in the 137.111.x.x range

Address Allocation

HOW DO YOU GET ONE?

IF YOU WANT A NETWORK – YOU DON'T GO TO ICANN

- You go to your ISP
- Your ISP has a large network – it assigns you a free subnet
- Your ISP got its subnet from a larger ISP
- The large ISP got its subnet from a regional address registry
- Regional Address registries authorised by ICANN

Dynamic Addressing

CAN CHANGE OVER TIME

ADDRESS SERVICES

- Addresses automatically assigned to clients only when they are connected to the network
- Eliminates permanently assigning addresses to clients
- When the computer is moved to another location, its new IP address is assigned automatically
- Provided by the Dynamic Host Control Protocol (DHCP)

Address Allocation

ADDRESS ALLOCATION SUMMARY

- Human administrator assigns all addresses
- Host itself assigns an address
  - Link local address space
    - IPv4 `169.254.0.0/16`
    - IPv6 `FE80::/10`
- DHCP
- Global Internet address (from ICANN via ISP)

IP Address

GLOBALLY UNIQUE ADDRESS

- IP implements a single addressing scheme over all underlying networks
- However there are currently two versions of IP
  - Each with their own addressing scheme, IPv4 and IPv6
  - This is a big problem
- The idea of a universal address is an important one:
  - All nodes are inside the network, none are outside
  - Hence, everything is reachable

IP Address

ADDRESSES IDENTIFY WHERE A NODE IS ON THE NETWORK

- Each address is comprised of two parts
  - Network ID — identifies systems that are located on the same network
    - All devices on the same network must have the same Network ID
    - The Network ID must be unique to the Internet
  - Host ID — identifies a device within a network
    - The Host ID must be unique within the Network ID

IPv4 Addresses

DID WE HAVE AN IP V1, V2, OR V3?

IPV4 ADDRESSES ARE A 32-BIT NUMBER

- Each address is comprised of two parts
  - Network ID — identifies systems that are located on the same network
    - All devices on the same network must have the same Network ID
    - The Network ID must be unique to the Internet
  - Host ID — identifies a device within a network
    - The Host ID must be unique within the Network ID

IPv4 Addresses

SEPARATING THE NETWORK AND HOST IDS

- In classful addressing, for humans, it’s easy, you just have to know which bytes belong to which part
- For routers it’s almost as easy.
- Routers need to know which of 32 bits of the address are the network id
- Routers use a mask to find the network part

IPv4 Addresses

IP ADDRESSES NOT JUST ASSIGNED AT RANDOM

- As we already know – machines in the same network have part of their address the same (network ID)
  - Analogous to how houses in the same street share part of their address
- The leading part is shared
- Not all networks are the same size, so the amount shared depends on the size of the network

IPv4 Addresses

NETWORK VS HOST BITS?

HOW DO WE KNOW

- which is the network part
- which is the host part

- We split up the whole IPv4 address space depending on the size of network
- The different sizes are called classes

Classless Addressing

CLASSFUL ADDRESSING

- Not flexible
- Not efficient

CLASSLESS ADDRESSING

- Boundary between Network ID and Host ID can be anywhere
- We will explore this idea through the mechanism of *subnetting*

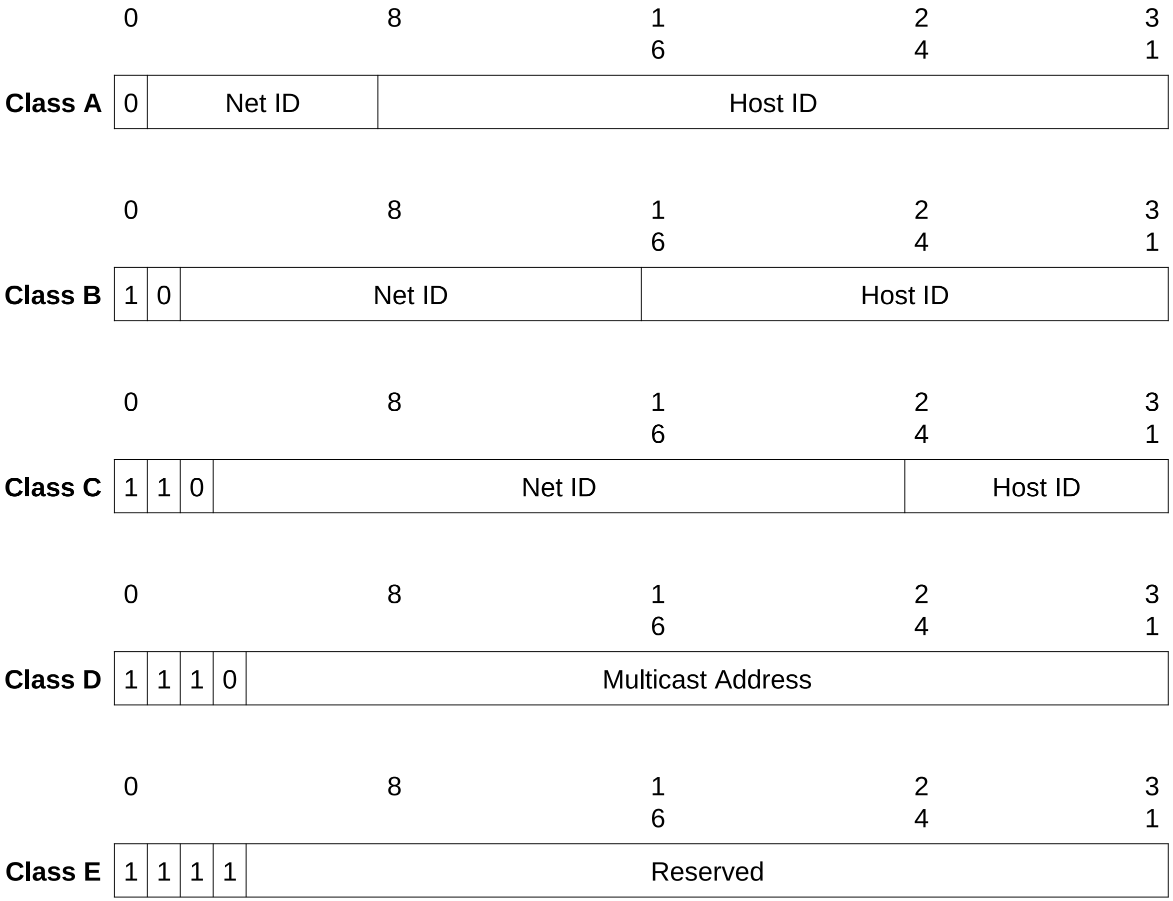
Special Addresses

SUMMARY OF SOME ONES TO KNOW ABOUT!

`All 0’s`	Startup Source Address
`All 1’s`	Limited Broadcast
`<network>All 0’s`	Directed Broadcast to <network>
`<network>All 1’s`	Network Address
`127.0.0.0/8`	Loopback
`224.0.0.0 - 239.255.255.255`	Multicast
`10.0.0.0/8`	RFC1918 NAT Private Network (Class A)
`172.16.0.0/12`	RFC1918 NAT Private Network (Class B)
`192.168.0.0/16`	RFC1918 NAT Private Network (Class C)
`169.254.0.0/16`	Link-local addresses

IPv4 Classful Addressing

NET ID = NETWORK ID (INCLUDING BITS AT THE START)



0 8 16 24 31

Class A | 0 | Net ID | Host ID

0 8 16 24 31

Class B | 1 | 0 | Net ID | Host ID

0 8 16 24 31

Class C | 1 | 1 | 0 | Net ID | Host ID

0 8 16 24 31

Class D | 1 | 1 | 1 | 0 | Multicast Address

0 8 16 24 31

Class E | 1 | 1 | 1 | 1 | Reserved

Numeric Addressing

HEX, DECIMAL, OR BINARY?

Read in decimal or hex, think of what is happening in binary

Decimal

NUMBERS REPRESENTED AS BASE 10

0 1 2 3 4 5 6 7 8 9

Example

DECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-10

Let’s look at an arbitrary number: 298

298

Example

DECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-10

Let's look at an arbitrary number: `298`

- Each digit has a weight and a position
- Weights are `0-9`

```
298 = 2×100 + 9×10 + 8×1
```

Example

DECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-10

Let’s look at an arbitrary number: 298

- Each digit has a weight and a position
- Weights are 0-9
- Positions are powers of 10

$$298 = 2\times100 + 9\times10 + 8\times1$$

$$= 2\times10^2 + 9\times10^1 + 8\times10^0$$

Example

DECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-10

Let's look at an arbitrary number: 298

- Each digit has a weight and a position
- Weights are 0-9
- Positions are powers of 10

If we increment any of the digits

$$298 = 2\times100 + 9\times10 + 8\times1$$

$$= 2\times10^2 + 9\times10^1 + 8\times10^0$$

$$+ 10$$

Example

DECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-10

Let's look at an arbitrary number: `298`

- Each digit has a weight and a position
- Weights are `0-9`
- Positions are powers of `10`

If we increment any of the digits

- and that digit is a `9` (`10` minus `1`),
- then that digits wraps around to `0`, and
- the next digit to the left increases by `1`.

$298 = 2\times100 + 9\times10 + 8\times1$

$= 2\times10^2 + 9\times10^1 + 8\times10^0$

$+ 10$

$= 308$

$= 3\times100 + 0\times10 + 8\times1$

$= 3\times10^2 + 0\times10^1 + 8\times10^0$

Example

DECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-10

Let's look at an arbitrary number: 298

- Each digit has a weight and a position
- Weights are 0-9
- Positions are powers of 10

If we increment any of the digits

- and that digit is a 9 (10 minus 1),
- then that digits wraps around to 0, and
- the next digit to the left increases by 1.

Adding any number of zeros on the left doesn't change the value of the number.

$298 = 2 \times 100 + 9 \times 10 + 8 \times 1$

$= 2 \times 10^2 + 9 \times 10^1 + 8 \times 10^0$
$+ 10$
$= 308$
$= 3 \times 100 + 0 \times 10 + 8 \times 1$
$= 3 \times 10^2 + 0 \times 10^1 + 8 \times 10^0$

$308 = 0308$

$= 0 \times 1000 + 3 \times 100 + 0 \times 10 + 8 \times 1$
$= 0 \times 10^3 + 3 \times 10^2 + 0 \times 10^1 + 8 \times 10^0$

Binary

NUMBERS REPRESENTED AS BASE 2

0 1

Example

BINARY ARITHMETIC

RADIX ARITHMETIC BASE-2

Let’s look at an arbitrary number: `1011`

```
1011
```

Example

BINARY ARITHMETIC

RADIX ARITHMETIC BASE-2

Let’s look at an arbitrary number: `1011`

- Each digit has a weight and a position
- Weights are `0-1`

```
1011 = 1×8 + 0×4 + 1×2 + 1×1
```

Example

BINARY ARITHMETIC

RADIX ARITHMETIC BASE-2

Let's look at an arbitrary number: 1011

- Each digit has a weight and a position
- Weights are 0-1
- Positions are powers of 2

$$\begin{aligned} 1011 &= 1\times8 + 0\times4 + 1\times2 + 1\times1 \\ &= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 \end{aligned}$$

Example

BINARY ARITHMETIC

RADIX ARITHMETIC BASE-2

Let's look at an arbitrary number: 1011

- Each digit has a weight and a position
- Weights are 0-1
- Positions are powers of 2

If we increment any of the digits

```
1011 = 1×8 + 0×4 + 1×2 + 1×1
```

$$= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$$

```
    + 10  (10 binary is 2 decimal)
```

Example

BINARY ARITHMETIC

RADIX ARITHMETIC BASE-2

Let’s look at an arbitrary number: 1011

- Each digit has a weight and a position
- Weights are 0-1
- Positions are powers of 2

If we increment any of the digits

- and that digit is a 1 (2 minus 1),
- then that digits wraps around to 0, and
- the next digit to the left increases by 1.

$$\begin{aligned}
1011 &= 1\times8 + 0\times4 + 1\times2 + 1\times1 \\
&= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 \\
&+ 10 \quad \text{(10 binary is 2 decimal)} \\
&= 1101 \\
&= 1\times8 + 1\times4 + 1\times2 + 1\times0 \\
&= 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0
\end{aligned}$$

Example

BINARY ARITHMETIC

RADIX ARITHMETIC BASE-2

Let’s look at an arbitrary number: 11

- Each digit has a weight and a position
- Weights are 0-1
- Positions are powers of 2

If we increment any of the digits

- and that digit is a 1 (2 minus 1),
- then that digits wraps around to 0, and
- the next digit to the left increases by 1.

Adding any number of zeros on the left doesn’t change the value of the number.

$$
\begin{aligned}
1011 &= 1\times8 + 0\times4 + 1\times2 + 1\times1 \\
&= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 \\
&+ 10 \quad \text{(10 binary is 2 decimal)} \\
&= 1101 \\
&= 1\times8 + 1\times4 + 0\times2 + 1\times1 \\
&= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0
\end{aligned}
$$

$$
\begin{aligned}
1101 &= 01101 \\
&= 0\times16 + 1\times8 + 1\times4 + 0\times2 + 1\times1 \\
&= 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0
\end{aligned}
$$

Hexadecimal

NUMBERS REPRESENTED AS BASE 16

0 1 2 3 4 5 6 7 8 9 a b c d e f

Example

HEXADECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-16

Let's look at an arbitrary number: `506`

`1fa`

Example

HEXADECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-16

Let's look at an arbitrary number: `506`

- Each digit has a weight and a position
- Weights are `0-15` (`10-15` use letters `a-f`)

```
1fa = 1×256 + 15×16 + 10×1
```

Example

HEXADECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-16

Let's look at an arbitrary number: `506`

- Each digit has a weight and a position
- Weights are `0-15` (`10-15` use letters `a-f`)
- Positions are powers of 16

```
1fa = 1×256 + 15×16 + 10×1
    = 1×16² + 15×16¹ + 10×16⁰
```

$$1fa = 1 \times 256 + 15 \times 16 + 10 \times 1 = 1 \times 16^2 + 15 \times 16^1 + 10 \times 16^0$$

Example

HEXADECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-16

Let’s look at an arbitrary number: `506`

- Each digit has a weight and a position
- Weights are `0-15` (`10-15` use letters `a-f`)
- Positions are powers of 16

If we increment any of the digits

$$
\begin{aligned}
\texttt{1fa} &= 1\times256 + 15\times16 + 10\times1 \\
&= 1\times16^{2} + 15\times16^{1} + 10\times16^{0} \\
&+ 10 \quad \text{(10 hex is 16 decimal)}
\end{aligned}
$$

Example

HEXADECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-16

Let's look at an arbitrary number: 506

- Each digit has a weight and a position
- Weights are 0-15 (10-15 use letters a-f)
- Positions are powers of 16

If we increment any of the digits

- and that digit is a f (16 minus 1),
- then that digits wraps around to 0, and
- the next digit to the left increases by 1.

1fa = 1×256 + 15×16 + 10×1

= $1 \times 16^2 + 15 \times 16^1 + 10 \times 16^0$

+ 10 (10 hex is 16 decimal)

= 20a

= 2×256 + 0×16 + 10×1

= $2 \times 16^2 + 0 \times 16^1 + 10 \times 16^0$

Example

HEXADECIMAL ARITHMETIC

RADIX ARITHMETIC BASE-16

Let’s look at an arbitrary number: 506

- Each digit has a weight and a position
- Weights are 0-15 (10-15 use letters a-f)
- Positions are powers of 16

If we increment any of the digits

- and that digit is a f (16 minus 1),
- then that digits wraps around to 0, and
- the next digit to the left increases by 1.

Adding any number of zeros on the left doesn’t change the value of the number.

$\texttt{1fa} = 1\times256 + 15\times16 + 10\times1$
$= 1\times16^2 + 15\times16^1 + 10\times16^0$
$+ 10$ (10 hex is 16 decimal)
$= \texttt{20a}$
$= 2\times256 + 0\times16 + 10\times1$
$= 2\times16^2 + 0\times16^1 + 10\times16^0$

$\texttt{20a} = \texttt{020a}$
$= 0\times4096 + 2\times256 + 0\times16 + 10\times1$
$= 0\times16^3 + 2\times16^2 + 0\times16^1 + 10\times16^0$

Converting between base 2-10-16

THERE’S A LONG WAY, AND THERE ARE SHORTCUTS

You won’t be able to use a calculator!!!

Radial Notation

USING SUBSCRIPT TO REPRESENT THE BASE

BASE-10	298_{10}	308_{10}
BASE-2	1011_{2}	1101_{2}
BASE-16	$1fa_{16}$	$20a_{16}$

Converting to Decimal

$$\begin{aligned} 1011_2 &= 1\times 8 + 0\times 4 + 1\times 2 + 1\times 1 \\ &= 8 + 0 + 2 + 1 \\ &= 11_{10} \end{aligned}$$

BINARY TO DECIMAL

Simple, just do the expansion and then add

$$1fa_{16} = 1\times 256 + 15\times 16 + 10\times 1$$

$$1fa_{16} = 256 + 240 + 10$$

$$1fa_{16} = 506_{10}$$

HEXADECIMAL TO DECIMAL

Almost as easy

Converting between Hex and Binary

WE CAN GO HIGHER THAN 16… BUT JUST THINK IN CHUNKS

Hex (base 16)	Decimal□(base 10)	Binary value (base 2)	4 bits…
0	0	0	0000
1	1	1	0001
2	2	10	0010
3	3	11	0011
4	4	100	0100
5	5	101	0101
6	6	110	0110
7	7	111	0111

Hex (base 16)	Decimal (base 10)	Binary value (base 2)	4 bits…
8	8	1000	1000
9	9	1001	1001
a	10	1010	1010
b	11	1011	1011
c	12	1100	1100
d	13	1101	1101
e	14	1110	1110
f	15	1111	1111

Try to convert 0xc0a8140a into binary

By the way the “0x” just means it is a number in hex

Treat it as c0a8140a

Converting between Hex and Binary

Example 1

$12a_{16}$ = $1\ 0010\ 1010_2$

Example 2

$11\ 1011\ 0111\ 0101_2$ = $3b75_{16}$

As 16 bit numbers

Example 3

$012a_{16}$ = $0000\ 0001\ 0010\ 1010_2$

Example 4

$0011\ 1011\ 0111\ 0101_2$ = $3b75_{16}$

HEXADECIMAL TO BINARY

Write each hex digit out as four bits.

BINARY TO HEXADECIMAL

Group bits into groups of 4, from the high and write each group as a hex digit.

A GOOD PRACTICE

If a number is specified as being *n*-bits in size then represented it with enough leading zeros to use all *n* bits.

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

`11`

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by `2`
2. Record the remainder (either `0` or `1`)
3. If the dividend is not zero, divide it by `2`
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3
6. Else, you're finished.

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by 2
2. Record the remainder (either 0 or 1)
3. If the dividend is not zero, divide it by 2
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3
6. Else, you're finished.

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5 remainder 1
```

```
1
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by 2
2. Record the remainder (either `0` or `1`)
3. If the dividend is not zero, divide it by 2
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5 remainder 1
5/2  = 2
```

```
1
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by 2
2. Record the remainder (either 0 or 1)
3. If the dividend is not zero, divide it by 2
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5 remainder 1
5/2  = 2 remainder 1
```

```
11
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by 2
2. Record the remainder (either `0` or `1`)
3. If the dividend is not zero, divide it by 2
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5 remainder 1
5/2  = 2 remainder 1
2/2  = 1



11
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by 2
2. Record the remainder (either 0 or 1)
3. If the dividend is not zero, divide it by 2
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5 remainder 1
5/2  = 2 remainder 1
2/2  = 1 remainder 0


011
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by 2
2. Record the remainder (either 0 or 1)
3. If the dividend is not zero, divide it by 2
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3

Converting From Decimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5 remainder 1
5/2  = 2 remainder 1
2/2  = 1 remainder 0
1/2  = 0

011
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by `2`
2. Record the remainder (either `0` or `1`)
3. If the dividend is not zero, divide it by `2`
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3

Final Result - Decimal to Binary

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
11/2 = 5 remainder 1
5/2  = 2 remainder 1
2/2  = 1 remainder 0
1/2  = 0 remainder 1

1011
```

DECIMAL TO BINARY

To convert a decimal number *n* to binary

1. Divide *n* by 2
2. Record the remainder (either `0` or `1`)
3. If the dividend is not zero, divide it by 2
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3
6. Else, you’re finished.

Decimal to Hexadecimal

THIS USUALLY TAKES THE LONGEST TIME (MANUALLY)

```
506/16 = 31 remainder 10
31/16  =  1 remainder 15
1/16   =  0 remainder  1



1fa
```

DECIMAL TO HEXADECIMAL

To convert a decimal number *n* to hex

1. Divide *n* by `16`
2. Record the remainder (between `0` and `f`)
3. If the dividend is not zero, divide it by `16`
4. Add the remainder to the **front**
5. If the dividend is not zero go to Step 3
6. Else, you’re finished.

Breathe!

THOSE PREVIOUS SLIDES ARE THERE IF YOU NEED TO GO OVER THEM

Working with Addresses

BROADLY, AND THEN WE’LL TALK ABOUT SUBNETTING

Addressing

AT VARIOUS LAYERS - IDENTIFYING DIFFERENT THINGS

Layer	Address Kind	Computer Representation	Human Representation	Example
Application	Application Dependent HTTP - URLs	String	String	http://www.mq.edu.au/...
Transport	Port - destination application	16 bit field	Number 0-65,535	24
Network	IP Address IPv4 IPv6	 32 bit field 128 bit field	 4 decimals 0-255 8x4 hex digits	 134.57.33.2 4534:4EF3:4AFD:A43F:4567:E34F:236B:453F
Data Link	MAC address (Ethernet)	48 bit field (6 bits)	6x2 hex digits	AE:56:23:F4:65:D3
Physical	Bits put in one end come out the other!	N/A	N/A	Bits are broadcast on link, i.e.., flood.

Ethernet Address

LAYER 2 (DATA LINK LAYER)

ADDRESSING SYSTEM FOR LAYER-2

- Used on the local network as part of media access
  - Also know as a MAC address, or sometimes as a hardware address
- Ethernet addresses are 48-bits in size and are written in hexadecimal
  - Normally each octet is separated by a colon, `B6:CD:35:3B:C7:6F`
  - However it may be a dash instead, `B6-CD-35-3B-C7-6F`
- The case of the digits doesn't matter, `b6:cd:35:3b:c7:6f` is also valid
- And, sometimes they may be group as 3 sets of 16 bits, `b6cd-353b-c76f`

IP Addresses

LAYER 3 (NETWORK LAYER)

ADDRESSES SYSTEM FOR LAYER-3

- Each address is comprised of two parts
  - Network ID — identifies systems that are located on the same network
    - All devices on the same network must have the same Network ID
    - The Network ID must be unique to the Internet
  - Host ID — identifies a device within a network
    - The Host ID must be unique within the Network ID

IP Addresses

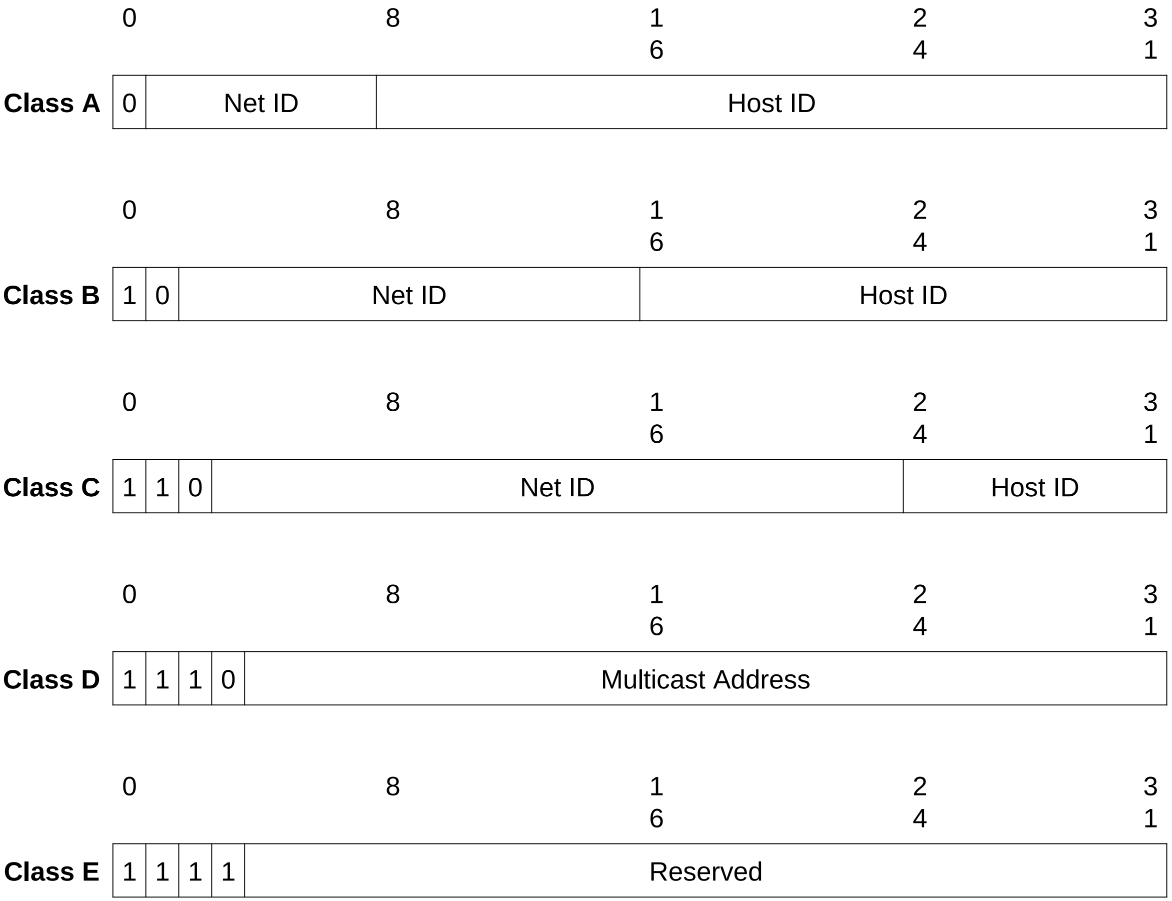
LAYER 3 (NETWORK LAYER)

HOW DO WE KNOW

- which is the network part of the IP address
- which is the host part of the IP address?

IPv4 Classful Addressing

RECALL FROM BEFORE…



0 8 16 24 31

Class A 0 | Net ID | Host ID

0 8 16 24 31

Class B 1 | 0 | Net ID | Host ID

0 8 16 24 31

Class C 1 | 1 | 0 | Net ID | Host ID

0 8 16 24 31

Class D 1 | 1 | 1 | 0 | Multicast Address

0 8 16 24 31

Class E 1 | 1 | 1 | 1 | Reserved

IPv4 Classful Addressing

RECALL FROM BEFORE…

	First Address	Last Address	Network ID	Host ID	Networks	Hosts*
Class A	0.0.0.0	127.255.255.255	8 bits	24 bits	128	16,777,216
Class B	128.0.0.0	191.255.255.255	16 bits	16 bits	16,384	65,536
Class C	192.0.0.0	223.255.255.255	24 bits	8 bits	4,194,304	256
Class D	224.0.0.0	239.255.255.255	268,435,456			
Class E	240.0.0.0	255.255.255.255	268,435,456			

Classless Addressing

COMPARED TO CLASSFUL ADDRESSING

CLASSFUL ADDRESSING

- Not flexible
- Not efficient

CLASSLESS ADDRESSING

- Boundary between Network ID and Host ID can be anywhere
- We will explore this idea through the mechanism of *subnetting*

Let’s use Network Masks

ANOTHER FIELD TO SAY WHICH BITS ARE NETWORK

00010111001010000100100100010011

Convert the above IP from binary to Decimal Octets.

23.40.73.19

Let’s say the network address is the first 8 bits… then we’d need to specify a network mask

Written in binary

IP address:	00010111001010000100100100010011
Net Mask:	11111111000000000000000000000000
Network Part:	*00010111000000000000000000000000*

8-bit chunks written as decimal numbers

Let’s use Network Masks

ANOTHER FIELD TO SAY WHICH BITS ARE NETWORK

00010111001010000100100100010011

Convert the above IP from binary to Decimal Octets.

23.40.73.19

Let’s say the network address is the first 11 bits… then we’d need to specify a network mask

Written in binary

IP address:	00010111001010000100100100010011
Net Mask:	11111111111000000000000000000000
Network Part:	*00010111001000000000000000000000*

8-bit chunks written as decimal numbers

Let’s use Network Masks

ANOTHER FIELD TO SAY WHICH BITS ARE NETWORK

00010111001010000100100100010011

Convert the above IP from binary to Decimal Octets.

23.40.73.19

Let’s say the network address is the first 23 bits… then we’d need to specify a network mask

Written in binary

IP address: 00010111001010000100100100010011

Net Mask: 11111111111111111111111000000000

Network Part: 00010111001010000100100000000000

8-bit chunks written as decimal numbers

Binary Operations

“NOT”, “OR”, “AND"

LOGICAL OPERATORS

Just as there are numeric operations such as addition, subtraction, and multiplication, there are also so-called logical operations that we can perform

NOT	
0	1
1	0

OR	0	1
0	0	1
1	1	1

AND	0	1
0	0	0
1	0	1

NOT	1011
	0100

1011	OR
1001	
1011	

1011	AND
1001	
1001	

Network Mask - a closer look

ANOTHER FIELD TO SAY WHICH BITS ARE NETWORK

DETERMINING THE NETWORK ID

Let’s write out 137.111.216.5 in binary…

	First octet	Second octet	Third octet	Fourth octet
IP address	10001001	01101111	11011000	00000101
Mask	11111111	11111111	00000000	00000000
Result	10001001	01101111	00000000	00000000

Which operation have we just performed…. *bitwise logical **and***

Network Mask - a closer look

IPV4 CLASSFUL NETMASKS

	Network ID	Host ID	Netmask Binary	Netmask Dotted-decimal
Class A	8 bits	24 bits	11111111 00000000 00000000 00000000	255.0.0.0
Class B	16 bits	16 bits	11111111 11111111 00000000 00000000	255.255.0.0
Class C	24 bits	8 bits	11111111 11111111 11111111 00000000	255.255.255.0

All-zeros and all-ones addresses

Layer-2	Function
All 0's (00:00:00:00:00:00)	Unused (OUI 00:00:00 belongs to Xerox, so technically a valid address)
All 1's (ff:ff:ff:ff:ff:ff)	Broadcast (All stations address, send to every node in the broadcast domain)

Layer-3	Function
All 0's (0.0.0.0)	Startup Source Address
All 1's (255.255.255.255)	Limited Broadcast
<network id>.all 0's (eg 137.111.0.0)	Network Address
<network id>.all 1's (eg 137.111.255.255)	Directed Broadcast to network (eg. All nodes on the 137.111.0.0/16 network)

Special Addresses

Special Address Range	Function
127.0.0.0/8	Loopback
224.0.0.0 – 239.255.255.255	Multicast (Class D)
10.0.0.0/8	RFC1918 NAT Private Network (Class A)
172.16.0.0/12	RFC1918 NAT Private Network (Class B)
192.168.0.0/16	RFC1918 NAT Private Network (Class C)
169.254.0.0/16	Link-local addresses

Subnetting

I’VE BEEN GIVEN A NETWORK ADDRESS, BUT I WANT TO SPLIT IT UP

Creating a number of “subnetworks” within my organisation

Subnetting Example

WE WANT TO SUBNET THE CLASS C NETWORK 197.15.22.0

- We have a Class C network, 24 bits of Network ID and 8 bits of Host ID
- There are 2^8 (256) possible addresses on the network.
- So, we have all the addresses from 197.15.22.0 through 197.15.22.255
- Addresses outside this range are not ours to use
- The subnetting process does not change the total number of addresses
- Subnetting divides the address space into multiple smaller networks
- So after subnetting, we should have 4 networks each with 64 addresses.

Subnetting Mechanism

SUBNETTING IS CHANGING THE NETMASK

- Recall that performing a bitwise logical and of the address against the netmask provides the Network ID
- If we change the netmask, then we also change which part of the address identifies which network we are dealing with.
- Subnetting is simply changing the netmask in an appropriate way.

Let’s use Network Masks

ANOTHER FIELD TO SAY WHICH BITS ARE NETWORK

00010111001010000100100100010011

Convert the above IP from binary to Decimal Octets.

23.40.73.19

Let’s say the network address is the first 11 bits… then we’d need to specify a network mask

Written in binary

IP address:	00010111001010000100100100010011
Net Mask:	11111111111000000000000000000000
Network Part:	*00010111001000000000000000000000*

8-bit chunks written as decimal numbers

Let’s use Network Masks

ANOTHER FIELD TO SAY WHICH BITS ARE NETWORK

00010111001010000100100100010011

Convert the above IP from binary to Decimal Octets.

23.40.73.19

Let’s say the network address is the first 11 bits… let’s split it in 4… We’ll need 2 extra bits to “extend” the mask

Written in binary

IP address:	00010111001010000100100100010011
Net Mask:	11111111111000000000000000000000
Network Part:	*00010111001000000000000000000000*

8-bit chunks written as decimal numbers

Let’s use Network Masks

ANOTHER FIELD TO SAY WHICH BITS ARE NETWORK

00010111001010000100100100010011

Convert the above IP from binary to Decimal Octets.

23.40.73.19

Let’s say the network address is the first 11 bits… let’s split it in **8**? We’ll need **3** extra bits to “extend” the mask

Written in binary

IP address:	00010111001010000100100100010011
Net Mask:	11111111111000000000000000000000
Network Part:	*00010111001000000000000000000000*

8-bit chunks written as decimal numbers

Subnetting Example

WE WANT TO SUBNET THE CLASS C NETWORK 197.15.22.0

We currently have one Network ID and one Directed Broadcast address.

	First octet	Second octet	Third octet	Fourth octet
Network Address	11000101	00001111	00010110	00000000
Broadcast Address	11000101	00001111	00010110	11111111
Netmask	11111111	11111111	11111111	00000000

In order to have four networks in the same space where there is currently one, we need to change the netmask to give four Network IDs.

Subnetting Example

WE WANT TO SUBNET THE CLASS C NETWORK 197.15.22.0

- We want to change the fourth octet for this example
- How many bits do we need to give us four IDs?

4 = 2^2, so we need 2 bits

	Fourth octet
Netmask	00000000

→

	Fourth octet
Netmask	11000000

- This changes the NetID:HostID bit ratio from 24:8 to 26:6
- This means that each subnet will have 2^6 = 64 hosts

Subnet IDs

CLASS C NETWORK 197.15.22.0

SUBNET NETWORK ADDRESSES

Subnet #	Subnet ID	First octet	Second octet	Third octet	Fourth octet	Network ID	Sanity check
0	**00**	197	15	22	**00** 000000	0	0×64
1	**01**	197	15	22	**01** 000000	64	1×64
2	**10**	197	15	22	**10** 000000	128	2×64
3	**11**	197	15	22	**11** 000000	192	3×64

SUBNET DIRECTED BROADCAST ADDRESSES

Subnet #	Subnet ID	First octet	Second octet	Third octet	Fourth octet	Broadcast	Sanity check
0	**00**	197	15	22	**00** 111111	63	Subnet 1 less 1
1	**01**	197	15	22	**01** 111111	127	Subnet 2 less 1
2	**10**	197	15	22	**10** 111111	191	Subnet 3 less 1
3	**11**	197	15	22	**11** 111111	255	original b/cast

Subnet IDs

CLASS C NETWORK 197.15.22.0

SUBNET NETWORK ADDRESSES

Subnet #	Network Address	First Host	Last Host	Broadcast Address
0	197.15.22.0	197.15.22.1	197.15.22.62	197.15.22.63
1	197.15.22.64	197.15.22.65	197.15.22.126	197.15.22.127
2	197.15.22.128	197.15.22.129	197.15.22.190	197.15.22.191
3	197.15.22.192	197.15.22.193	197.15.22.254	197.15.22.255

MAXIMUM HOSTS PER SUBNET

Every network has a network address and a broadcast address

Maximum hosts is 2^n-2 where *n* is the number of bits for the HostID

Thus there are a maximum of 62 hosts per subnet.

Subnet Mask

Netmask	First octet	Second octet	Third octet	Fourth octet	Dotted-decimal
Before subnetting	11111111	11111111	11111111	00000000	255.255.255.0
After subnetting	11111111	11111111	11111111	11000000	255.255.255.192

Useful Netmask Values

Binary	Decimal
11111111	255
11111110	254
11111100	252
11111000	248
11110000	240
11100000	224
11000000	192
10000000	128
00000000	0

TABLE OF USEFUL NETMASK VALUES

It might seem daunting to have to calculate these values, however there are only 9 possible legal values for each octet in a netmask.

Summary of the Example

CREATING SUBNETS FOR CLASS C NETWORK 197.15.22.0

197.15.22.0

Written in binary

Network address:	11000101000011110001011010000000
Broadcast address:	11000101000011110001011011111111
Net Mask:	11111111111111111111111111100000

Let's create 4 subnetworks (so use 2 extra bits)

Summary of the Example

197.15.22.0 INTO 4 SUBNETWORKS (USABLE?)

197.15.22.0

Written in binary

Network address: 11000101000011110001011000000000

Broadcast address: 11000101000011110001011011111111

Net Mask: 11111111111111111111111111000000

Let's create 4 subnetworks (so use 2 extra bits)

Network 1 address: 11000101000011110001011000000000

Broadcast address: 11000101000011110001011000111111

Net Mask: 11111111111111111111111111000000

Network 2 address: 11000101000011110001011001000000

Broadcast address: 11000101000011110001011001111111

Net Mask: 11111111111111111111111111000000

Network 2 address: 11000101000011110001011010000000

All Zero’s and all One’s subnets

BEING POLITE TO YOUR NEIGHBOURS

Traditionally, it has been recommended to avoid using the first and last subnets

This is due to the possible ambiguity between the network and broadcast addresses of the original network (before subnetting) and the network address of the first subnet and the broadcast address of the last subnet respectively.

These two subnets are useable, however the convention is to only use them when those subnets are internal to your organisation, and nobody else’s router need know about the subnetting, otherwise it is considered polite to not use them.

But…

197.15.22.0 INTO 4 SUBNETWORKS (USABLE?)

197.15.22.0

Written in binary

Network address: 11000101000011110001011000000000

Broadcast address: 11000101000011110001011011111111

Net Mask: 11111111111111111111111111000000

Let’s create 4 subnetworks (so use 2 extra bits)

Network 1 address: 11000101000011110001011000000000

Broadcast address: 11000101000011110001011000111111

Net Mask: 11111111111111111111111111000000

Network 2 address: 11000101000011110001011001000000

Broadcast address: 11000101000011110001011001111111

Net Mask: 11111111111111111111111111000000

If we wanted to be “polite”, but still need 4 usable subnets, how many subnets do we actually need to create?

Network 2 address: 11000101000011110001011010000000

Let’s use Network Masks

197.15.22.0 INTO 4 SUBNETWORKS (USABLE?)

197.15.22.0

Written in binary

Network address: 11000101000011110001011000000000

Broadcast address: 11000101000011110001011011111111

Net Mask: 11111111111111111111111111100000

We would actually need to create 8 subnetworks to get 4 USABLE ones in this circumstance… So we need to use 3 bits rather than 2

Go through the steps outlined in the previous slides…

Question 1: What are the network addresses for the 8 subnets?

Question 2: Which are the two subnetworks that would not be used?

Big Ideas from today

WHAT HAVE WE BEEN FOCUSING ON?

1) Routing vs Forwarding

2) Unreliable connections? Why? + Failure Models

3) Classful vs Classless IP addresses

4) Network Masks

5) Working with Binary, Decimal and Hexadecimal for IP addresses

6) Addresses: Network, broadcast, and other special addresses

7) Creating subnets

- given a network, create n subnetworks
- given a network, create as many subnets that can support x hosts per subnet