CSE 523S: Systems Security

k

Co

Systems

Spring 2018
Jon Shidal

Plan for Today

- Announcements
  - You should have completed the Python tutorial
  - Get started on HW2… There is an account creation step that requires operator approval. **Don’t wait until the last minute**, the operator may not be

- Security News? Question
- Assignment
- System Design & Security
  - [x] Why are our computer systems vulnerable?
  - Why are our networks vulnerable?

Assignment

- Wednesday
  - HTAOE: Ch. 2 81-114
- Monday
  - HW2 due
  - HTAOE: C

WHY ARE OUR NETWORKS VULNERABLE?

Networks are Vulnerable

- IP has an any-to-any communications model
  - Within IP you cannot control who sends you a packet

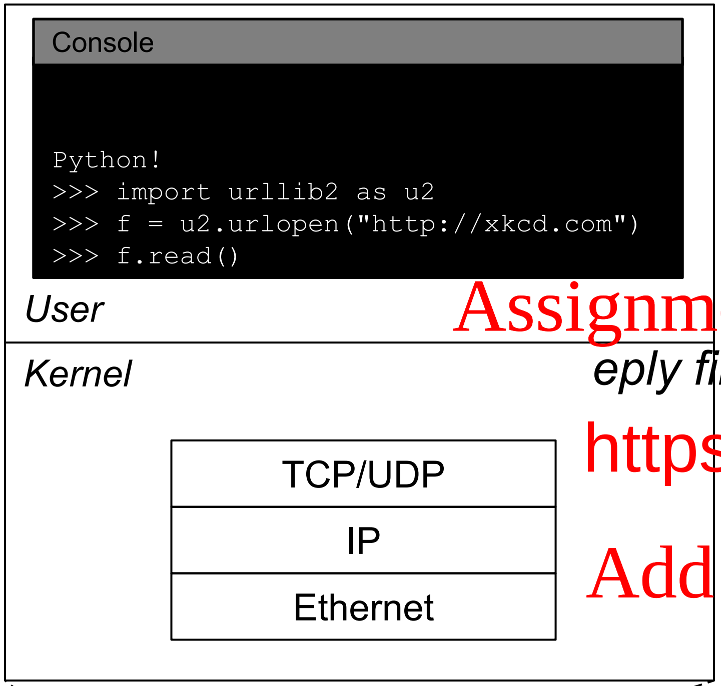
- Networks have weak authentication
  - When a packet arrives, you trust the source address

- mes& Binding betwee addresses are b

  - Insecure services map betwe layers (eg, IP to Ethernet), and names to addr

- Secure the “channel” only
  - You really want to secure the data and its source, not an address

Understanding Networks



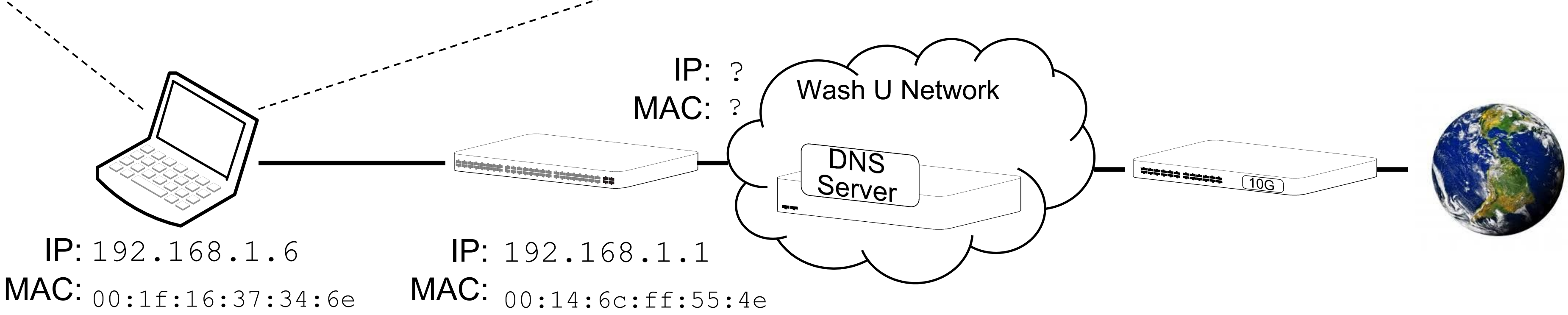
```
Python!
>>> import urllib2 as u2
>>> f = u2.urlopen("http://xkcd.com")
>>> f.read()
```

What do we need to know to answer these questions:

How does the request find its way to the server?

eply find its way back to the

On how does the reply find its app?



Packets are bit strings

```
ffffffffffff001f
1637346e08060001
080006040001001f
1637346ec0
0000000000
0101000000000000
0000000000000000
00000000
```

```
char pkt[] =
"\xff\xff\xff\xff\xff\x
ff\x00\x1f\x16\x37\x34\
x6e\x08\x06\x00\x01\x08
\x00\x06\x04\x00\x01\x0
\x37\x34\x6e\x
1\x06\x00\x
x00\x00\x00\x00\xc0\xa8
\x01\x01\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\
x00\x00\x00";
```

If we knew the format rules we understand this packet to be… we’ll decode it in a later slide

Network Layering

- Network protocols are layered; they have well-defined interfaces and separation of concerns

- Typical Internet layering
  - Application
  - TCP
  - IP
  - Ethernet
  - Physical link: wired or wifi

- Network packets encapsulate one protocol inside another
  - (Ethernet (IP (TCP (Application))))

- Applications typically use the “sockets” interface, and specify TCP or UDP
  - All lower-level details are the concern of the OS and underlying infrastructure

- **Our concern is with TCP/IP and Ethernet**

Ethernet

- Is the dominant wired-LAN technology

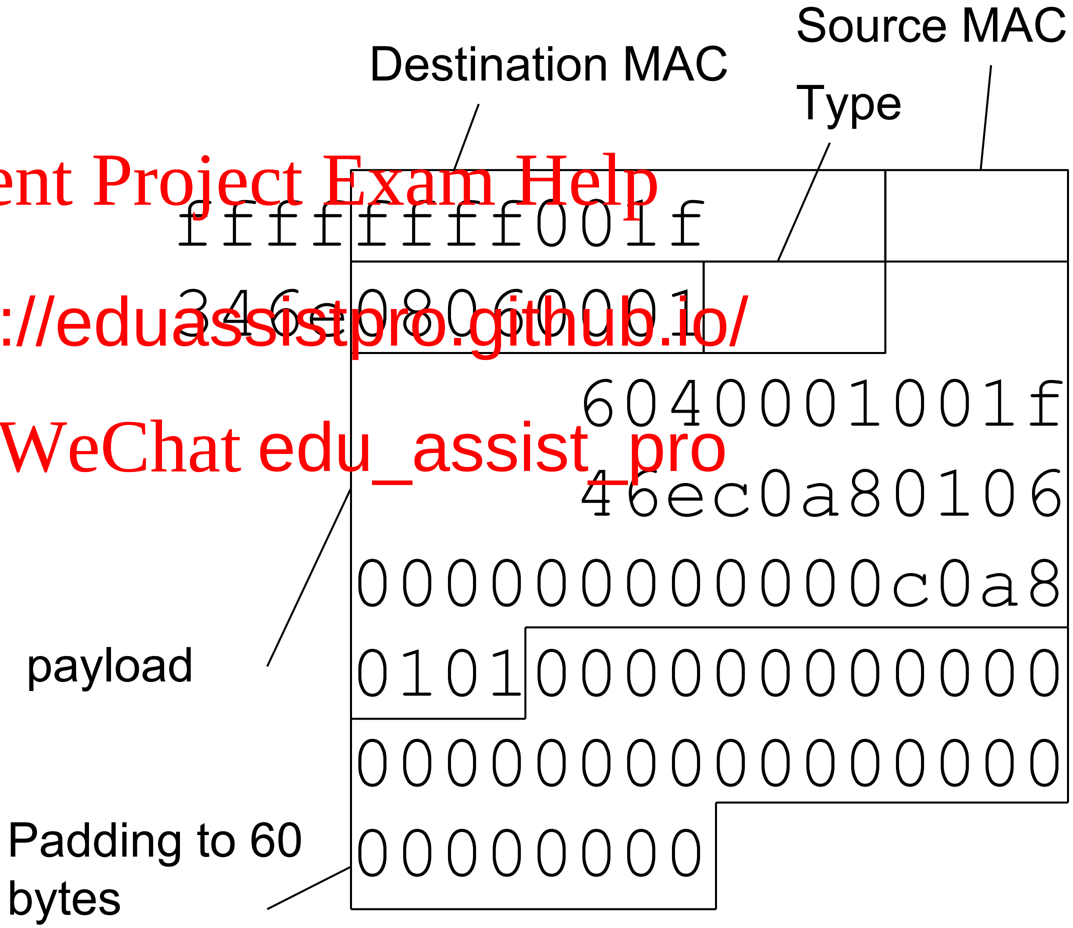
- Much to learn about its history, in your spare time
  - Used to be proprietary, now an IEEE standard
  - Used to be shared medium, now is switched
  - Always gets faster: 1M, 10M, 100M, 1G, 10G, …
  - Is rapidly becoming the only wired protocol that matters (LAN, campus, metro,

- Ethernet features
  - Variable length packets
  - Point-to-point communication between machines with MAC addresses
  - Broadcast: send packet to all nodes on local network
  - Virtual LANs (VLANs): limit broadcast domains to a VLAN
  - Uses "**type**" field to help receiver know what to do next

Ethernet II Frame Format

<table>
<tr><td>Byte Offset</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>0</td><td colspan="7">Preamble (pattern 10101010 repeated 7 times)</td><td>SFD
10101011</td></tr>
<tr><td>8</td><td colspan="6">Desti</td><td colspan="2">Source MAC address</td></tr>
<tr><td>16</td><td colspan="4">Source MAC address, continued</td><td colspan="4">ag (opt)</td></tr>
<tr><td>24</td><td colspan="2">Type</td><td colspan="6">42-1500 payload octets</td></tr>
<tr><td>68 to 1526</td><td colspan="4">32-bit CRC</td><td colspan="4">Interframe gap</td></tr>
<tr><td>72 to 1532</td><td colspan="8">Interframe gap, continued</td></tr>
</table>

Ethernet II Illustrated Frame



```
ffffffffffff001f
346e08060001
     604000 1001f
     46ec0a80106
0000000000000c0a8
0101000000000000
0000000000000000
00000000
```

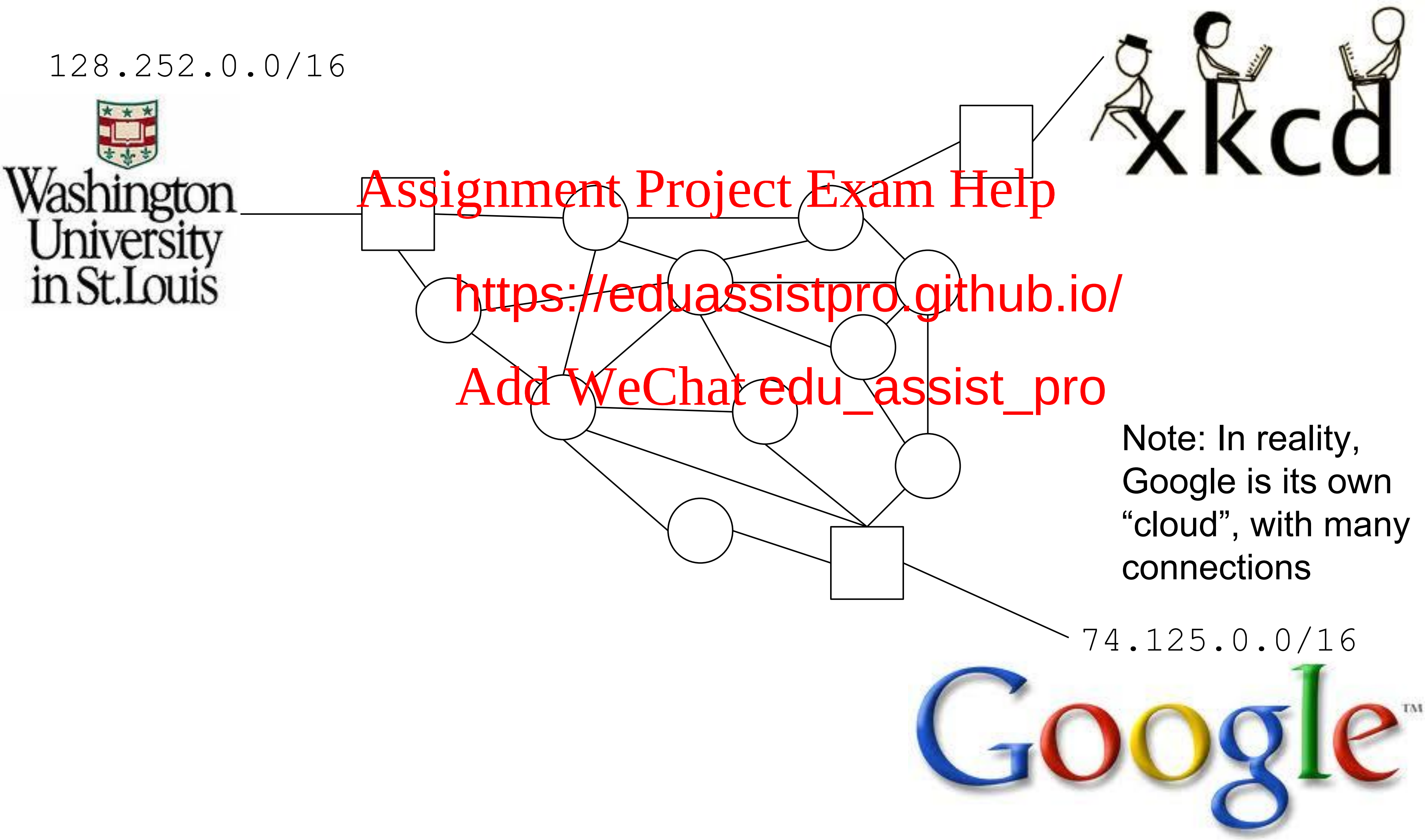
Internet Protocol, IP

- IP allows distinct networks to be connected

- From 30,000 feet
  - Each network is assigned an **IP address range**
    - WU: 128.252.0.0 - 128.252.255.255 (128.252.0.0/16)
  - A dynamic, globally distributed protocol is used to create **routes** betwee
  - A dynamic, glo used to map **domain names** to IP address
  - IP supports multiple protocols nications: UDP, TCP, ICMP, …

- Two aspects of IP to understand
  - Node model
  - Packet format

IP Nodes and Routes



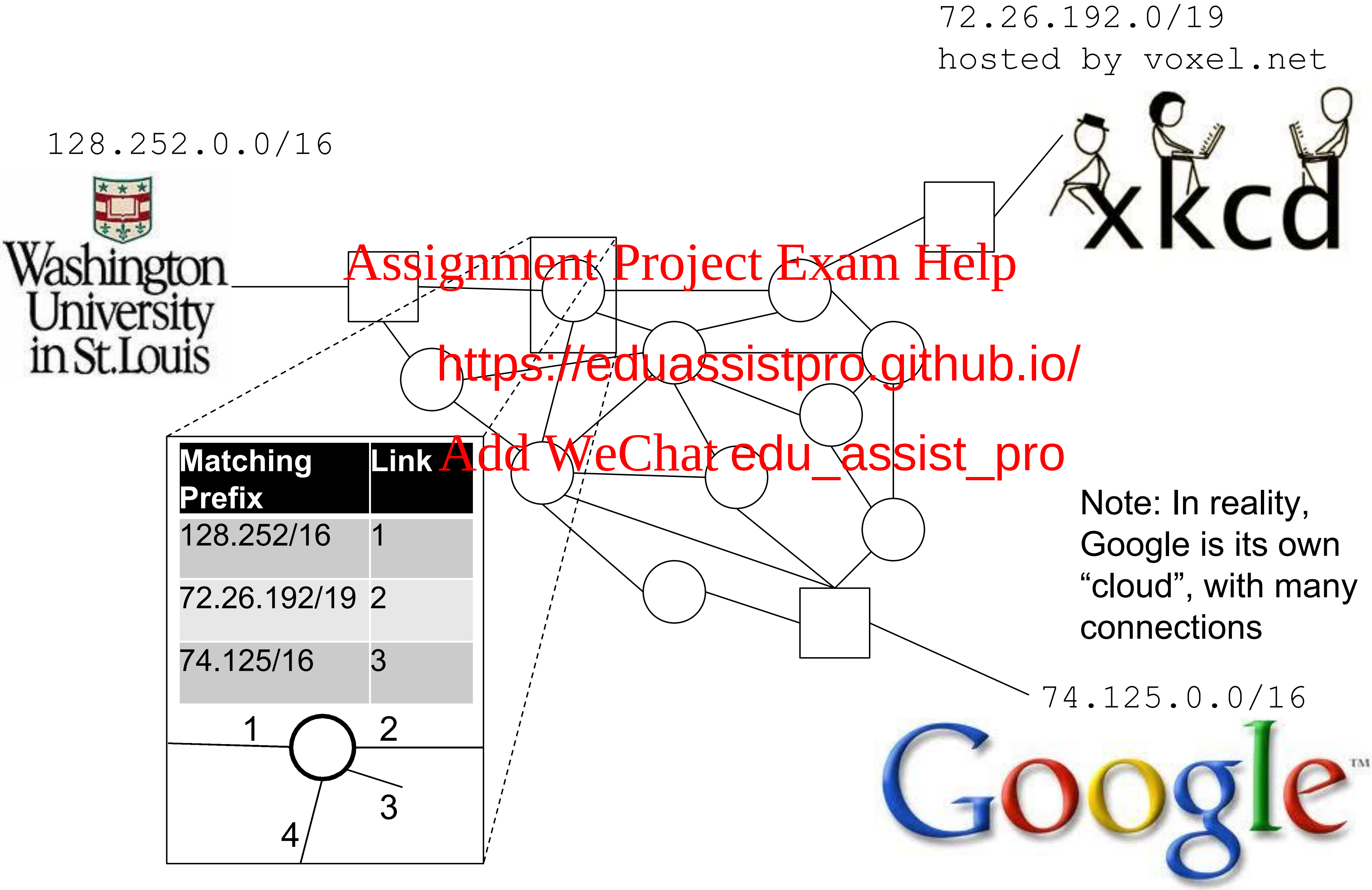
72.26.192.0/19
hosted by voxel.net

128.252.0.0/16

74.125.0.0/16

Note: In reality, Google is its own “cloud”, with many connections

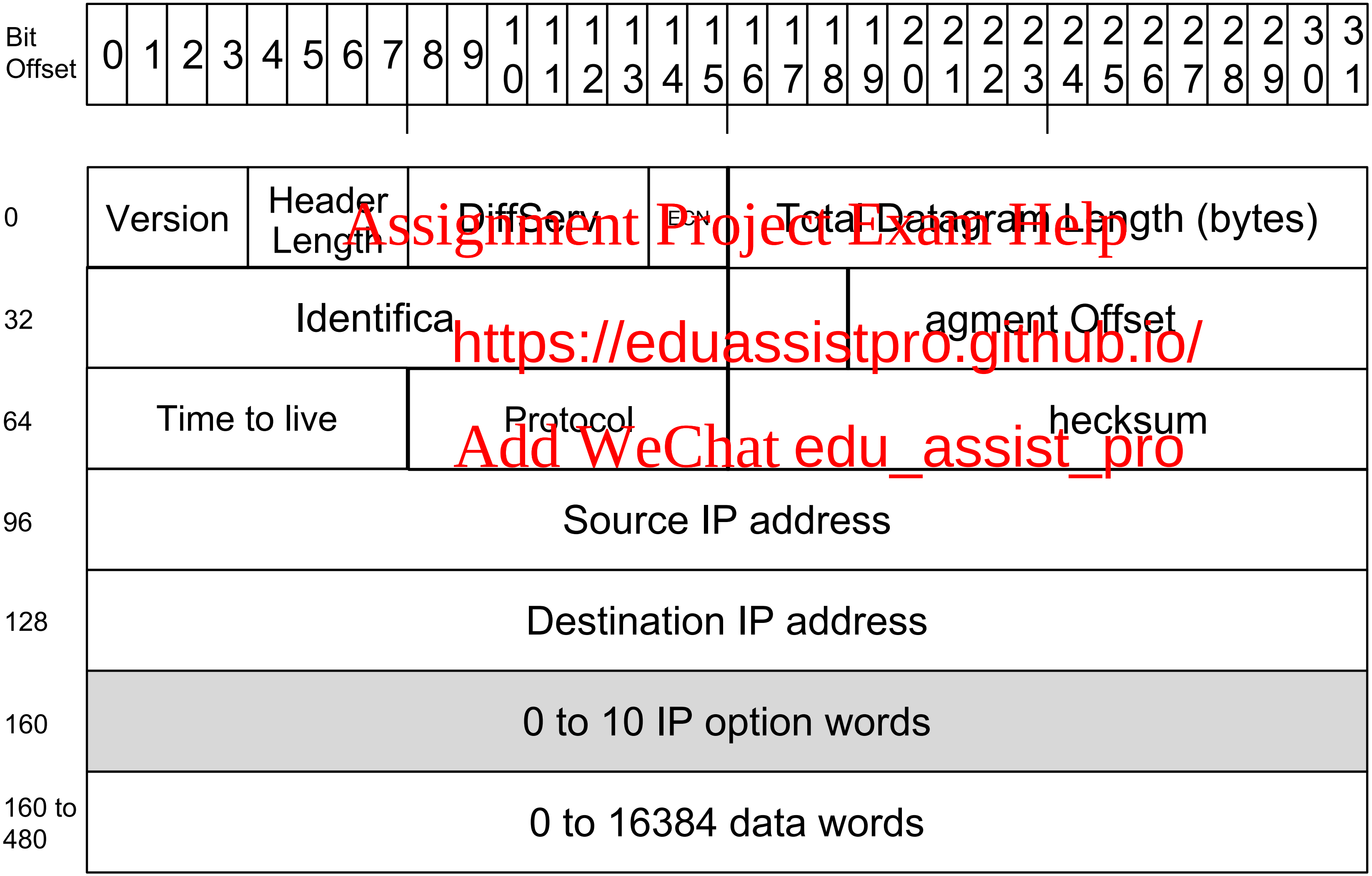
IP Nodes and Routes



Matching Prefix	Link
128.252/16	1
72.26.192/19	2
74.125/16	3

Note: In reality, Google is its own “cloud”, with many connections

IP Packet Format



UDP & TCP

- Two primary protocols for applications
  - UDP: unreliable datagrams
  - TCP: reliable, in-order byte streams

- “Ports” locate
o apps hosts
  - Example in a few slides

User Datagram Protocol, UDP

- Connection-less communications
  - Messages are sent, no in-protocol means for reliability

- Not reliable
  - May not arrive
  - May arrive out of order
  - May be duplicated

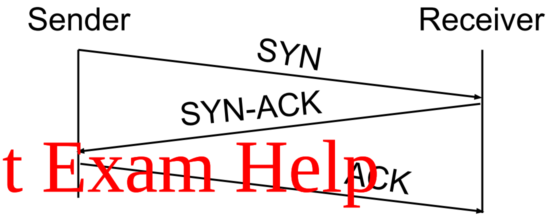
- No support for managing congestion

UDP Packet Format

<table>
<tr><td>Bit Offset</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>0</td><td colspan="16">Source port number (opt)</td><td colspan="16">Destination port number</td></tr>
<tr><td>32</td><td colspan="16">Lengt</td><td colspan="16">checksum (opt)</td></tr>
<tr><td>64</td><td colspan="32">0 to 16376 dat</td></tr>
</table>

Transport Control Protocol, TCP

- Connection-oriented
  - 3-way handshake used between communicating end hosts
    - SYN, SYN-ACK, ACK



treams
- Reliable, order
  - All will arrive
  - Will arrive in order
  - Will not be duplicated

- Includes provision for “congestion control” so that sender-receiver pairs scale up/down their data rates in response to (un)dropped packets.

TCP Packet Format

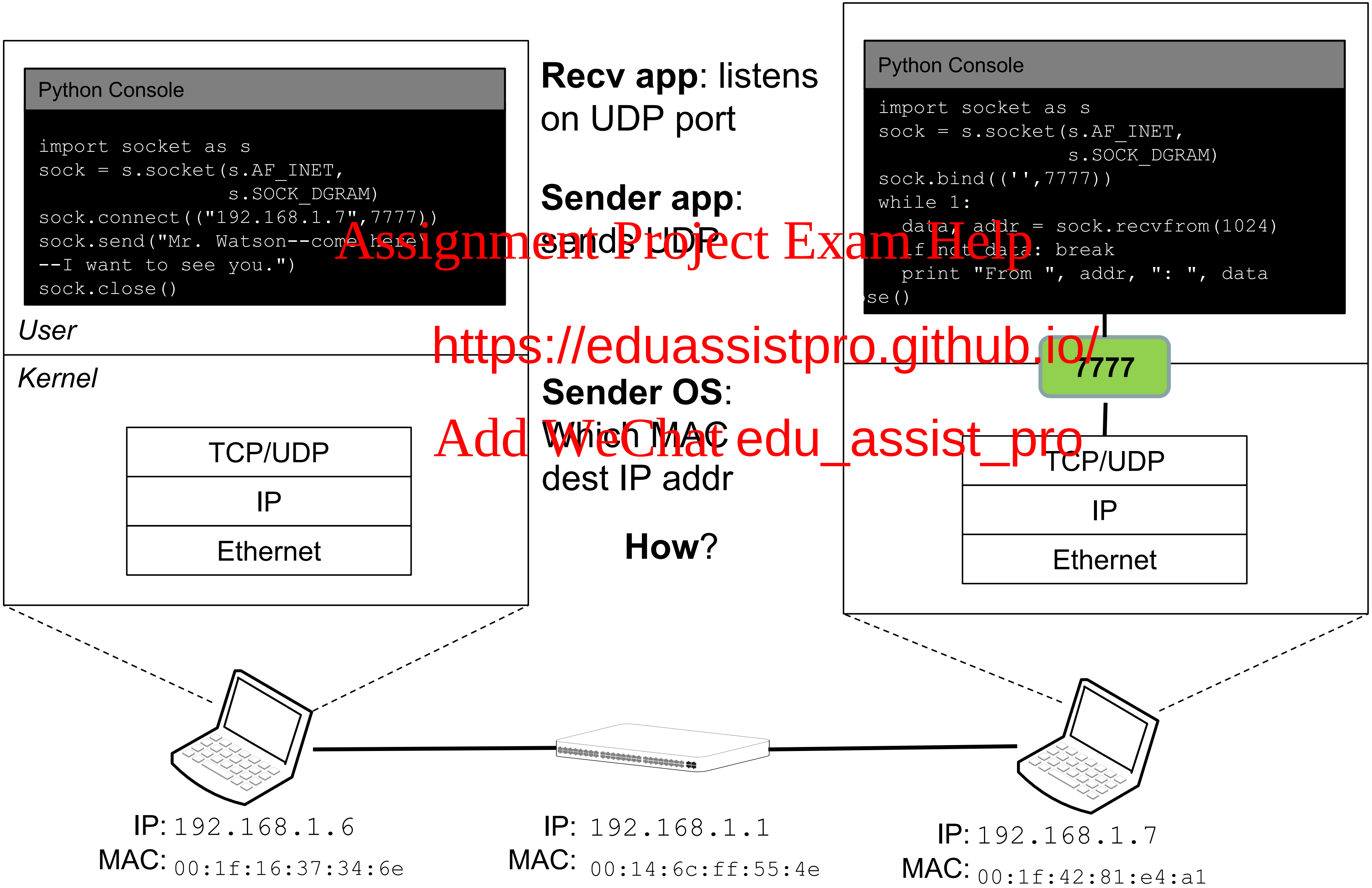
Bit Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port number																Destination port number															
32																																
64	Acknowledgeme																															
96	Data offset				Res			NS	CWR	ECE	URG	**ACK**	PSH	RST	**SYN**	FIN	Window size (bytes)															
128	Checksum																Urgent pointer															
160	Options and data																															

Sockets

- Apps primarily use sockets API to connect
  - Create a socket by specifying address family (AF_INET), and type (SOCK_DGRAM or SOCK_STREAM)
  - Connect it to an address and port
  - Send and receive
  - Library also inc

- Network byte ordering is disti ost byte ordering
  - Little-endian: least significant er address
  - Big-endian: most significant byte at lower address
  - X86: little-endian; network: big-endian
  - Apps must convert to and from network byte order: `ntohl(), htonl()`

Two Machines on an Ethernet LAN



```
import socket as s
sock = s.socket(s.AF_INET,
                s.SOCK_DGRAM)
sock.connect(("192.168.1.7",7777))
sock.send("Mr. Watson--come here
--I want to see you.")
sock.close()
```

```
import socket as s
sock = s.socket(s.AF_INET,
                s.SOCK_DGRAM)
sock.bind(('',7777))
while 1:
  data, addr = sock.recvfrom(1024)
  if not data: break
  print "From ", addr, ": ", data
se()
```

Recv app: listens on UDP port

Sender app: sends UDP

Sender OS: Which MAC dest IP addr

How?

IP: 192.168.1.6
MAC: 00:1f:16:37:34:6e

IP: 192.168.1.1
MAC: 00:14:6c:ff:55:4e

IP: 192.168.1.7
MAC: 00:1f:42:81:e4:a1

Address Resolution, ARP

- General protocol for mapping between protocol layers

- In practice, P address to a p Ethernet MAC

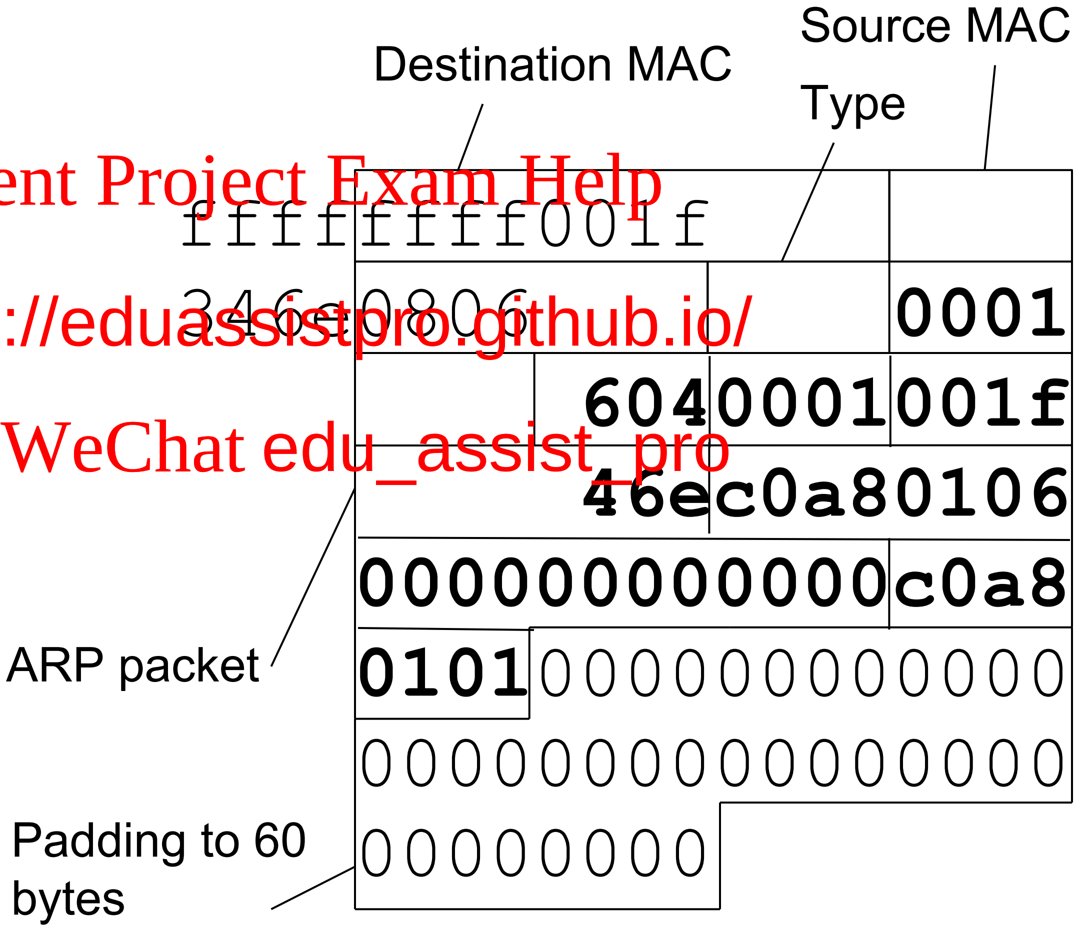
  - Not part of TCP/IP per-se, n't find a network without it

- Two operations
  - Request: Who has <TGT-IP>? Tell <MY-MAC>
  - Reply: <TGT-IP> is at <TGT-MAC>

ARP Ethernet:IP Packet Format

Byte Offset	0	1	2	3
0	Hardware type (Eth is 1)		Protocol type (IP is 0x0800)	
4	HW Addr Len (Eth is 6)	n (1 request, 2 reply)		
8	Sender HW A			
12	SHA, continued		Sender Protocol Address (SPA)	
16	SPA, continued		Target HW Address (THA)	
20	THA, continued			
24	Target Protocol Address (TPA)			

ARP Illustrated Packet



Destination MAC

Source MAC

Type

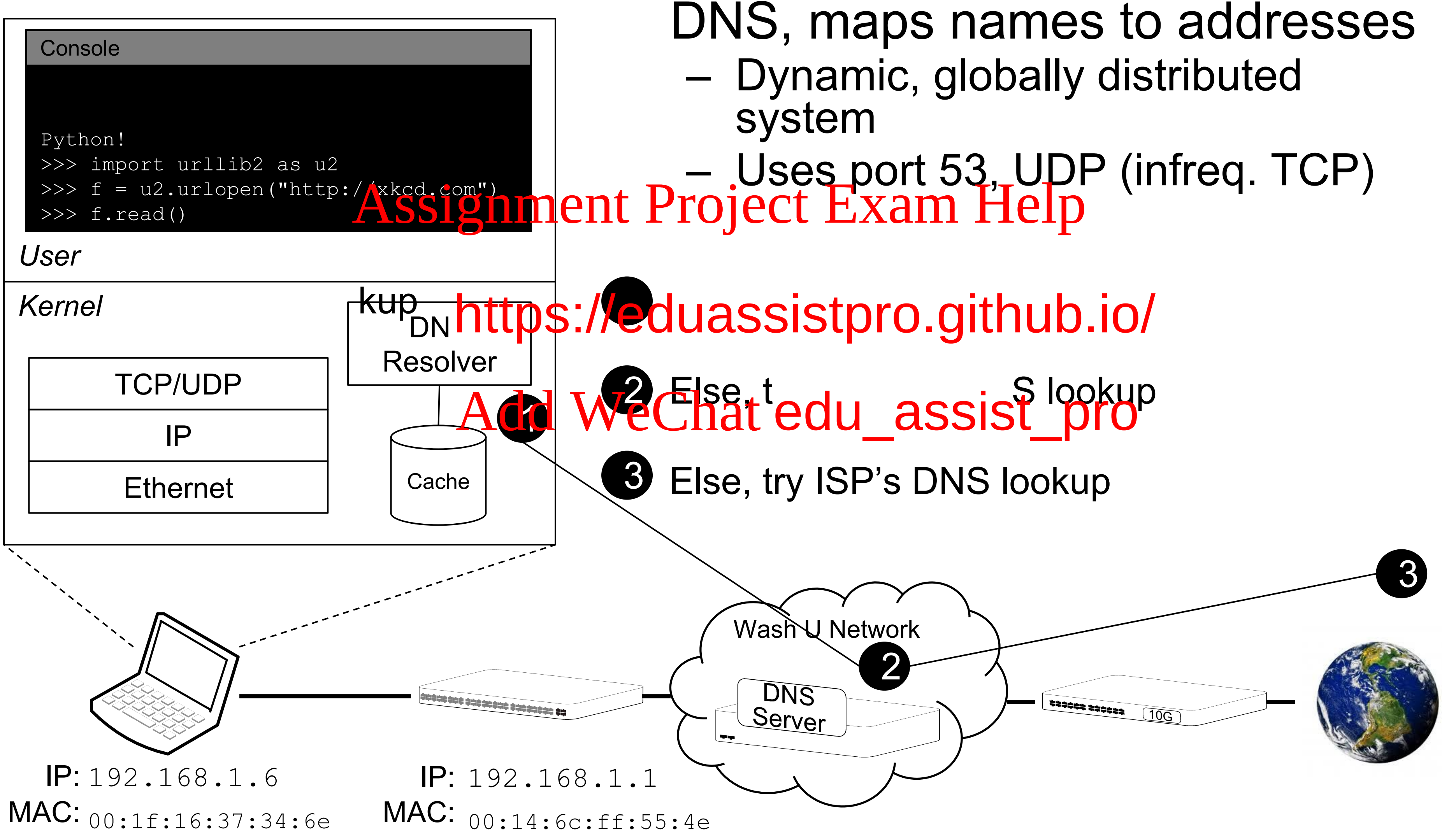
ARP packet

Padding to 60 bytes

```
ffffffffffff001f
346e0806        0001
     604 0001 001f
     46e c0a8 0106
000000000000 c0a8
0101 000000000000
0000000000000000
00000000
```

Internet Names and Addresses

- The Domain Name System, DNS, maps names to addresses
  - Dynamic, globally distributed system
  - Uses port 53, UDP (infreq. TCP)



```
Python!
>>> import urllib2 as u2
>>> f = u2.urlopen("http://xkcd.com")
>>> f.read()
```

kup

2 Else, t S lookup

3 Else, try ISP’s DNS lookup

Other questions to answer

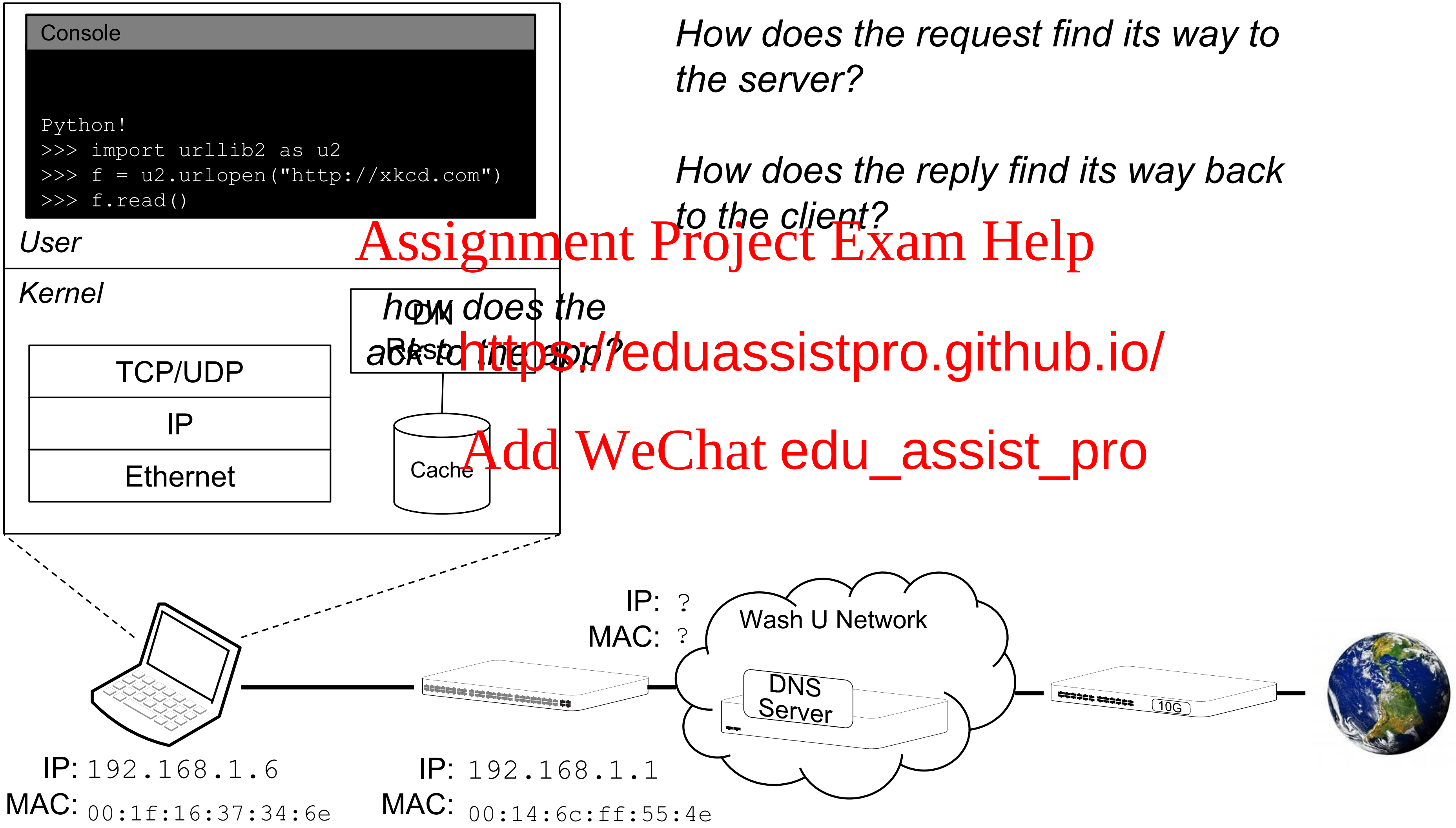
- How do we get a MAC address?
  - Pre-configured or set it yourself

- How do we g

  - Static allocation or via DH

- How do we get to the Internet from within LAN?
  - Default gateway. How do we find it?

Understanding Networks



Console

```
Python!
>>> import urllib2 as u2
>>> f = u2.urlopen("http://xkcd.com")
>>> f.read()
```

How does the request find its way to the server?

How does the reply find its way back to the client?

Issues we will revisit

- Where do protocols assume trust?
  - Are addresses valid?
  - Are gateway
  - Are name:ad

- What can someone else observe?

Helpful Tools

- On your machine
  - wireshark to log and inspect packets
  - host, dig

  es to addresses and

- On the Internet
  - ARIN’s service to name:address mappings and prefix owners
    - https://www.arin.net/

Assignment

- Wednesday
  - HTAOE: Ch. 2 81-114
- Monday
  - hw2 due
  - HTAOE: C