Networking
Data Networks

- A set of interconnected nodes exchange information
- Sharing of the transmission circuits = "switching".
- Links allow more than one path between every 2 nodes.
- Network must select an appropriate path for each required connection.
1961: Kleinrock - queueing theory shows effectiveness of packet-switching

1964: Baran - packet-switching in military nets

1967: ARPAnet conceived by Advanced Research Projects Agency

1969: first ARPAnet node operational

1972:
- ARPAnet public demonstration
- NCP (Network Control Protocol) first host-host protocol
- First e-mail program
- ARPAnet has 15 nodes
Internet History

1972-1980: Internetworking, new and proprietary nets

- **1970**: ALOHA.net satellite network in Hawaii
- **1974**: Cerf and Kahn - architecture for interconnecting networks
- **1976**: Ethernet at Xerox PARC
- **late 70's**: proprietary architectures: DECnet, SNA, XNA
- **late 70's**: switching fixed length packets (ATM precursor)
- **1979**: ARPAnet has 200 nodes

Cerf & Kahn’s internetworking principles

- minimalism, **autonomy** - no internal changes required to interconnect networks
- best effort service model
- stateless routers
- decentralized control
define today’s Internet architecture
Internet History

1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: smtp e-mail protocol
- 1983: DNS defined for name-to-IP-address translation
- 1985: ftp protocol defined
- 1988: TCP congestion control
- 100,000 hosts connected to confederation of networks
Internet History

1990-2000s: commercialization, the Web, new apps

- Early 1990s: ARPAnet decommissioned
- early 1990s: Web
  - hypertext [Bush 1945, Nelson 1960’s]
  - HTML, HTTP: Berners-Lee
  - 1994: Mosaic, later Netscape
  - late 1990’s: commercialization of the Web

- More killer applications
  - instant messaging, P2P file sharing
- Network security to forefront
- 50 million host, 100 million+ users
Internet Architecture
Air travel organized into layers

- layers: each layer implements a service
  - via its own internal-layer actions
  - relying on services provided by layer below
Layering manages complexity
dealing with complex systems:

❖ explicit structure allows identification, relationship of complex system’s pieces
  layered reference model for discussion
❖ modularization eases maintenance, updating of system
  change of implementation of layer’s service transparent to rest of system
  e.g., change in gate procedure doesn’t affect rest of system
❖ layering considered harmful?
Internet layers - OSI layer model
Intermediate nodes may vary in the number of layers they process, but the Internet is at layer 3

hub (1) vs. switch (2) vs. router (3) vs. proxy sever (5)
Internet Packet Encapsulation

Application Message

Application Layer
Internet Packet Encapsulation

Application Layer

Transport Layer

Application Message

Segment Header

Segment Data
Internet Packet Encapsulation

Application Layer

Transport Layer

Network Layer
Internet Packet Encapsulation

What you care about

Application Message

Segment Header  Segment Data

Packet Header  Packet Data

Frame Header  Frame Data  Frame Footer

Application Layer
Transport Layer
Network Layer
Link Layer
Internet Packet Encapsulation

What you care about

What gets sent

Application Layer
Transport Layer
Network Layer
Link Layer
Internet protocol stack

- **application**: supporting network applications
  - FTP, SMTP, HTTP
- **transport**: process-process data transfer
  - TCP, UDP
- **network**: routing datagrams
  - IP, routing protocols
- **link**: data transfer between neighboring network elements
  - PPP, Ethernet
- **physical**: bits “on the wire”
Link layer
**Link layer: introduction**

- **Terminology:**
  - hosts and routers: *nodes*
  - communication channels that connect adjacent nodes along communication path: *links*
    - wired links
    - wireless links
    - LANs
  - layer-2 packet: frame, encapsulates datagram

*data-link layer* has responsibility of transferring datagram from one node to **physically adjacent** node over a link
MAC Addresses

- Most network interfaces come with a predefined MAC address

- A MAC address is a 48-bit number usually represented in hex
  - E.g., 00-1A-92-D4-BF-86

- MAC address can be reconfigured by network interface driver software
  - This makes MAC address filtering insecure
Ethernet

- Dominant wired LAN technology:
- First widely used LAN technology
- Simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 100 Gbps

<table>
<thead>
<tr>
<th>preamble</th>
<th>source address</th>
<th>data (payload)</th>
<th>CRC</th>
</tr>
</thead>
</table>

**Ethernet frame**

Metcalfe’s Ethernet sketch
Where is the link layer implemented?

- In each and every host
- Link layer implemented in “adaptor” (network interface card NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- Attaches into host’s system buses
- Combination of hardware and firmware
Network layer
Network layer

- Transport segment from sending to receiving host
  - sending side encapsulates segments into datagrams
  - receiving side delivers segments to transport layer

- Network layer protocols in every host, router

- Router examines the header of IP datagrams passing through it
Two key network-layer functions

- **Forwarding:**
  - move packets from router’s input to appropriate router output

- **Routing:**
  - determine route taken by packets from source to destination
  - Routing algorithms

- **Analogy:**
  - routing: process of planning trip from source to dest
  - forwarding: process of getting through single interchange
Interplay between routing and forwarding

The routing algorithm determines the end-end-path through the network. The forwarding table determines the local forwarding at this router. The header value in the arriving packet's header is used to determine the output link.

<table>
<thead>
<tr>
<th>Header Value</th>
<th>Output Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

In the diagram, the routing algorithm is linked to the local forwarding table, which in turn is linked to the forwarding table and the output links. The value in the arriving packet's header determines the output link, which is connected to the network path.
The Internet network layer

host, router network layer functions:

- **Routing protocols**: path selection, RIP, OSPF, BGP
- **IP protocol**: addressing conventions, datagram format, packet handling conventions
- **ICMP protocol**: error reporting, router “signaling”
**IP Addressing**

- IP address used to route datagrams through Internet.
  - IPv4 32 bit address, IPv6 128 bit address

- Usually represented in dotted decimal notation: 141.211.144.212
- Each number represents 8 bits: 0-255.
IP packet layout

- **IP header includes**
  - Source address
  - Destination address
  - Packet length (up to 64KB)
  - Time to live (up to 255)
  - IP protocol version
  - Fragmentation information
  - Transport layer protocol information (e.g., TCP)
**ARP protocol: Same LAN**

- A wants to send datagram to B
  - B’s MAC address not in A’s ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF-FF
- All nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
- Frame sent to A’s MAC address (unicast)
Transport layer
Transport services and protocols

- Provide logical communication between app processes running on different hosts
- Transport protocols run in end systems
  - send: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP
A tale of two transport services

TCP service:

- **Reliable** transport between sending and receiving process
- **flow control**: sender won’t overwhelm receiver
- **congestion control**: throttle sender when network overloaded
- **does not provide**: timing, throughput guarantee, **security**
- **connection-oriented**: setup required between client and server processes
A tale of two transport services

UDP service:
- **Unreliable** data transfer between sending and receiving process
- **does not provide:** reliability, flow control, congestion control, timing, throughput guarantee, **security**, or connection setup
A tale of two transport services

UDP service:

- **Unreliable** data transfer between sending and receiving process
- **does not provide**: reliability, flow control, congestion control, timing, throughput guarantees, security, or connection setup
What transport service does an app need?

**Data integrity**
- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

**Throughput**
- some apps (e.g., multimedia) require minimum amount of throughput to be “effective”
- other apps (“elastic apps”) make use of whatever throughput they get

**Latency**
- some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

**Security**
- encryption, data integrity, confidentiality, …
Application Layer
App-layer protocol defines

- Types of messages exchanged
  - e.g., request, response

- Message syntax:
  - what fields in messages & how fields are delineated

- Message semantics
  - meaning of information in fields

- Rules for when and how processes send & respond to messages
Application Layer: internet apps

- E-mail
- Web
- Instant Messaging
- Remote Login
- P2P File Sharing
- Multi-user Network Games
- Video Streaming

- Voice Over IP
- Real-time Video
- Grid Computing
- Online Social Networks
**Client-server architecture**

**Server:**
- always-on host
- permanent IP address
- server farms for scaling

**Clients:**
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other
Pure P2P architecture

- No always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

Highly scalable
but difficult to manage
Sockets

- Process sends/receives messages to/from its socket
- socket analogous to door
  - Sending process shoves message out door
  - Receiving process relies on transport infrastructure to bring message to socket
Addressing Processes

- To receive messages, process must have *identifier*
- **Q:** does IP address of host suffice for identifying the process?
  - A host have many processes
  - *identifier* includes both IP address and port numbers
  - Example port numbers:
    - HTTP server: 80
    - Mail server: 25
  - To send HTTP message to cs.vt.edu web server:
    - IP address: 198.82.184.178
    - Port number: 80
HTTP & Web
Web and HTTP

- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio files
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL

Example URL:

www.someschool.edu/someDept/pic.gif

- **host name**
- **path name**
HTTP overview

HTTP: hypertext transfer protocol
- Application layer protocol
- client/server model
  - **client**: browser that requests, receives, “displays” Web objects
  - **server**: web server sends objects in response to requests
HTTP overview (continued)

Uses TCP:

- Client initiates TCP connection (creates socket) to server port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is “stateless”

- Server maintains no information about past client requests

Protocols that maintain “state” are complex!

- past history (state) must be maintained
- if server/client crashes, their views of “state” may be inconsistent, must be reconciled
HTTP request message

- two types of HTTP messages: request, response
- HTTP request message: ASCII (human-readable format)

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Carriage return, line feed indicates end of message

(extra carriage return, line feed)
Uploading form input

Post method:
- Web page often includes form input
- Input is uploaded to server in entity body

URL method:
- Uses GET method
- Input is uploaded in URL field of request line

Example of URL method
www.somesite.comanimal-search?monkeys&banana
HTTP response message

status line
(protocol status code
status phrase)

HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 ...
Content-Length: 6821
Content-Type: text/html

data, e.g., requested HTML file

data data data data data data ...

header lines
HTTP response status codes

In first line in server->client response message.

200 OK
- request succeeded, requested object later in this message

301 Moved Permanently
- requested object moved, new location specified later in this message (Location)

400 Bad Request
- request message not understood by server

404 Not Found
- requested document not found on this server

505 HTTP Version Not Supported
I’M BACK
HTTP overview (continued)

Uses TCP:

- Client initiates TCP connection (creates socket) to server port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is “stateless”
- Server maintains no information about past client requests

Protocols that maintain “state” are complex!
- past history (state) must be maintained
- if server/client crashes, their views of “state” may be inconsistent, must be reconciled
User-server state: cookies

Four components:

1) cookie header line of HTTP response message
2) cookie header line in HTTP request message
3) cookie file kept on user’s host (user browser)
4) back-end database at Web site

Example:

- Susan always access Internet always from PC
- Visits specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates
  - unique ID
  - entry in backend database for ID
Cookies: keeping “state” (cont.)

Client

- eBay 8734
- Cookie file
- eBay 8734 Amazon 1678

Server

- Amazon server creates ID 1678 for user
- Backend database

One week later:

- eBay 8734 Amazon 1678
- Cookie file

Usual HTTP request msg

- Cookie: 1678
- Cookie: 1678

Usual HTTP response msg

Access

Specific action
Cookies (continued)

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state

How to keep “state”:
- Maintain state at sender/receiver over multiple transactions
- Cookies: http messages carry state

Cookies and privacy:
- cookies permit sites to learn about you
- you may supply name and e-mail
DNS

Slides credit to J.F Kurose and K.W. Ross
DNS: Domain Name System

- Internet hosts, routers:
  - IP address (32 bit) - used for addressing datagrams
  - “name”, e.g., ww.yahoo.com - used by humans

- Q: map between IP addresses and name?

- Domain Name System
  - Distributed database implemented in hierarchy of many name servers
  - Application-layer protocol to resolve names
  - Core Internet function
DNS services

- Hostname -> IP translation
- Host aliasing
- Mail server aliasing
- Load distribution
  - replicated web servers
  - set of IP addresses for one canonical name
  - E.g., amazon.com

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- ...
- Doesn’t scale!
TLD and Authoritative Servers

- **Top-level domain (TLD) servers:**
  - Responsible for com, org, net, edu, etc
  - Responsible all top-level country domains uk, fr, ca, jp.
  - Network solutions maintains servers for com TLD
  - Education for edu TLD

- **Authoritative DNS servers:**
  - Organization’s DNS servers, providing authoritative hostname to IP mappings for organization’s servers (e.g., Web, mail).
  - Can be maintained by organization or service provider
Client wants IP for www.amazon.com; 1st approx:

- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS
- client queries amazon.com DNS server to get IP address for www.amazon.com
DNS: Root name servers

- Contacted by local name server that can not resolve name
- Root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping, returns mapping to local name server

13 root name servers worldwide

(more than 13 physical servers for reliability and redundancy)
Local Name Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
  - also called “default name server”
- When host makes DNS query, query is sent to local DNS server
  - acts as proxy, forwards query into hierarchy
- If a host name cannot be resolved by local name server, then forward the query to upper-level name server
DNS name resolution example

- Host at me.cs.vt.edu
- Wants IP address for gaia.cs.umass.edu

**Iterated Query:**
- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”
DNS name resolution

Recursive Query:
- puts burden of name resolution on contacted name server
- heavy load?
DNS: caching and updating records

- Once (any) name server learns mapping, it *caches* mapping
  - Cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited

- update/notify mechanisms under design by IETF
  - RFC 2136
The “dig” and “nslookup” programs provide an interface to DNS

nslookup cs.vt.edu

Server: 192.168.2.1
Address: 192.168.2.1#53

Non-authoritative answer:
Name: cs.vt.edu
Address: 198.82.184.178
Reverse DNS

- We have the IP address, but want the name
- Use DNS to perform the lookup function

- Special domain, “in-addr.arpa” domain for reverse lookups
  - Internet address is reversed in the lookup

nslookup 198.82.184.178

178.184.82.198.in-addr.arpa name = cs.vt.edu
E-mail (Demo)
E-Mail

Three major components:

- user agents
- mail servers
- mail transfer protocol: SMTP

User Agent

- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g. Outlook,
- outgoing, incoming messages stored on server
E-Mail: mail servers

Mail Servers

- **mailbox** contains incoming messages for user
- **message queue** of outgoing (to be sent) mail messages
- **SMTP protocol** between mail servers to send email messages
  - client: sending mail server
  - server: receiving mail server
E-Mail: SMTP [RFC 2821]

- Uses TCP to reliably transfer email message from client to server,
- Port 25
- Direct transfer: sending server to receiving server
- Three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- Command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII
**Scenario: Alice sends message to Bob**

1) Alice uses UA to compose message and “to” bob@hamburger.edu

2) Alice’s UA sends message to her mail server (e.g. crepes.fr); msg placed in message queue

3) Client side of SMTP opens TCP connection with Bob’s mail server

4) SMTP client sends Alice’s message over the TCP connection

5) Bob’s mail server places the message in Bob’s mailbox

6) Bob invokes his user agent to read message
Sample SMTP interaction

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
Try SMTP interaction for yourself

- telnet smtp.gmail.com 587
  - Why not port 25?

- Helo google
- see 220 reply from server

- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands
- Model SMTP server usually require TLS to encrypt traffic
Mail access protocols

- Mail access protocol: retrieval from server
  - **POP**: Post Office Protocol [RFC 1939]
    - authorization (agent <---> server) and download
  - **IMAP**: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
  - **HTTP(s)**: gmail, Hotmail, Yahoo! Mail, etc.
Email Spoofing

- SMTP has no authentication
  - “MAIL FROM” can be set to anything (e.g., a spoofing target)
  - “From” fields in header can be changed to anything → visible to users

- Widely used for spear phishing
Example

Urgent! Response needed immediately.

visa@uscis.gov
To uitest12767@yahoo.com

Dear Mr. Hu,

This email is to inform you that your VISA is going to expire soon. If you don't renew it in 48 hours, your status will be illegal. 
Click Here to renew your VISA.

Thank you,
The United States Citizenship and Immigration Services (USCIS)
Example

Urgent! Response needed

To uittest12767@yahoo.com

Dear Mr. Hu,

This email is to inform you that your current VISA status will be illegal. Click Here to renew your VISA.

Thank you,
The United States Citizenship and Immigration Services (USCIS)
**Anti-Spoofing: SMTP extensions**

- **SPF**: authentication by IP
  - Proposed in 2001, standardized in 2014
  - DNS: specifies the IP range that can send email on behalf of x.com

- **DKIM**: public key based method
  - Drafted in 2004, standardized in 2011
  - Sender domain signs the email

- **DMARC**:
  - Drafted in 2011, standardized in 2015
  - Complementary to SPF and DKIM (addressed the identifier alignment issue)
  - Specific “what to do” when authentication fails
Anti-Spoofing Protocols: SPF/DKIM

- Verifying sender identity, email integrity
  - SPF (Sender Policy Framework)
  - DKIM (DomainKeys Identified Mail)

Example: SPF

Sender
vt.edu
IP: 1.1.1.1

Attacker
evil.com
IP: 2.2.2.2

Receiver
yahoo.com
IP: 3.3.3.3

Mail FROM: “Alice”
<alice@vt.edu>

Check:
IP 1.1.1.1
sends <vt.edu>?

Check:
IP 2.2.2.2
sends <vt.edu>?

SPF Record
IP 1.1.1.1 <vt.edu>

Telegram icon
DNS Server

84
Anti-Spoofing Protocols Not Widely Adopted?

- Measuring SPF adoption in January 2017
  - Scanned 1,000,000 most popular web domains
  - Queried their SPF records in DNS

- 47.8% domains adopted SPF to prevent spoofing
- By the way, DMARC: 4.5%

*Ranked by Alexa.com
Anti-Spoofing protocols: Technical Weaknesses

- **SPF**: authentication by IP
  - DNS: specifies the IP range that can send email on behalf of *x.com*
  - Mail forwarding (why?); identifier alignment

- **DKIM**: public key based method
  - Sender domain signs the email

- **DMARC**:
  - Still not widely used
  - Weakly configured
  - Cannot handle mailing lists
Example: identifier alignment?

<table>
<thead>
<tr>
<th>Domain used for SPF test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return-Path: <a href="mailto:attacker@attacker.com">attacker@attacker.com</a></td>
</tr>
<tr>
<td>Received-SPF: pass (domain of attacker.com designates 123.456.789.00 as permitted sender)</td>
</tr>
<tr>
<td>Subject: Urgent! Response needed immediately.</td>
</tr>
<tr>
<td>From: <a href="mailto:visa@uscis.gov">visa@uscis.gov</a></td>
</tr>
<tr>
<td>To: <a href="mailto:uitest12767@yahoo.com">uitest12767@yahoo.com</a></td>
</tr>
</tbody>
</table>

The email sender domain that user sees is different from the one that is actually used to perform SPF authentication.
SPF Weakness: Mail Forwarding

- I forward all my emails received by [vt.edu] to my [Gmail] inbox
  - [Outlook]?
  - [Yahoo Mail]?

- Problem: breaking SPF

- From the email receiver perspective (Gmail):
  - The email is sent by vt.edu
  - The IP and sender domain no longer matches → SPF fails
Anti-Spoofing protocols: Technical Weaknesses

- **SPF**: authentication by IP
  - DNS: specifies the IP range that can send email on behalf of x.com
  - Mail forwarding (why?); identifier alignment

- **DKIM**: public key based method
  - Sender domain signs the email header and content
  - Mailing list, or legitimate content change; identifier alignment

- **DMARC**:
  - Still not widely used
  - Weakly configured
  - Cannot handle mailing lists
DKIM Weaknesses: Identifier Alignment

The email sender domain that user sees is different from the one that is actually used to perform DKIM authentication.

Domain that signs the message

DKIM-Signature: v=1; a=rsa-sha256; d=attacker.com; s=pp-dkim1; c=relaxed/relaxed; q=dns/txt; i=@attacker.com; t=1480474251;
  h=From:From:Subject:Date:To:MIME-Version:Content-Type;
  bh=...; b=...

Subject: Urgent! Response needed immediately.

From: visa@uscis.gov
To: uitest12767@yahoo.com

Domain displayed to users
Email Providers cannot (fully) detect Spoofing

Security indicator?
- Unknown usability
- Do people understand?

The meeting will be on Friday

Gang Wang <gangwang@vt.edu>

This sender failed our fraud detection checks and may not be who they appear to be.