Internet Technology and Web Services

1. Introduction: The Evolution of Internet and Web Technology

This lecture explores the foundations of Internet technology, focusing on the problems and solutions of packet-switched networking, the layered protocol model, and the development of the World Wide Web. We cover the technical and historical context, including protocol design, the TCP/IP stack, and the evolution of web services such as HTTP, HTTPS, and HTML.

2. Fundamental Concepts of Networking

2.1 Circuit Switching vs. Packet Switching

- **Circuit Switching**: Establishes a dedicated path between endpoints (e.g., traditional telephony). Predictable but inefficient for bursty or variable traffic.
- **Packet Switching**: Data is sent in small units (packets) that may take different paths to the destination. Used in the Internet. More efficient and robust, but introduces new challenges.

Analogy:

- Circuit switching is like reserving an entire lane on a highway for one car's trip.
- Packet switching is like sending many cars (packets) that can take any available lane and route to reach the destination.

ASCII Diagram:

```
Circuit Switching:
A =======[Dedicated Path]====== B

Packet Switching:
A ==\    /== Router ==\    /== B
      ==/             ==/
(Packets can take different routes)
```

Feature	Circuit Switching	Packet Switching
Path setup	Required	Not required
Reliability	More predictable	Less predictable
Efficiency	Low (fixed path)	High (dynamic routing)
Fault Tolerance	Low	High
Resource Sharing	Poor	Excellent

2.2 What is a Packet?

- A packet is a small unit of data (typically ~1 KiB) sent over the network.
- **Header**: Contains control information (destination, protocol type, etc.), used by routers and the network.
- **Payload**: Contains application data, used by the recipient application.
- The division between header and payload is sometimes a judgment call.

Example Packet Structure:

```
+-------------------+-----------------------+
|      Header       |       Payload         |
+-------------------+-----------------------+
| Src IP | Dst IP | ... |  Data (e.g., file, email) |
+-------------------+-----------------------+
```

Field	Purpose
Header	Routing, protocol type, TO/FROM
Payload	Email text, file content, etc.

3. Protocols and Packet Switching Challenges

3.1 What is a Protocol?

- A protocol is an agreed-upon set of rules for message transmission and processing.
- Analogy: Diplomatic protocols—if you don't follow the rules, you get ignored.
- Protocols specify both message formats (headers, fields) and behaviors (how to respond to messages, what to do on error, etc.).
- Protocols are written as specifications (specs), not code.

Real-World Analogy:

- Like the postal system: envelopes (headers) carry letters (payloads), and everyone agrees on addressing and delivery rules.

3.2 Challenges in Packet Switching

Packet switching introduces several fundamental problems:

1. **Packet Loss**
    - Packets can be silently dropped, usually due to router buffer overflow or network congestion.
    - Loss rates can range from 1% to 50%+ under heavy load.
2. **Packets Received Out of Order**
    - Packets may arrive in a different order than sent, due to multiple network paths.
    - Can be problematic for applications requiring strict order (e.g., file uploads).
3. **Packet Duplication**
    - Rare, but possible due to network misconfigurations (e.g., routers misconfigured as bridges).
4. **Packet Corruption**
    - Bit errors from unreliable hardware or faulty links.
    - Requires checksums or error-detection mechanisms.

---

4. Layered Network Architecture

To manage complexity, protocols are organized into layers, each providing a different abstraction.

Layer	Functionality	Example Protocols
Application	High-level data exchange (e.g., HTTP, FTP, SMTP, HTML)	HTTP, FTP, SMTP, DNS
Transport	Reliable or fast delivery (e.g., TCP, UDP)	TCP, UDP
Internet	Routing across the network (e.g., IP)	IP (IPv4, IPv6), ICMP
Link	Data link between adjacent network nodes (e.g., Ethernet)	Ethernet, Wi-Fi, ARP

- Other models (e.g., OSI) define 7 layers, but the 4-layer model is most common in practice.
- Each layer adds its own header to the data (encapsulation).

ASCII Diagram: Layered Model

```
+--------------------+
|   Application      |
+--------------------+
|    Transport       |
+--------------------+
|    Internet        |
+--------------------+
|      Link          |
+--------------------+
```

Encapsulation Example:

```
Application Data
   ↓
[Application Header | Data]
   ↓
[Transport Header | Application Header | Data]
   ↓
[Internet Header | Transport Header | Application Header | Data]
   ↓
[Link Header | Internet Header | Transport Header | Application Header | Data]
```

4.1 The Link Layer

- Handles communication over a single, direct link.
- Protocols are hardware-specific (Ethernet, Wi-Fi, USB).
- Only cares about local addresses (e.g., MAC address).

Technology	Description
Ethernet	Wired LAN standard
Wi-Fi	Wireless communication
USB	Peripheral connections

4.2 The Internet Layer

- Main protocol: IP (IPv4 or IPv6).
- Responsible for global addressing and routing.
- Stateless, best-effort delivery (no guarantees).
- Introduces problems like loss, reordering, duplication.
- Introduces IP addresses (32-bit for IPv4, 128-bit for IPv6).

IP Address Breakdown Example:

```
IPv4: 192.168.1.42
Binary: 11000000.10101000.00000001.00101010
         |   192   |   168   |    1    |   42   |
```

- **Subnetting:** Divides address space for efficient routing and security (e.g., 192.168.1.0/24).
- **Private IPs:** Reserved for local networks (e.g., 10.0.0.0/8, 192.168.0.0/16).

4.3 The Transport Layer

- Provides reliable (or fast) delivery over the unreliable Internet layer.
- Introduces data channels (logical streams), reliability, ordering, error correction.
- Protocols: TCP (reliable), UDP (unreliable, fast).
- Solves the three main packet problems (loss, order, duplication).

Ports:

- Allow multiple services to run on one host (e.g., web server on port 80, SSH on port 22).

ASCII Diagram: Ports on a Host

```
[Host]
 |-- Port 22: SSH
 |-- Port 80: HTTP
 |-- Port 443: HTTPS
 |-- Port 25: SMTP
```

4.4 The Application Layer

- Protocols for specific services (HTTP, FTP, SMTP, DNS, etc.).
- Focuses on user/application concerns.

---

5. Internet Protocol (IP)

5.1 IPv4 Overview

- Introduced in 1983 (John Postel, UCLA alum).
- Connectionless: just delivers packets, no guarantee of delivery or order.

5.2 IPv4 Header Fields

Field	Description
Length	Packet size in bytes
Protocol Number	Defines encapsulated protocol (e.g., TCP = 6; UDP = 17)

Field	Description
Source IP	32-bit address of sender
Destination IP	32-bit address of recipient
TTL	Time-to-live (prevents infinite loops; decremented at each router hop)
Checksum	16-bit checksum for packet corruption detection

- IP addresses are written as four decimal numbers (e.g., 192.168.0.1), each representing 8 bits.
- TTL (Time To Live) prevents packets from looping forever.
- Checksum is for accidental corruption, not security.

5.3 IPv4 vs. IPv6

Version	Address Size	Release Year	Supports
IPv4	32-bit	1983	~4.3B addresses
IPv6	128-bit	1998	2^128 addresses

- IPv4: 32-bit address space (~4.3 billion addresses), now exhausted.
- IPv6: 128-bit address space, more header fields, future-proofing.
- IPv6 adoption: ~40% (IPv4 still ~60%) due to inertia and legacy systems.
- IPv6 is required in regions with insufficient IPv4 addresses (e.g., China).

5.4 Checksums and Reliability

- 16-bit checksum for error detection (not cryptographic).
- End-to-End Principle: Each layer should do its own error checking, not trust lower layers.
- Not sufficient for malicious actors, only for accidental errors.

End-to-End Principle Example:

- If you download a file, the application (e.g., browser) should verify the file's integrity (e.g., with a hash), not just trust the network.

---

6. Transport Layer Protocols

6.1 User Datagram Protocol (UDP)

- Designed by David Reed (MIT, 1980s).
- Very thin layer over IP; adds source/destination port numbers.
- Unreliable, unordered, no guarantees—application must handle loss, order, duplication.
- Used for streaming, DNS, telemetry, IoT (where speed > reliability).
- Example: IoT thermometer sends a UDP packet every 10 minutes; if lost, no big deal.

6.2 Transmission Control Protocol (TCP)

- Designed by Vint Cerf (UCLA) and Bob Kahn (Princeton).
- Reliable, connection-oriented, stream-based.
- Guarantees:
    - **Reliability**: Retransmits lost packets.
    - **Ordering**: Reassembles out-of-order packets.
    - **Error Checking**: End-to-end validation.
    - **Flow Control**: Adjusts sending rate to avoid network overload.
- Used for web traffic, file transfers, assignment submissions, etc.

Challenge	TCP Feature That Solves It	Description
Loss	Retransmission	Re-sends lost data
Out-of-Order	Reassembly	Orders packets before delivering to application
Duplication	Sequence numbers, ACK	Identifies and discards redundant data
Overflow	Flow Control	Sender adjusts speed based on network conditions

- Retransmission: Sender keeps packets in RAM until acknowledged.
- Reassembly: Recipient buffers and reorders packets before delivering to application.
- Flow control: Prevents flooding the network.

TCP Three-Way Handshake:

```
Client           Server
  | SYN  ------> |
  |     <------ SYN-ACK
  | ACK  ------> |
Connection established
```

---

7. Protocol Specification: How to Create a Protocol

- Protocols are written as specifications (not code).
- Must define:
    - Packet formats and headers (field order, required fields)
    - Expected behaviors (how to respond to valid/invalid requests)
- Analogy: Like the C++ language standard (C++23, C++26)—if you don't follow the spec, your code won't work.

Example: Simple Protocol Spec

- Message: [4 bytes length][UTF-8 string payload]
- Behavior: If length doesn't match payload, receiver discards message.

---

8. The Web: Fundamentals and Evolution

8.1 Web History and Components

- Invented by Tim Berners-Lee at CERN.
- Two main components:
    1. **HTTP** (Hypertext Transfer Protocol)
    2. **HTML** (Hypertext Markup Language)
- Original goal: Make it easy to share and navigate research papers.
- Relied on simple document formatting (HTML) and a simple application-layer protocol (HTTP over TCP).
- First web server: Berners-Lee's NeXT workstation at CERN ("Do not power off" sign).

8.2 HTTP: Hypertext Transfer Protocol

- Built atop TCP (originally, now also UDP/QUIC for HTTP/3).
- Request-response model: client sends a request, server sends a response.
- Each request/response has a header and a body (data).

Example HTTP/1.0 Request

```
GET / HTTP/1.0
<empty line>
```

Example HTTP/1.1 Response Header

```
HTTP/1.1 200 OK
Date: Tue, 01 Jan 2020 00:00:00 GMT
Server: Apache
Content-Length: 12345
Content-Type: text/html
Connection: close
```

Common HTTP Header Fields

Field	Purpose
Date	Timestamp of response
Server	Web server software (e.g., Apache)
Last-Modified	When resource last changed
E-Tag	Resource identifier/version
Accept-Ranges	Allow clients to request only part of a resource

Field	Purpose
Content-Length	Byte count of body
Connection	Whether connection should be closed after response
Content-Type	Media type (e.g., text/html, image/jpeg)

- E-Tag: Unique identifier for a version of a resource; used for caching and conditional requests.

8.3 HTTP Protocol Evolution

Version	Year	Key Features	Approx Usage
HTTP/1.0	1990s	One request per connection	Legacy
HTTP/1.1	1999	Persistent connections, Host headers, multiple requests per connection	~10%
HTTP/2	2015	Multiplexing, header compression, server push, pipelining, based on TCP	~59%
HTTP/3	2022	Built on UDP/QUIC, avoids TCP head-of-line blocking, better for real-time media	~32%

HTTP/1.1

- Adds persistent connections (multiple requests per TCP connection).
- Host header allows multiple domains per server.
- Improves network efficiency.
- HTTPS (HTTP over TLS) encrypts data for privacy and integrity.
- Tools: GnuTLS CLI, OpenSSL.

HTTP/2

- Launched 2015, ~59% adoption.
- Solves head-of-line blocking by multiplexing multiple streams over a single TCP connection.
- Features:
    - **Header Compression**: Reduces size of repetitive headers.
    - **Server Push**: Server can send multiple resources in response.
    - **Pipelining**: Multiple requests sent without waiting for responses.
    - **Multiplexing**: Interleaved transmission of multiple streams.

HTTP/1.1 vs HTTP/2 Multiplexing:

```
HTTP/1.1 (Pipelining):
Client: |--A--|--B--|--C--|
Server: |--A--|--B--|--C--|
(Responses must be sent in order)

HTTP/2 (Multiplexing):
Client:  |--A--|--B--|--C--|--D--|
Server:  |--B--|--D--|--A--|--C--|
(Responses can be interleaved)
```

HTTP/3

- Developed by Google (Jim Roskind), 2022, ~32% adoption.
- Built atop QUIC (UDP-based), not TCP.
- Motivation: Avoid TCP's head-of-line blocking, improve real-time streaming (e.g., Zoom).
- Features:
    - Streams with tolerable loss (partial loss tolerance).
    - Multiple streams between endpoints.
    - Encryption by default (TLS is integral).
    - Better performance over unreliable networks.

---

9. HTML: HyperText Markup Language

- Based on SGML (Standard Generalized Markup Language).
- HTML: Simplified, adds "hypertext" (links, interactivity).
- Uses angle-bracket tags (e.g., `<p>`, `<a href>`, `<h1>`, `<img src>`).
- Lowercase convention (vs. SGML's uppercase).

- DOM structure: Tree of nested elements.

DOM Tree Example:

```
<html>
  <head>
    <title>
  <body>
    <h1>
    <p>
```

Example HTML

```html
<html>
  <head><title>My Web Page</title></head>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

---

10. Additional Questions and Examples

1. **Why is pipelining/multiplexing useful?**
   - Reduces latency, especially for asynchronous data.
   - Lets users interact without waiting for all data.
2. **Why can TCP be inefficient for video?**
   - Retransmitted packets introduce delay/stutter; QUIC/UDP solves this.
3. **Why don't all use IPv6?**
   - Legacy hardware/software, institutional inertia, IPv4 still works.
4. **What is head-of-line blocking?**
   - One lost packet blocks all subsequent ones from being processed.

Summary

Layer	Main Protocols	Key Functionality
Application	HTTP, FTP, SMTP, DNS	User-facing services, data formats
Transport	TCP, UDP	Reliable/unreliable delivery, ports
Internet	IP, ICMP	Routing, addressing, fragmentation
Link	Ethernet, Wi-Fi, ARP	Local delivery, MAC addressing

This lecture introduced foundational concepts of Internet technology and packet-switched networking, focusing on the importance of protocol layers in handling reliability, order, and abstraction. It covered the four key network layers (link, internet, transport, application), the problems of packet loss, duplication, and disorder, and how these are abstracted away in the transport layer (TCP). The evolution of the web was discussed, from HTTP and HTML to modern protocols like HTTP/2 and HTTP/3, with their performance optimizations and trade-offs. The lecture also emphasized the importance of protocol specifications and the end-to-end principle in network design.