Internet Technology and Web Services

1. Introduction to Web Technology Evolution

This lecture covers the foundations of Internet technology, key concepts around protocol layers, and how these technologies underpin the development of the World Wide Web. Topics include the problems related to packet-switched networking, layered protocols, TCP/IP stack, and core web services such as HTTP, HTTPS, and HTML.

---

2. Fundamental Concepts of Networking

2.1 Circuit Switching vs. Packet Switching

- **Circuit Switching**: Dedicated path between endpoints (e.g., traditional telephony)
- **Packet Switching**: Data is sent in small units (packets) that may take different paths to the destination (used in the Internet).

Feature	Circuit Switching	Packet Switching
Path setup	Required	Not required
Reliability	More predictable	Less predictable
Efficiency	Low (fixed path)	High (dynamic routing)

2.2 Definition and Structure of Packets

- Packet: Data unit typically 1–2 KiB in size.
- Structure:
  - **Header**: Control info (e.g., destination, protocol type); for routers
  - **Payload**: Application data; for recipient applications

Example Packet Composition:

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?

- Protocol: Agreed rules for message transmission and processing
- Analogy: Diplomatic protocols required to speak to foreign leaders
- Each protocol layer defines what the structure and behavior of data exchanges look like.

Types of Protocol Behaviors: - Message formats (e.g., header structures) - Response behavior (e.g., what to do on success/failure)

3.2 Challenges in Packet Switching

Because packets travel independently, several challenges arise:

1. Packet Loss

- Common causes: router overload (buffer overflow), network congestion.
- Packets can be silently dropped.
- Loss rate varies: 1% to over 50% under high load.

2. Packets Received Out of Order

- Common due to multiple network paths.
- Can confuse applications requiring strict order (e.g., file uploads).

3. Packet Duplication

- Unusual, but possible with network misconfigurations (e.g., routers misconfigured as bridges).

4. Packet Corruption

- Bit errors due to unreliable hardware, faulty links.
- Need checksum/error-detection mechanisms.

---

4. Layered Network Architecture

4.1 Overview

To handle network complexity and address challenges modularly, protocols are structured into layers.

Layer	Functionality
Application	High-level data exchange (e.g., HTTP, HTML)
Transport	Reliable or fast delivery (e.g., TCP, UDP)
Internet	Routing across the network (e.g., IP)
Link	Data link between adjacent network nodes

Other models (e.g., OSI) define 7 layers (application, presentation, session, transport, network, data link, physical)

Notes:

- Layers act as abstractions over lower ones.
- Data encapsulation uses headers at each layer.

4.2 The Link Layer

- Hardware-specific protocols (e.g., Ethernet, Wi-Fi)
- Handles communication over a single, direct link

Example Link Layer Technologies:

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

4.3 The Internet Layer

- Primary protocol: IP (IPv4 or IPv6)
- Responsible for global addressing and packet routing via IP Addresses
- Stateless, best-effort delivery
- Introduces problems like loss, reordering, duplication

4.4 The Transport Layer

- Manages reliable delivery over unreliable internet layer
- Introduces concept of:
    - Data channels (logical streams)
    - Reliability
    - Ordering

    - Error Correction
- Protocols: TCP and UDP

4.5 The Application Layer

- Protocols meant for specific services (e.g., HTTP, FTP, SMTP)
- Builds on lower layers but focuses on the user's domain concerns

---

5. Internet Protocol (IP)

5.1 IPv4 Overview

- Introduced in 1983 (John Postel, UCLA alum).
- Connectionless — just delivers packets.

5.2 Fields in IPv4 Header

Field	Description
Length	Packet size in bytes
Protocol Number	Defines encapsulated protocol (e.g., TCP = 6; UDP = 17)
Source IP	32-bit address of sender
Destination IP	32-bit address of recipient
TTL	Time-to-live to prevent infinite loops, decremented at each router hop
Checksum	Simple 16-bit checksum for packet corruption detection

Notation:

- Binary: 11000000.10101000.00000000.00000001
- Decimal: 192.168.0.1

5.3 Addressing Schemes – IPv4 vs IPv6

Limitations of IPv4

- 32-bit address space (~4.3 billion addresses) proved insufficient.
- Large headers and lack of certain metadata features.

IPv6

- Introduced in 1998, addresses shortage issues.
- Features:
    - 128-bit addresses
    - More header fields
    - Additional features (not covered in detail)
- 40% adoption (still increasing)
- Efficiency trade-offs due to extra header size

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

Adoption status: - IPv4 still widely used (60%) - IPv6 rising (40%) due to address exhaustion

5.4 Checksums and Reliability

- Internet checksum (16-bits) for error detection
- Cannot stop malicious actors, not cryptographic

- Used to catch accidental data corruption
- End-to-End Principle: Each layer should do its error checking

---

6. Transport Layer Protocols

6.1 User Datagram Protocol (UDP)

- Barebones protocol
- Source Port & Destination Port
- Unreliable and unordered delivery
- No guarantees: Sender must handle ordering, errors, etc.
- Used in:
  - Streaming, DNS, real-time telemetry
  - Situations where speed > reliability
- Does not resolve packet issues like loss or order

Use Case Example: IoT device sending temperature once every 10 minutes

6.2 Transmission Control Protocol (TCP)

- Reliable, connection-oriented
- Implements:
  - **Reliability**: Retransmission of lost packets
  - **Ordered data**: Reassembly of out-of-order packets
  - **Error checking**: End-to-end validation
- Guarantees:
  - In-order, lossless, error-checked delivery

Reliability Features Table:

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	Using sequence numbers to identify and discard redundant data
Overflow	Flow Control	Avoid flooding networks. Sender adjusts speed based on network conditions.

Common use cases: - Submitting assignments - Large data transfers - Web traffic with HTTP(S)

---

7. Protocol Specification – How to Create a Protocol

- Define:
  - Packet formats and headers
  - Expected behaviors
- Documentation (spec), not code
- Analogy: Like C++ standard (C++23, C++26)
- Failing to follow spec = non-functional applications

Example: - What headers are required? - What order are fields? - What responses to invalid requests?

---

8. The Web Fundamentals

8.1 Introduction and Web History

Invented by Tim Berners-Lee at CERN with two main components:

1. **HTTP** (Hypertext Transfer Protocol)

2. **HTML** (Hypertext Markup Language)

Original Goals:

- Create a simple way to share and navigate research papers.
- Relied on:
  - Simple document formatting (HTML)
  - Simple app-layer protocol (HTTP over TCP)

First Web Server:

- Hosted on Berners-Lee's NeXT workstation
- Sign: "Do not power off" — critical to early web functionality- First web server hosted on his workstation (CERN)
- Protocol: HTTP, Markup: HTML

8.2 HTTP – Hypertext Transfer Protocol

8.3 HTTP Protocol Evolution

Version	Year	Key Features	Approx Usage
HTTP/1.0	1990s	One request per connection	Legacy usage
HTTP/1.1	1999	Persistent connections, Host headers	~10%
HTTP/2	2015	Based on TCP; improves performance	~59%
HTTP/3	2022	Based on UDP via QUIC; optimized for real-time media	~32%

HTTP/1.0

- Built on top of TCP
- Request-response model (initially one request per TCP connection)

Example Request:

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

Example 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
```

HTTP Header Fields (Example)

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

HTTP/1.1

- Adds:
  - Persistent connections
  - Enables multiple requests per TCP connection
  - Host header support (supports multiple domains per server)
  - Improves network efficiency

HTTPS and Security

- Used today predominantly
- Encrypts HTTP using TLS
- Prevents:
  - Packet inspection (privacy)
  - Packet tampering (integrity)
- Tools: GnuTLS CLI, OpenSSL

HTTP/2

Launched 2015, 59% adoption

In HTTP/1.1, each request–response pair typically required either:

- Its own TCP connection, or
- Had to wait (block) if reusing a connection (due to head-of-line blocking). This made web pages slow to load because:
- You could only send one request at a time per connection.
- Or, you had to open lots of TCP connections, which increased overhead.

What HTTP/2 does differently with multiplexing:

- Single TCP connection: All communication between the client and server happens over one connection.
- Streams: Within this connection, many streams can be open at once — each representing a request/response.
- Interleaving: HTTP/2 breaks data into small packets called frames and interleaves them — so parts of different streams can be sent back and forth simultaneously.
- No blocking: Because of this, one slow response won't block others.

Feature	Purpose
Header Compression	Reduces size of repetitive headers
Server Push	Server sends multiple resources as a response
Pipelining	Multiple requests dispatched without waiting for responses sequentially
Multiplexing	Allows interleaved transmission of multiple streams (faster responses)

Example Multiplex Order:

Request	Response
A	B
B	D
C	A

HTTP/3

- Developed by Google, 2022
- 32% adoption
- Built atop **QUIC**, which uses **UDP**
- Motivation: avoid TCP's "head-of-line blocking"
- Enables:
  - Partial loss tolerance (key for streaming apps like Zoom)
  - Encryption by default (TLS is integral)
  - Better performance over unreliable networks

Characteristics of QUIC: - Built by Google (Jim Roskind) - Supports: - Streams with tolerable loss - Multiple streams between two endpoints - Improved performance compared to TCP

---

9. HTML – HyperText Markup Language

- Based on SGML (Standard Generalized Markup Language)
- HTML: Simplified + includes "hypertext" (links, interactivity)
- Uses angle-bracket tags:

```
<p>This is a paragraph</p>
```

- Lowercase convention (vs SGML's uppercase)

Elements: - Tags: <p>, <a href>, <h1>, <img src>, etc. - DOM structure: Tree of nested elements

HTML Example:

```
<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 (particularly with async data)
    - Lets humans interact without waiting
2. Why can TCP be inefficient for video?
    - Retransmitted packets introduce delay/stutter. QUIC solves this.
3. Why don't all use IPv6?
    - Legacy hardware, software
    - Institutional inertia (IPv4 still works)
4. What is head-of-line blocking?
    - If one lost packet blocks all subsequent ones from being processed

---

Summary

This lecture introduced foundational concepts of Internet technology and packet-switched networking, with a focus on the importance of protocol layers in handling reliability, order, and communication abstraction. It covered key network layers (link, internet, transport, application) and exemplified how the issues of packet loss, duplication, and disorder are abstracted away primarily in the transport layer via protocols like TCP. The lecture also analyzed the evolution of the web, beginning with the HTTP and HTML protocols invented by Tim Berners-Lee, illustrating how higher-layer web services rely on and build upon the lower networking layers. Finally, it delved into modern advancements such as HTTP/2 and HTTP/3 and their performance optimizations and trade-offs.