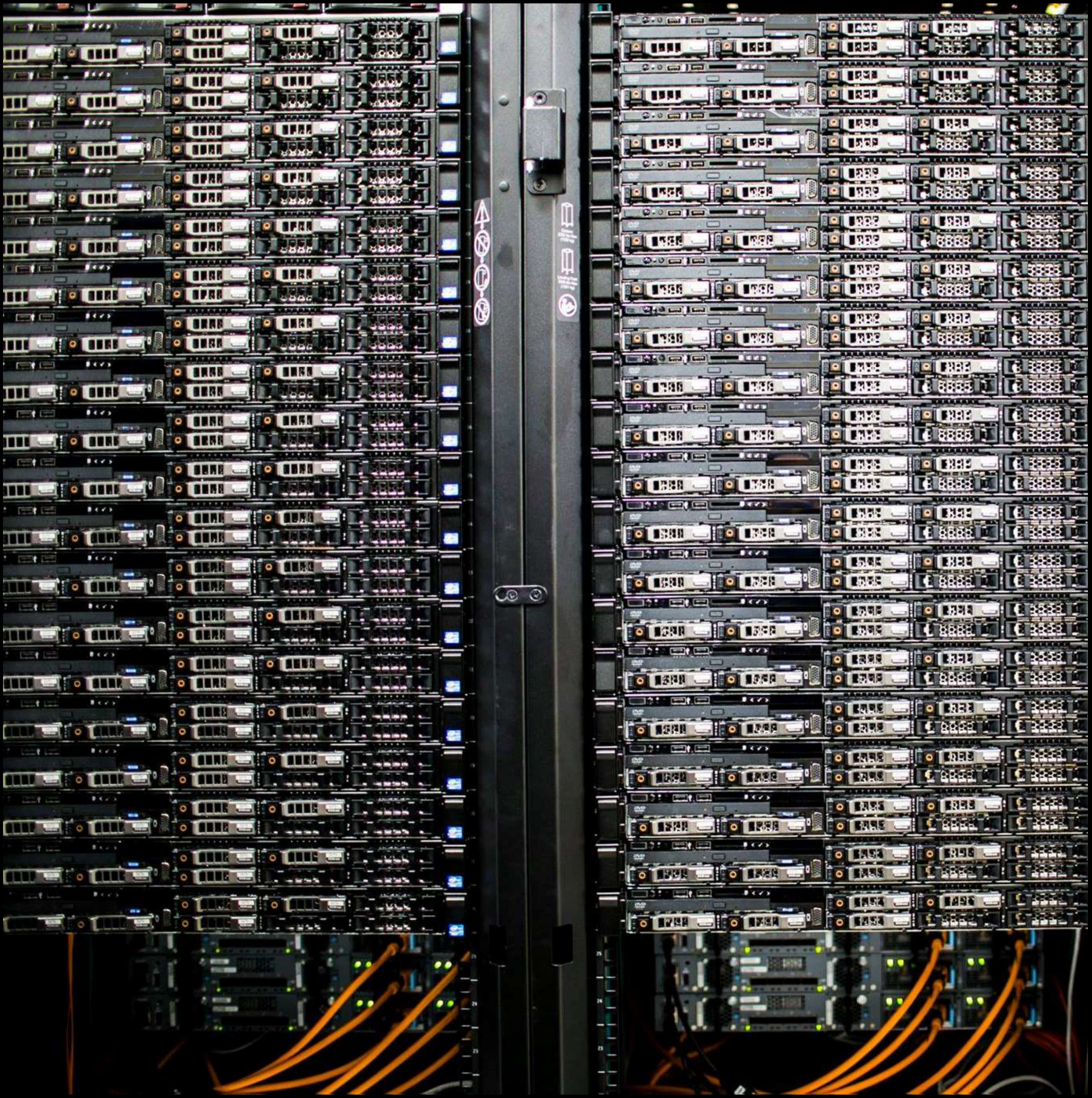


HTTP2 Server

Computer Networks

Ain Shams University
Faculty of Engineering
CESS
CSE351 Computer Networks
Fall 24
Proposal Document

Proposal Done by:

Youssef Mohamed AlSayed	21P0094
Youssef Mohamed Salah	21P0159
Saleh Ahmed ElSayed	21P0324

Table of Contents

Components....1

Connection Handler....1

Frame Processor....1

Stream Manager....2

HPACK Compression/Decompression Module....3

Flow Control Module....3

Error Handler....3

Interaction Between Components....4

Connection Handler....4

Frame Processor....4

Stream Manager....5

HPACK Compression/Decompression Module....5

Flow Control Module....5

Error Handler....5

System Communication Protocols....6

HTTP/2 Frames....6

Header Compression....6

Multiplexing and Flow Control....6

Key Functionalities of Web Server....6

Handle Client Connections....6

Protocol Handling (HTTP/2)....6

Error Handling and Custom Error Pages....6

COMPONENTS

Connection Handler

- Manages TCP connections from clients.
- Validates HTTP/2 connection preface.
- Initializes client-specific settings.

The Connection Handler is responsible for managing TCP connections that clients establish with the server. Its first task is to validate the HTTP/2 connection preface, ensuring that the client adheres to the protocol's initial handshake requirements as specified in RFC7540. This involves checking for the specific magic string and correctly handling settings exchanged during the initial communication. Additionally, it initializes client-specific settings such as flow control parameters and stream limits, tailoring the server's behavior to each client's needs while ensuring compatibility with the HTTP/2 protocol.

Frame Processor

- Parses incoming HTTP/2 frames (HEADERS, DATA, SETTINGS,....).
- Validates frame types, flags, and lengths.
- Delegates frame processing to the appropriate handler.

```
+-----------------------------------------------+
|                 Length (24)                   |
+---------------+---------------+---------------+
|   Type (8)    |   Flags (8)   |
+-+-------------+---------------+-------------------------------+
|R|                 Stream Identifier (31)                      |
+=+=============================================================+
|                   Frame Payload (0...)                      ...
+---------------------------------------------------------------+
```

The Frame Processor is designed to handle the parsing and validation of incoming HTTP/2 frames, which include HEADERS, DATA, SETTINGS, and other types. It ensures that each frame complies with the protocol’s structure, verifying frame types, flags, and lengths as outlined in the specification. Once a frame passes validation, the Frame Processor delegates its handling to the appropriate component, whether it's routing a HEADER frame to stream initialization or passing a DATA frame for further processing.

Stream Manager

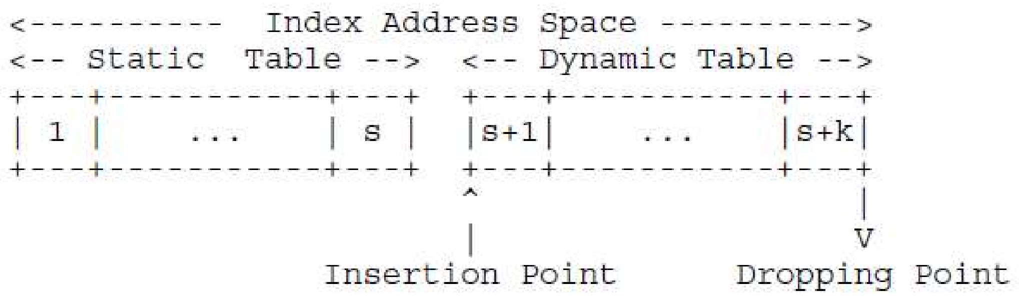
- Manages the lifecycle of streams (creation, activity, closure).
- Handles stream prioritization and dependencies.
- Ensures compliance with HTTP/2 multiplexing rules.

```
                         +--------+
                 send PP |        | recv PP
                ,--------|  idle  |--------.
               /         |        |         \
              v          +--------+          v
       +----------+          |           +----------+
       |          |          | send H /  |          |
,------| reserved |          | recv H    | reserved |------.
|      | (local)  |          |           | (remote) |      |
|      +----------+          v           +----------+      |
|          |             +--------+             |          |
|          |     recv ES |        | send ES     |          |
|   send H |     ,-------|  open  |-------.     | recv H   |
|          |    /        |        |        \    |          |
|          v   v         +--------+         v   v          |
|      +----------+          |           +----------+      |
|      |   half   |          |           |   half   |      |
|      |  closed  |          | send R /  |  closed  |      |
|      | (remote) |          | recv R    | (local)  |      |
|      +----------+          |           +----------+      |
|           |                |                 |           |
|           | send ES /      |       recv ES / |           |
|           | send R /       v        send R / |           |
|           | recv R     +--------+   recv R   |           |
| send R /  `----------->|        |<-----------'  send R / |
| recv R                 | closed |               recv R   |
`----------------------->|        |<----------------------'
                         +--------+
```

The Stream Manager oversees the lifecycle of HTTP/2 streams, from their creation and activity to their eventual closure. It handles complex tasks like stream prioritization and managing dependencies between streams, ensuring that the server adheres to HTTP/2's multiplexing rules. This includes respecting priority weights and dependencies to optimize resource allocation and stream scheduling. Additionally, the Stream Manager ensures compliance with protocol constraints, such as avoiding stream ID reuse and maintaining proper state transitions.

HPACK Compression/Decompression Module

- Implements header compression and decompression using static and dynamic tables.
- Encodes and decodes HTTP header blocks as specified in RFC 7541.



The HPACK Compression/Decompression Module is responsible for efficiently compressing and decompressing HTTP headers using both static and dynamic tables, as outlined in RFC 7541. This module plays a critical role in minimizing the size of HTTP header blocks, which reduces bandwidth usage and improves performance in HTTP/2 communication. When compressing headers, it maps header field names and values to indexes in a pre-defined static table or dynamically maintained tables shared between the client and server. During decompression, it reverses this process, translating the compressed representation back into human-readable HTTP headers.

Flow Control Module

- Manages connection-level and stream-level flow control.
- Processes WINDOW_UPDATE frames and adjusts flow control windows accordingly.

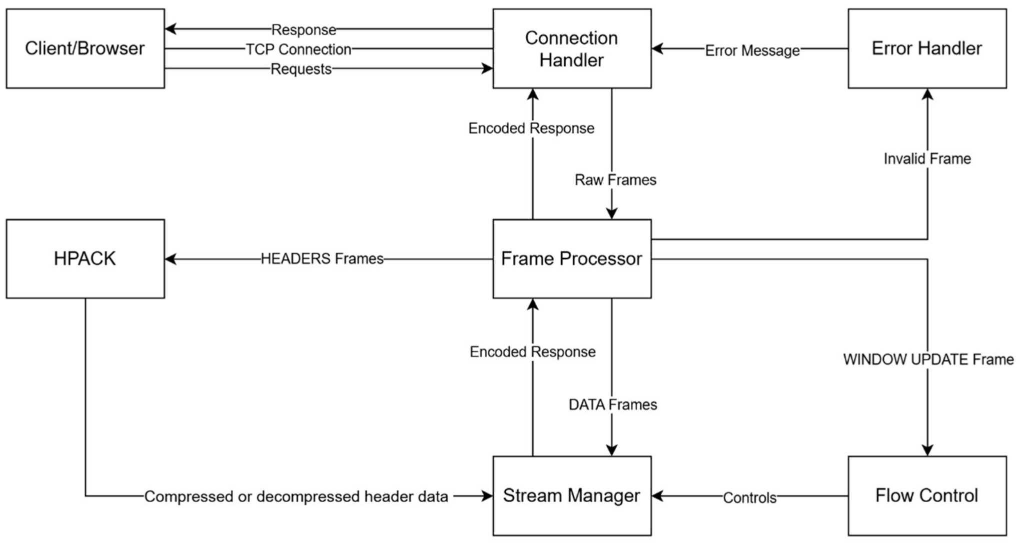
The Flow Control Module handles the critical task of managing data flow between the client and server at both the connection and stream levels, as specified in RFC 7540. It ensures that neither side becomes overwhelmed by too much incoming data by regulating the size of the flow control windows. This module processes `WINDOW_UPDATE` frames sent by peers, which inform it of available space in their respective flow control windows. By dynamically adjusting these windows, the module prevents buffer overflows and ensures smooth and reliable data transmission. This flow control mechanism is essential for optimizing throughput and fairness in multi-stream scenarios where multiple streams share the same connection.

Error Handler

- Detects and handles protocol errors such as malformed frames or invalid stream states.
- Sends appropriate HTTP/2 error codes (RST_STREAM, GOAWAY, ...).

The Error Handler is designed to detect and respond to protocol errors that may occur during HTTP/2 communication. These errors can include malformed frames, invalid stream states, or violations of HTTP/2's stateful protocol rules, as described in RFC 7540. When an error is detected, the handler sends the appropriate HTTP/2 error code, such as **RST_STREAM** to terminate a specific stream or **GOAWAY** to signal the end of the connection. This ensures that protocol errors are addressed gracefully without disrupting other active streams or degrading overall connection performance. By maintaining proper error handling, this component preserves the integrity and robustness of the HTTP/2 server.

INTERACTION BETWEEN COMPONENTS



Connection Handler

Input	Incoming client TCP connection.
Tasks	Read and validate the HTTP/2 connection preface. Exchange initial SETTINGS frames with the client. Forward incoming frames to the Frame Processor.
Output	Validated and parsed frames passed to the Frame Processor.

Frame Processor

Input	Raw frames received from the Connection Handler.
Tasks	Parse frame headers and payloads. Dispatch frames to the appropriate modules Enforce protocol rules, such as stream ID validation.
Output	Processed frames forwarded to respective modules

Stream Manager

Input	HEADERS and DATA frames.
Tasks	Manage stream states (idle, open, closed). Handle prioritization and dependency hierarchies. Forward stream-specific data to the application layer.
Output	Encoded/decoded HTTP/2 responses sent to the Frame Processor.

HPACK Compression/Decompression Module

Input	Header blocks from HEADERS frames.
Tasks	Compress HTTP/2 response headers into header blocks. Decompress client header blocks and populate dynamic tables.
Output	Compressed or decompressed header data passed to the Stream Manager.

Flow Control Module

Input	WINDOW_UPDATE frames.
Tasks	Manage flow control windows for streams and the connection. Ensure that outgoing DATA frames respect flow control limits.
Output	Adjusted flow control windows passed to the Frame Processor.

Error Handler

Input	Detected protocol violations or internal errors.
Tasks	Log errors and send appropriate HTTP/2 error frames (e.g., GOAWAY). Gracefully terminate streams or the connection.

SYSTEM COMMUNICATION PROTOCOLS

HTTP/2 Frames

- The server will implement the frame structure defined in RFC 7540, with each frame comprising:
    - **9-byte header**: Includes length, type, flags, and stream ID.
    - **Variable-length payload**: Depends on the frame type.

Header Compression

- Use HPACK (RFC 7541) to compress and decompress HTTP/2 headers.
- Maintain static and dynamic tables for efficient header encoding/decoding.

Multiplexing and Flow Control

- Use the stream identifier to manage multiple streams over the same connection.
- Respect flow control windows to ensure efficient data transmission.

KEY FUNCTIONALITIES OF WEB SERVER

Handle Client Connections

The web server will be designed to efficiently accept incoming connections from clients, ensuring smooth communication. It will maintain persistent connections by implementing the "keep-alive" mechanism. This avoids the need to repeatedly establish new connections for every request, reducing latency and improving performance. By keeping the connection alive, the server can handle multiple requests and responses over a single connection, making it suitable for modern applications where speed and resource optimization are critical.

Protocol Handling (HTTP/2)

The server will handle HTTP/2 protocol processing, which includes managing HTTP/2 frames as defined in RFC7540. Frames are the fundamental units of communication in HTTP/2, and the server will parse and interpret them to deliver content accurately and efficiently. Additionally, it will implement HPACK, a header compression mechanism detailed in RFC7541. HPACK reduces the size of HTTP headers by using a combination of static and dynamic header tables, which saves bandwidth and enhances performance, especially for repeated headers in subsequent requests. This capability ensures faster and more efficient data transmission, essential for modern web applications.

Error Handling and Custom Error Pages

The server will provide robust error handling, returning appropriate HTTP status codes to indicate success, client errors, or server issues. For instance, it will send codes like 404 for "Not Found" or 500 for "Internal Server Error" when needed. Furthermore, it will include the ability to serve custom error pages, offering a user-friendly and branded way to handle errors. These pages can provide more context about the problem and guide users on what to do next, making the server more polished and user-centric.