System Lab Practice Exam

Video Streaming System Implementation

Section 1: Multithreading and Synchronization

Q1) In the server implementation, multiple mutex locks are used. Explain why each of these mutexes is necessary:

```
pthread_mutex_t stats_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t queue_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t udp_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t log_mutex = PTHREAD_MUTEX_INITIALIZER;
```

Answer:

- `stats_mutex`: Protects access to shared client statistics data structure (ClientStats array) which is accessed by multiple threads for updating and reading client information
- `queue_mutex`: Guards the client queue data structure used in FCFS scheduling, preventing race conditions when enqueueing/dequeuing clients
- `udp_mutex`: Protects the shared UDP socket which is used by multiple client threads for sending data
- `log_mutex`: Ensures thread-safe logging by preventing interleaved output from multiple threads

Q2) What potential race condition could occur in the following code snippet if the mutex lock was not used? How does the mutex prevent it?

```
#ifdef _WIN32
    MUTEX_LOCK(stats_mutex);
#else
    pthread_mutex_lock(&stats_mutex);
#endif
    client_stats[client_id].bytes_sent += bytes;
    client_stats[client_id].chunks_sent++;
    client_stats[client_id].current_time = get_time();
#ifdef _WIN32
    MUTEX_UNLOCK(stats_mutex);
#else
    pthread_mutex_unlock(&stats_mutex);
#endif
```

Answer: Without the mutex, multiple threads could try to update the statistics simultaneously, leading to:

- Lost updates if two threads read the old value, increment, and write back
- Incorrect chunk counting if increments overlap

- Inconsistent timing information if updates are interleaved The mutex ensures these operations happen atomically by allowing only one thread to modify the statistics at a time.

Section 2: Network Programming

Q3) Explain the two-phase communication protocol implemented in the system. Why was this design chosen?

Answer: The system uses a two-phase approach:

1. Connection Phase:
    - Uses TCP for reliable parameter negotiation
    - Client sends Type 1 Request (resolution, protocol)
    - Server responds with Type 2 Response (bandwidth, client ID)
    - Connection is closed after negotiation
2. Streaming Phase:
    - New connection on different port
    - Can use either TCP or UDP based on client choice
    - Actual video data transmission

This design was chosen because:

- Separates control (reliable TCP) from data transfer (TCP/UDP)
- Allows protocol flexibility for streaming
- Enables better resource management
- Simplifies error handling and connection state management

Q4) Analyze the following code snippet and explain how it handles TCP streaming reliability:

```
while (total_sent < TCP_CHUNK_SIZE && retry_count < max_send_retries) {
    fd_set write_fds;
    FD_ZERO(&write_fds);
    FD_SET(client_socket, &write_fds);

    struct timeval write_tv;
    write_tv.tv_sec = 1;
    write_tv.tv_usec = 0;

    int select_result = select(client_socket + 1, NULL, &write_fds, NULL,
&write_tv);
    // ... rest of the code
}
```

Answer: This code implements reliable TCP streaming by:

1. Using select() to check if socket is ready for writing
2. Implementing a retry mechanism (max_send_retries)

3. Using timeouts to prevent indefinite blocking
4. Tracking partial sends (total_sent) to ensure complete data transmission
5. Allowing for backpressure by respecting socket buffer availability

Section 3: System Design

Q5) The server implements two scheduling policies: FCFS and Round-Robin. Compare their implementations and discuss their trade-offs.

Answer: FCFS (First-Come-First-Serve):

- Implementation: Simple queue-based system
- Advantages:
  - Predictable order of execution
  - Simple to implement
  - Fair for short streams
- Disadvantages:
  - Can lead to head-of-line blocking
  - Long streams can block others
  - Higher latency for later requests

Round-Robin:

- Implementation: Cycles through active clients
- Advantages:
  - Better fairness for multiple clients
  - Lower average latency
  - Prevents starvation
- Disadvantages:
  - More complex implementation
  - Additional overhead for context switching
  - May not be optimal for streams of different sizes

Section 4: Debugging and Error Handling

Q6) Identify potential issues in the following code segment and suggest improvements:

```
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
}

if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    perror("Connection Failed");
    exit(EXIT_FAILURE);
}
```

Answer: Issues and improvements:

1. Resource Leakage:
    - Socket not closed before exit
    - Should add: close(sock) before exit()
2. Error Handling:
    - Too aggressive with exit()
    - Should return error code instead for better error handling
3. Platform Dependency:
    - Doesn't handle Windows/UNIX differences
    - Should use platform-independent macros

Improved version:

```
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET_VALUE) {
    print_socket_error("Socket creation failed");
    return -1;
}

if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    print_socket_error("Connection Failed");
    CLOSE_SOCKET(sock);
    return -1;
}
```

Section 5: Performance Analysis

Q7) The system implements different chunk sizes for TCP and UDP. Explain why and how it affects performance:

```
#define TCP_CHUNK_SIZE 131072  // 128KB for TCP
#define UDP_CHUNK_SIZE 8192    // 8KB for UDP
```

Answer: TCP Chunk Size (128KB):

- Larger size because TCP handles:
    - Flow control
    - Congestion control
    - Reliable delivery
- Benefits:
    - Better throughput for large transfers
    - Reduced overhead from fewer packets
    - Efficient use of TCP sliding window

UDP Chunk Size (8KB):

- Smaller size because:
  - No fragmentation handling
  - Must fit in single packet
  - Higher risk of packet loss
- Benefits:
  - Lower latency
  - Less impact from packet loss
  - Better real-time performance

Performance Impact:

- TCP: Higher throughput, higher latency
- UDP: Lower latency, potential packet loss
- Choice depends on application needs (reliability vs. speed)

Section 6: Practical Application

Q8) Given a scenario where your video streaming system needs to handle 1000 concurrent clients, what modifications would you make to the current implementation? Consider resource limitations and performance optimization.

Answer: Required modifications:

1. Connection Management:
   - Implement connection pooling
   - Add client request queuing
   - Implement load balancing
   - Use epoll/kqueue instead of select
2. Resource Management:
   - Dynamic thread pool instead of thread-per-client
   - Implement client priority system
   - Add resource usage limits per client
   - Implement graceful degradation
3. Memory Optimization:
   - Use shared memory for static data
   - Implement buffer pooling
   - Add memory usage monitoring
   - Optimize data structures
4. Performance Enhancements:
   - Add caching layer
   - Implement request batching
   - Use zero-copy networking where possible
   - Add performance metrics collection

5. Error Handling:
    - Implement circuit breakers
    - Add automatic recovery mechanisms
    - Enhanced logging and monitoring
    - Graceful shutdown procedures

Section 7: Additional Practice Questions

Q9) Analyze this thread pool implementation and identify potential issues:

```
typedef struct {
    pthread_t *threads;
    int num_threads;
    queue_t *task_queue;
    pthread_mutex_t queue_lock;
    pthread_cond_t queue_notify;
} threadpool_t;

void *thread_function(void *arg) {
    threadpool_t *pool = (threadpool_t *)arg;
    while (1) {
        pthread_mutex_lock(&pool->queue_lock);
        while (queue_empty(pool->task_queue)) {
            pthread_cond_wait(&pool->queue_notify, &pool->queue_lock);
        }
        task_t *task = queue_dequeue(pool->task_queue);
        pthread_mutex_unlock(&pool->queue_lock);
        if (task != NULL) {
            task->function(task->arg);
            free(task);
        }
    }
    return NULL;
}
```

Answer: Critical issues in this implementation:

1. No shutdown mechanism - threads run forever
2. Missing error handling for pthread functions
3. Potential memory leak if task allocation fails
4. No thread pool size validation
5. Missing signal handling for clean shutdown

Improvements needed:

```
// Add shutdown flag
volatile int shutdown = 0;

// Modified thread function
```

```
void *thread_function(void *arg) {
    threadpool_t *pool = (threadpool_t *)arg;
    while (!shutdown) {
        pthread_mutex_lock(&pool->queue_lock);
        while (queue_empty(pool->task_queue) && !shutdown) {
            pthread_cond_timedwait(&pool->queue_notify, &pool->queue_lock,
&timeout);
        }
        if (shutdown) {
            pthread_mutex_unlock(&pool->queue_lock);
            break;
        }
        // ... rest of the code
    }
    return NULL;
}
```

Q10) Implement a video frame buffer with circular queue. What synchronization mechanisms are needed?

Answer: Here's a thread-safe circular buffer implementation:

```
typedef struct {
    uint8_t **frames;
    size_t frame_size;
    int capacity;
    int head;
    int tail;
    int count;
    pthread_mutex_t lock;
    pthread_cond_t not_full;
    pthread_cond_t not_empty;
} frame_buffer_t;

int frame_buffer_push(frame_buffer_t *buf, uint8_t *frame) {
    pthread_mutex_lock(&buf->lock);
    while (buf->count == buf->capacity) {
        pthread_cond_wait(&buf->not_full, &buf->lock);
    }

    memcpy(buf->frames[buf->tail], frame, buf->frame_size);
    buf->tail = (buf->tail + 1) % buf->capacity;
    buf->count++;

    pthread_cond_signal(&buf->not_empty);
    pthread_mutex_unlock(&buf->lock);
    return 0;
}

int frame_buffer_pop(frame_buffer_t *buf, uint8_t *frame) {
    pthread_mutex_lock(&buf->lock);
```

```
    while (buf->count == 0) {
        pthread_cond_wait(&buf->not_empty, &buf->lock);
    }

    memcpy(frame, buf->frames[buf->head], buf->frame_size);
    buf->head = (buf->head + 1) % buf->capacity;
    buf->count--;

    pthread_cond_signal(&buf->not_full);
    pthread_mutex_unlock(&buf->lock);
    return 0;
}
```

Required synchronization:

1. Mutex for exclusive access to buffer state
2. Condition variables for:
    - Buffer not full (producers wait)
    - Buffer not empty (consumers wait)
3. Atomic operations for count updates

Q11) Explain the difference between adaptive bitrate streaming and fixed bitrate streaming.

Answer: Adaptive bitrate streaming:

1. Adjusts video quality based on network conditions
2. Provides better user experience by reducing buffering
3. Uses multiple bitrate versions of the same video
4. Dynamically switches between bitrates during playback

Fixed bitrate streaming:

1. Maintains constant video quality throughout playback
2. Simpler to implement compared to adaptive streaming
3. May lead to buffering under poor network conditions
4. Suitable for stable network environments

Q12) What is the role of a CDN in video streaming?

Answer: A Content Delivery Network (CDN) plays a crucial role in video streaming:

1. Distributes content geographically to reduce latency
2. Improves scalability by offloading traffic from the origin server
3. Provides redundancy to ensure high availability
4. Enhances user experience by delivering content from the nearest edge server

Q13) How does the system handle packet loss in UDP streaming?

Answer: Handling packet loss in UDP streaming involves:

1. Implementing error detection mechanisms to identify lost packets
2. Using retransmission strategies to recover lost data

3. Employing forward error correction to allow recovery without retransmission
4. Adjusting bitrate dynamically to minimize impact of packet loss

Q14) Describe the importance of load balancing in a video streaming system.

Answer: Load balancing is essential for:

1. Distributing client requests evenly across multiple servers
2. Preventing server overload and ensuring optimal performance
3. Improving system reliability and availability
4. Enhancing user experience by reducing latency and avoiding bottlenecks

Q15) What are the benefits of using epoll/kqueue over select for handling multiple connections?

Answer: Using epoll/kqueue offers several benefits over select:

1. Better scalability for handling large numbers of connections
2. Lower overhead due to more efficient event notification mechanisms
3. Improved performance under high load conditions
4. Reduced CPU usage and faster response times

Q16) Explain the concept of zero-copy networking.

Answer: Zero-copy networking involves:

1. Avoiding data copying between user and kernel space
2. Reducing CPU usage by eliminating unnecessary memory operations
3. Improving data transfer speed and overall system performance
4. Enhancing efficiency in high-throughput applications

Q17) How does the system ensure secure communication between clients and the server?

Answer: Secure communication is ensured through:

1. Using SSL/TLS for encrypting data in transit
2. Implementing authentication mechanisms to verify client identity
3. Employing secure key exchange protocols to protect encryption keys
4. Regularly updating security measures to address vulnerabilities

Q18) What is the role of a buffer pool in a video streaming system?

Answer: A buffer pool helps in:

1. Managing memory efficiently by reusing buffers
2. Reducing allocation overhead and improving performance
3. Preventing memory fragmentation and optimizing resource usage
4. Ensuring smooth and continuous data flow during streaming

Q19) How does the system handle high network latency during streaming?

Answer: High network latency is managed by:

1. Implementing adaptive bitrate streaming to adjust video quality

2. Using buffering techniques to smooth out playback
3. Employing error correction mechanisms to recover lost data
4. Adjusting streaming parameters dynamically based on network conditions

Q20) Describe the importance of monitoring in a video streaming system.

Answer: Monitoring is crucial for:

1. Tracking system performance and identifying bottlenecks
2. Helping capacity planning and resource allocation
3. Enabling proactive maintenance to prevent issues
4. Improving user experience by ensuring smooth operation

Q21) What are the challenges of implementing DRM in a video streaming system?

Answer: Challenges include:

1. Ensuring content protection against unauthorized access
2. Managing licenses and handling different DRM standards
3. Balancing security and performance to avoid degradation
4. Integrating DRM with existing infrastructure and workflows

Q22) How does the system handle client authentication?

Answer: Client authentication is handled through:

1. Secure login mechanisms to verify user identity
2. Implementing token-based authentication for session management
3. Employing multi-factor authentication for added security
4. Regularly updating authentication protocols to address vulnerabilities

Q23) Explain the concept of forward error correction in video streaming.

Answer: Forward error correction involves:

1. Adding redundant data to streams to allow error recovery
2. Enabling recovery without retransmission, improving reliability
3. Reducing impact of packet loss and ensuring smooth playback
4. Enhancing overall streaming quality and user experience

Q24) What are the benefits of using a dynamic thread pool in a video streaming server?

Answer: Benefits include:

1. Adapting to varying load conditions by adjusting thread count
2. Improving resource utilization and preventing thread exhaustion
3. Reducing thread creation overhead and enhancing performance
4. Ensuring efficient handling of client requests and improving response times

Q25) How does the system handle network congestion during streaming?

Answer: Network congestion is managed by:

1. Implementing congestion control mechanisms to regulate data flow
2. Using adaptive bitrate streaming to adjust video quality
3. Employing buffering techniques to smooth out playback
4. Adjusting streaming parameters dynamically based on network conditions

Q26) Describe the importance of caching in a video streaming system.

Answer: Caching is important for:

1. Reducing server load by storing frequently accessed content
2. Improving content delivery speed and reducing latency
3. Enhancing user experience by providing faster access to content
4. Reducing network traffic and optimizing resource usage

Q27) How does the system handle client disconnections?

Answer: Client disconnections are handled by:

1. Detecting connection timeouts and tracking TCP connection state
2. Cleaning up resources such as memory buffers and file handles
3. Logging disconnection events and updating client statistics
4. Implementing reconnection logic and supporting session resumption

Q28) Explain the concept of graceful degradation in a video streaming system.

Answer: Graceful degradation involves:

1. Maintaining service quality under high load conditions
2. Reducing resource usage to prevent system crashes
3. Ensuring continuous operation and minimizing impact on users
4. Implementing fallback mechanisms to handle overload scenarios

Q29) What are the benefits of using a caching layer in a video streaming system?

Answer: Benefits include:

1. Improving content delivery speed and reducing latency
2. Reducing server load by storing frequently accessed content
3. Enhancing user experience by providing faster access to content
4. Reducing network traffic and optimizing resource usage

Q30) How does the system handle high client load during streaming?

Answer: High client load is managed by:

1. Implementing load balancing to distribute requests evenly
2. Using dynamic thread pools to adapt to varying load conditions
3. Employing connection pooling to optimize resource usage
4. Adjusting streaming parameters dynamically based on client load

Q31) Describe the importance of error handling in a video streaming system.

Answer: Error handling is crucial for:

1. Ensuring system reliability and preventing data loss
2. Improving user experience by addressing issues promptly
3. Enabling quick recovery from errors and maintaining service continuity
4. Identifying and resolving potential problems before they escalate

Q32) How does the system handle video frame synchronization?

Answer: Video frame synchronization is managed by:

1. Using timestamps to ensure accurate playback timing
2. Implementing buffering techniques to smooth out playback
3. Employing synchronization algorithms to align audio and video
4. Adjusting playback dynamically based on synchronization requirements

Q33) Explain the concept of circuit breakers in a video streaming system.

Answer: Circuit breakers involve:

1. Preventing system overload by limiting resource usage
2. Ensuring service continuity by handling failures gracefully
3. Improving system reliability by isolating problematic components
4. Enabling quick recovery from errors and maintaining service quality

Q34) What are the benefits of using a performance metrics collection system in a video streaming server?

Answer: Benefits include:

1. Tracking system performance and identifying bottlenecks
2. Helping capacity planning and resource allocation
3. Enabling proactive maintenance to prevent issues
4. Improving user experience by ensuring smooth operation

Q35) How does the system handle video frame encoding?

Answer: Video frame encoding is managed by:

1. Using efficient codecs to compress video data
2. Implementing hardware acceleration to improve encoding speed
3. Employing adaptive bitrate streaming to adjust encoding parameters
4. Ensuring high-quality video output while optimizing resource usage

Q36) Describe the importance of resource usage monitoring in a video streaming system.

Answer: Resource usage monitoring is crucial for:

1. Ensuring efficient utilization of system resources
2. Preventing system overload and optimizing performance
3. Identifying potential issues and addressing them proactively
4. Improving overall system reliability and user experience

Q37) How does the system handle video frame decoding?

Answer: Video frame decoding is managed by:

1. Using efficient codecs to decompress video data
2. Implementing hardware acceleration to improve decoding speed
3. Employing buffering techniques to smooth out playback
4. Adjusting decoding parameters dynamically based on network conditions

Q38) Explain the concept of request batching in a video streaming system.

Answer: Request batching involves:

1. Grouping multiple client requests together to reduce server load
2. Improving performance by processing requests in bulk
3. Enhancing user experience by reducing latency
4. Optimizing resource usage and minimizing overhead

Q39) What are the benefits of using a shared memory system in a video streaming server?

Answer: Benefits include:

1. Reducing memory usage by sharing data between processes
2. Improving performance by eliminating unnecessary data copying
3. Enabling efficient data sharing and communication
4. Preventing memory fragmentation and optimizing resource usage

Q40) How does the system handle video frame rendering?

Answer: Video frame rendering is managed by:

1. Using efficient rendering algorithms to display video data
2. Implementing hardware acceleration to improve rendering speed
3. Employing buffering techniques to smooth out playback
4. Adjusting rendering parameters dynamically based on network conditions

Q41) Describe the importance of access control mechanisms in a video streaming system.

Answer: Access control mechanisms are crucial for:

1. Ensuring content protection and preventing unauthorized access
2. Managing user permissions and restricting access based on roles
3. Improving system security and preventing data breaches
4. Enhancing user experience by providing personalized content access

Q42) How does the system handle video frame transmission?

Answer: Video frame transmission is managed by:

1. Using efficient transmission protocols to send video data
2. Implementing error correction mechanisms to recover lost data
3. Employing adaptive bitrate streaming to adjust transmission parameters

4. Ensuring smooth and continuous data flow during streaming

Q43) Explain the concept of automatic recovery mechanisms in a video streaming system.

Answer: Automatic recovery mechanisms involve:

1. Detecting system failures and implementing recovery strategies
2. Ensuring service continuity by handling errors gracefully
3. Improving system reliability by addressing issues promptly
4. Enabling quick recovery from errors and maintaining service quality

Q44) What are the benefits of using a memory usage monitoring system in a video streaming server?

Answer: Benefits include:

1. Ensuring efficient memory utilization and preventing memory leaks
2. Improving performance by optimizing resource usage
3. Identifying potential issues and addressing them proactively
4. Enhancing overall system reliability and user experience

Q45) How does the system handle video frame storage?

Answer: Video frame storage is managed by:

1. Using efficient storage algorithms to save video data
2. Implementing caching techniques to improve access speed
3. Employing buffering techniques to smooth out playback
4. Adjusting storage parameters dynamically based on network conditions

Q46) Describe the importance of content encryption in a video streaming system.

Answer: Content encryption is crucial for:

1. Ensuring content protection and preventing unauthorized access
2. Improving system security by encrypting data at rest and in transit
3. Enhancing user experience by providing secure content delivery
4. Preventing data breaches and ensuring compliance with regulations

Q47) How does the system handle video frame retrieval?

Answer: Video frame retrieval is managed by:

1. Using efficient retrieval algorithms to access video data
2. Implementing caching techniques to improve access speed
3. Employing buffering techniques to smooth out playback
4. Adjusting retrieval parameters dynamically based on network conditions

Q48) Explain the concept of enhanced logging in a video streaming system.

Answer: Enhanced logging involves:

1. Tracking system events and identifying issues
2. Helping debugging and improving system reliability

3. Providing detailed logs for analysis and troubleshooting
4. Ensuring continuous monitoring and proactive maintenance

Q49) Implement a robust error handling mechanism for network operations. How would you handle different types of failures?

Answer: Here's a comprehensive error handling implementation:

```
typedef enum {
    ERR_NONE = 0,
    ERR_NETWORK_TIMEOUT,
    ERR_CONNECTION_LOST,
    ERR_BUFFER_FULL,
    ERR_INVALID_STATE
} error_code_t;

typedef struct {
    error_code_t code;
    const char *message;
    int severity;
    void (*handler)(error_code_t);
} error_handler_t;

error_handler_t handlers[] = {
    {ERR_NETWORK_TIMEOUT, "Network timeout", 2, handle_timeout},
    {ERR_CONNECTION_LOST, "Connection lost", 3, handle_disconnect},
    {ERR_BUFFER_FULL, "Buffer full", 1, handle_buffer_full},
    {ERR_INVALID_STATE, "Invalid state", 3, handle_invalid_state}
};

void handle_error(error_code_t error) {
    for (int i = 0; i < sizeof(handlers)/sizeof(error_handler_t); i++) {
        if (handlers[i].code == error) {
            log_error(handlers[i].message, handlers[i].severity);
            if (handlers[i].handler != NULL) {
                handlers[i].handler(error);
            }
            break;
        }
    }
}

// Implementation example
int send_video_chunk(connection_t *conn, uint8_t *data, size_t len) {
    if (!is_valid_state(conn)) {
        handle_error(ERR_INVALID_STATE);
        return -1;
    }

    int retry_count = 0;
    while (retry_count < MAX_RETRIES) {
        if (send_with_timeout(conn, data, len, TIMEOUT_MS) < 0) {
```

```
            if (errno == ETIMEDOUT) {
                handle_error(ERR_NETWORK_TIMEOUT);
                retry_count++;
                continue;
            }
            handle_error(ERR_CONNECTION_LOST);
            return -1;
        }
        return 0;
    }
    return -1;
}
```

Key features:

1. Error categorization and severity levels
2. Custom handlers for different error types
3. Retry mechanism for recoverable errors
4. Logging and monitoring integration
5. Clean state management

Q50) Design a secure token-based authentication system for the streaming service. Include implementation details.

Answer: Here's a secure implementation:

```
#include <openssl/hmac.h>
#include <openssl/sha.h>

typedef struct {
    char user_id[64];
    int64_t timestamp;
    int64_t expires;
    char session_id[32];
    uint8_t hmac[SHA256_DIGEST_LENGTH];
} auth_token_t;

#define SECRET_KEY "your-secret-key"
#define TOKEN_VALIDITY 3600  // 1 hour

auth_token_t* generate_token(const char *user_id) {
    auth_token_t *token = malloc(sizeof(auth_token_t));
    if (!token) return NULL;

    strncpy(token->user_id, user_id, sizeof(token->user_id) - 1);
    token->timestamp = time(NULL);
    token->expires = token->timestamp + TOKEN_VALIDITY;
    generate_random_session_id(token->session_id);

    // Calculate HMAC
    HMAC_CTX *ctx = HMAC_CTX_new();
```

```
    HMAC_Init_ex(ctx, SECRET_KEY, strlen(SECRET_KEY), EVP_sha256(), NULL);
    HMAC_Update(ctx, (unsigned char*)token, offsetof(auth_token_t, hmac));
    HMAC_Final(ctx, token->hmac, NULL);
    HMAC_CTX_free(ctx);

    return token;
}

bool verify_token(auth_token_t *token) {
    if (!token) return false;

    // Check expiration
    if (time(NULL) > token->expires) {
        return false;
    }

    // Verify HMAC
    uint8_t expected_hmac[SHA256_DIGEST_LENGTH];
    HMAC_CTX *ctx = HMAC_CTX_new();
    HMAC_Init_ex(ctx, SECRET_KEY, strlen(SECRET_KEY), EVP_sha256(), NULL);
    HMAC_Update(ctx, (unsigned char*)token, offsetof(auth_token_t, hmac));
    HMAC_Final(ctx, expected_hmac, NULL);
    HMAC_CTX_free(ctx);

    return memcmp(token->hmac, expected_hmac, SHA256_DIGEST_LENGTH) == 0;
}
```

Security features:

1. HMAC-SHA256 for token integrity
2. Time-based token expiration
3. Random session IDs
4. Protection against token tampering
5. Secure memory handling

Section 8: Quick Questions

One-Liners:

Q51) What is the primary function of `pthread_cond_wait`? Answer: To block a thread until a specific condition is signaled.

Q52) Which socket option is commonly used to prevent the "Address already in use" error? Answer: `SO_REUSEADDR`.

Q53) What does CDN stand for in the context of video streaming? Answer: Content Delivery Network.

Multiple Choice Questions (MCQ):

Q54) Which protocol is generally preferred for low-latency video streaming? (a) TCP (b) UDP (c) HTTP (d) FTP **Answer:** (b) UDP

Q55) What is the main purpose of a mutex in multithreading? (a) Signaling between threads (b) Ensuring mutual exclusion to shared resources (c) Thread creation (d) Load balancing **Answer:** (b) Ensuring mutual exclusion to shared resources

Q56) Adaptive Bitrate Streaming (ABR) primarily aims to: (a) Encrypt the video content (b) Reduce server load (c) Adjust video quality based on network conditions (d) Compress video files **Answer:** (c) Adjust video quality based on network conditions

Fill in the Blanks:

Q57) In TCP, the three-way handshake involves SYN, SYN-ACK, and ____ packets. Answer: ACK

Q58) A common technique to handle packet loss in UDP without retransmission is ____ Error Correction. Answer: Forward

Q59) The `select()` system call is used for ____ I/O multiplexing. Answer: synchronous

Q60) Digital Rights Management (DRM) is used to control access to ____ content. Answer: copyrighted / protected

Section 9: More Quick Questions

One-Liners:

Q61) What system call is used to create a new thread in POSIX? Answer: `pthread_create()`

Q62) What is the purpose of the `volatile` keyword in C? Answer: To tell the compiler that a variable's value may change at any time without any action being taken by the code the compiler finds nearby.

Q63) Which command is used to check network connectivity to a host? Answer: `ping`

Q64) What does the `htons()` function do? Answer: Converts a short integer from host byte order to network byte order.

Q65) What is the default port number for HTTP? Answer: 80

Q66) What is the primary difference between `malloc()` and `calloc()`? Answer: `calloc()` initializes the allocated memory to zero, while `malloc()` does not.

Q67) What signal is typically sent by Ctrl+C in a terminal? Answer: SIGINT

Q68) What is the purpose of a semaphore? Answer: To control access to a shared resource by multiple processes or threads using a counter.

Q69) What does MTU stand for in networking? Answer: Maximum Transmission Unit.

Q70) What system call is used to wait for a child process to terminate? Answer: `wait()` or `waitpid()`

Q71) What is the function of `inet_pton()`? Answer: Converts an IP address string (IPv4 or IPv6) into its binary representation.

Q72) What is a deadlock? Answer: A situation where two or more threads are blocked forever, waiting for each other.

Q73) What is the purpose of the `shutdown()` socket call? Answer: To selectively disable reads and/or writes on a socket.

Q74) What does TTL stand for in an IP packet header? Answer: Time To Live.

Q75) What is the main advantage of non-blocking I/O? Answer: It allows a program to perform other tasks while waiting for I/O operations to complete.

Multiple Choice Questions (MCQ):

Q76) Which of the following is NOT a standard POSIX signal? (a) SIGINT (b) SIGTERM (c) SIGKILL (d) SIGWAIT **Answer:** (d) SIGWAIT

Q77) What is the typical size of an IPv4 address? (a) 32 bits (b) 64 bits (c) 128 bits (d) 16 bits **Answer:** (a) 32 bits

Q78) Which system call is used to associate a socket with a specific IP address and port? (a) `socket()` (b) `connect()` (c) `bind()` (d) `listen()` **Answer:** (c) `bind()`

Q79) In the producer-consumer problem, what synchronization primitive prevents the producer from adding to a full buffer? (a) Mutex (b) Semaphore (c) Condition Variable (d) Spinlock **Answer:** (c) Condition Variable (often used with a mutex)

Q80) Which command displays active network connections and listening ports? (a) `ifconfig` (b) `netstat` or `ss` (c) `traceroute` (d) `arp` **Answer:** (b) `netstat` or `ss`

Q81) What is the primary role of the `listen()` system call? (a) Accept incoming connections (b) Bind a socket to an address (c) Mark a socket as passive (listening for connections) (d) Create a new socket **Answer:** (c) Mark a socket as passive (listening for connections)

Q82) Which streaming protocol often uses RTP (Real-time Transport Protocol)? (a) HLS (b) DASH (c) RTSP (d) Smooth Streaming **Answer:** (c) RTSP

Q83) What does "endianness" refer to? (a) Network protocol type (b) Byte order within a multi-byte data type (c) Thread scheduling policy (d) File system format **Answer:** (b) Byte order within a multi-byte data type

Q84) Which of the following is a characteristic of UDP? (a) Connection-oriented (b) Guaranteed delivery (c) Ordered delivery (d) Low overhead **Answer:** (d) Low overhead

Q85) What is the purpose of `pthread_join()`? (a) Create a new thread (b) Terminate a thread (c) Wait for a specific thread to terminate (d) Detach a thread **Answer:** (c) Wait for a specific thread to terminate

Q86) Which HTTP method is typically used to request video segments in HLS? (a) POST (b) PUT (c) GET (d) DELETE **Answer:** (c) GET

Q87) What is "jitter" in the context of streaming? (a) Packet loss (b) Variation in packet arrival time (c) Low bandwidth (d) High latency **Answer:** (b) Variation in packet arrival time

Q88) Which component is responsible for resolving domain names to IP addresses? (a) DHCP Server (b) Gateway (c) DNS Server (d) Firewall **Answer:** (c) DNS Server

Q89) What is a "race condition"? (a) When threads run faster than expected (b) When the outcome of a computation depends on the non-deterministic scheduling of threads (c) When a thread pool runs out of threads (d) When network packets arrive out of order **Answer:** (b) When the outcome of a computation depends on the non-deterministic scheduling of threads

Q90) Which socket API function is used by a server to accept an incoming client connection? (a) `connect()` (b) `bind()` (c) `listen()` (d) `accept()` **Answer:** (d) `accept()`

Fill in the Blanks:

Q91) A ____ socket is used for connectionless communication. Answer: UDP (or datagram)

Q92) The `fcntl()` system call can be used to set a socket to ____ mode. Answer: non-blocking

Q93) In video compression, ____ frames contain the full image information. Answer: I (Intra)

Q94) The standard C library function for printing formatted output to stderr is ____. Answer: `fprintf`

Q95) A ____ is a synchronization primitive that allows only a fixed number of threads to access a resource concurrently. Answer: semaphore

Q96) The IP address `127.0.0.1` typically refers to the ____ interface. Answer: loopback

Q97) MPEG-DASH stands for Dynamic Adaptive Streaming over ____. Answer: HTTP

Q98) A thread that has finished execution but has not yet been joined is called a ____ thread. Answer: zombie (though less common term than zombie process)

Q99) The ____ system call duplicates an existing file descriptor. Answer: `dup()` or `dup2()`

Q100) TCP uses a ____ window mechanism for flow control. Answer: sliding

Q101) The `getaddrinfo()` function is used for network address and service ____. Answer: translation

Q102) A ____ lock is a mutex that prevents other threads from acquiring the lock if it's already held, without blocking. Answer: spin

Q103) HLS typically uses ____ files to list available media segments and bitrates. Answer: M3U8 (playlist)

Q104) The `errno` variable stores the error code of the last failed ____ call. Answer: system (or library function)

Q105) A ____ occurs when a process tries to access memory that has not been allocated to it. Answer: segmentation fault (segfault)

Q106) The `setsockopt()` function is used to set socket ____. Answer: options

Q107) _____ is a technique where a server pushes data to a client without the client explicitly requesting it. Answer: Server-Sent Events (SSE) or WebSockets

Q108) A _____ mutex allows the same thread to acquire the lock multiple times without deadlocking. Answer: recursive

Q109) The `poll()` system call is another mechanism for I/O _____. Answer: multiplexing

Q110) In video streaming, _____ refers to the initial delay before playback starts. Answer: buffering (or startup latency)