VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Design Framework CS K23

CNPM & HTTT CS23

Thiết kế giao thức truyền tin CS23

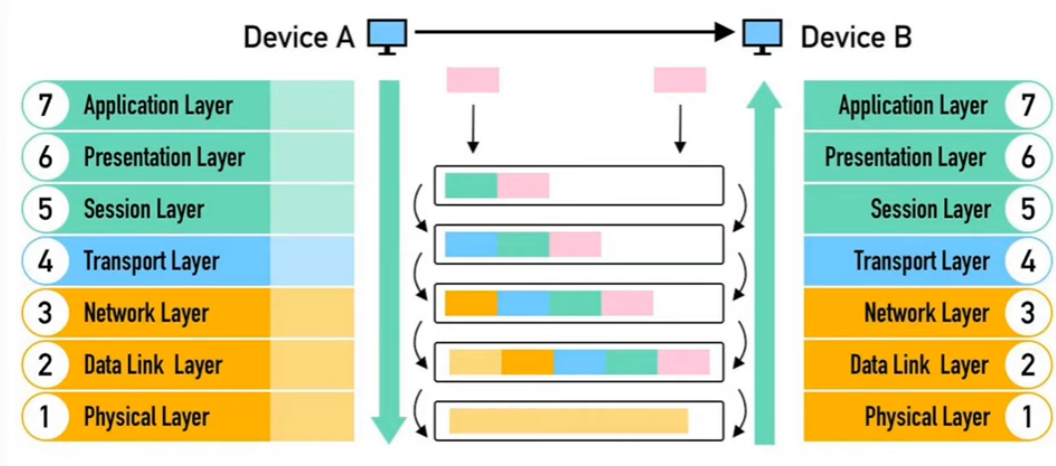
Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 5/2025

Mục lục

1 Protocol

Giao thức (**Protocol**) trong lĩnh vực công nghệ thông tin và truyền thông là **tập hợp các quy tắc và quy định** để hai hoặc nhiều thiết bị (máy tính, điện thoại, server,...) **giao tiếp với nhau một cách chính xác và hiệu quả**.



1. Khi dữ liệu được truyền qua mạng, nó sẽ **đi qua nhiều lớp (layers)**, mỗi lớp sử dụng **một giao thức khác nhau**. Mỗi lớp **đóng gói** dữ liệu từ lớp trên thành **một đơn vị truyền thông**, gọi là **PDU (Protocol Data Unit)**.
2. Tầng mà chúng ta có thể trực tiếp thiết kế và giao tiếp là Application Layer, với các giao thức điển hình như HTTP, FTP, SMTP, DNS, WebSocket, gRPC.
3. Tầng Transport chịu trách nhiệm vận chuyển dữ liệu đáng tin cậy hoặc nhanh chóng giữa các thiết bị, với các giao thức điển hình như TCP (đảm bảo độ tin cậy) và UDP (nhanh, không đảm bảo). Sử dụng **TCP** khi cần **truyền dữ liệu chính xác, đầy đủ, theo đúng thứ tự** (ví dụ: **web, email, truyền file**); dùng **UDP** khi cần **tốc độ cao, chấp nhận mất mát dữ liệu nhỏ** (ví dụ: **video call, game online, livestream**).
4. Các tầng còn lại phần lớn không cần quan tâm, vì nếu ta dùng các giao thức có sẵn ở tầng Application thì tầng Transport cũng đã được xử lý kèm theo và không cần tự thiết kế thêm.

Giao thức	Chức năng	Ví dụ thực tế	Vận chuyển
HTTP / HTTPS	Truyền nội dung web	Trình duyệt, REST API	TCP
FTP / SFTP	Truyền tệp	FileZilla, máy chủ lưu trữ	TCP
SMTP / IMAP / POP3	Gửi & nhận email	Gmail, Outlook	TCP
DNS	Tra tên miền ra địa chỉ IP	Truy cập chat.openai.com	UDP / TCP
WebSocket	Giao tiếp 2 chiều, thời gian thực	Chat, game online	TCP
MQTT	Giao thức nhẹ cho IoT	Thiết bị thông minh	TCP
gRPC	RPC tốc độ cao, dùng HTTP/2	Microservices, backend	TCP
SOAP	Truyền thông tin XML dạng RPC	Dịch vụ web cổ điển	TCP
CoAP	Giao thức nhẹ cho IoT (thay HTTP)	Hệ thống cảm biến	UDP

2 Mô tả giao thức CS23

Giao thức CS23 được thiết kế để các client gửi request (GET hoặc POST) đến một endpoint (URL), **kèm Bearer Token** và dữ liệu dạng JSON.

Server xử lý và **trả về một đoạn mã text (thường là kết quả hoặc trạng thái)**.

2.1 Request

Trong giao thức **CS23**, mỗi yêu cầu (request) từ client đến server được mô tả bằng một cấu trúc `Request`.

Cấu trúc này đại diện cho **một gói tin HTTP đơn giản**, bao gồm:

- Thông tin phương thức (`GET` / `POST`)
- Đường dẫn (URL)
- Token xác thực
- Dữ liệu gửi kèm theo dạng JSON đơn giản (key-value)

Giao thức này được thiết kế **nhẹ, dễ parse**, và phù hợp với các hệ thống xử lý lệnh nhanh, API gateway hoặc giao tiếp giữa client-server trong các hệ thống riêng.

```
struct Request
{
    string method;                      // Phương thức HTTP: "GET" hoặc "POST"
    string path;                        // Đường dẫn API (ví dụ: "/CS23/do-task")
    string token;                 // Token xác thực dạng Bearer
    map<string, string> body;           // Dữ liệu JSON dạng key-value
};
```

Ví Dụ

```
POST /CS23/add token_abc123
{
  "a": "5",
  "b": "7",
  "operation": "sum",
  "user_id": "1001"
}
```

2.2 Response

Trong giao thức **CS23**, `Response` là cấu trúc phản hồi từ server sau khi xử lý một `Request`. Nó truyền đạt kết quả cho client dưới dạng **mã trạng thái (status code)** và một đoạn **nội dung văn bản (text message)**.

Trạng thái của CS23

- **403 Forbidden:** Token xác thực sai hoặc không đủ quyền truy cập.
- **500 Internal Server Error:** Lỗi server do request sai cấu trúc hoặc lỗi xử lý nội bộ.
- **200 OK:** Yêu cầu xử lý thành công, trả về kết quả hợp lệ.

```
struct Response
{
    int status_code;
    std::string status_text;

```

```
    static Response unauthorized();
    static Response error();
    std::string toCS23Format() const;
};
```

```
200 VOTIEN OK NHA
403 Forbidden
500 Internal Server Error
```

2.3 Giao thức TCP

Server TCP đơn giản chạy vòng lặp nhận request theo giao thức CS23, xử lý rồi gửi response về client. Mình sẽ phân tích nhanh từng phần, đồng thời đưa ra một số lưu ý và đề xuất cải tiến nếu cần.

1. **Tạo socket TCP** (`socket()`).
2. **Gán địa chỉ và port** cho socket (`bind()`).
3. **Bắt đầu lắng nghe kết nối** (`listen()`).
4. **Vòng lặp chấp nhận kết nối mới** (`accept()`).
5. **Nhận dữ liệu từ client** (`recv()`).
6. **Phân tích cú pháp Request** (`RequestParser::parse()`).
7. **Xác thực token** (`AuthMiddleware::authorize()`).
8. **Phân quyền**
9. **Chọn controller xử lý theo Request** (`Router::getController()`).
10. **Xử lý Request và tạo Response** (`controller->handle()`).
11. **Gửi Response về client** (`send()`).
12. **Đóng kết nối client** (`close()`).

3 Xác thực (Authentication) và Phân quyền (Authorization)

3.1 Authentication

Authentication là quá trình kiểm tra danh tính người dùng, xem họ có thực sự là người mà họ khai báo không.

Ví dụ thực tế: Khi bạn **đăng nhập vào Facebook** bằng email và mật khẩu, hệ thống sẽ xác thực bạn.

- `Login` không cần xác thực
- Dựa vào `TokenStore`

```
class AuthMiddleware
{
public:
    static bool authorize(const Request &req);
};
```

- Nhập đúng => Xác thực thành công → bạn được đăng nhập.
- Nhập sai mật khẩu => Xác thực thất bại → không đăng nhập được.

Tóm lại: Xác thực = Đăng nhập.

3.2 Xác thực Bearer

Xác thực Bearer là một hình thức **xác thực qua token** (mã thông báo), được gửi kèm trong **HTTP Header** mỗi lần gọi API.

- "Bearer" nghĩa là "người cầm token" – **nếu bạn có token đúng, bạn được truy cập.**
- Token thường là một **chuỗi ký tự mã hóa** (ví dụ: JWT – JSON Web Token).

`Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6...`

```
enum class Role
{
    USER,
    ADMIN
};

class TokenStore
{
private:
    std::map<std::string, Role> tokenRoleMap;

    TokenStore() = default;

public:
    static TokenStore &instance();

    void store(const std::string &token, Role role);
    Role getRole(const std::string &token);
    bool contains(const std::string &token);
};
```

- `enum class Role`: Định nghĩa các loại vai trò người dùng (USER, ADMIN).
- `class TokenStore`: Lớp quản lý lưu trữ token và vai trò tương ứng.
- `tokenRoleMap`: Bản đồ ánh xạ từ token (chuỗi) sang vai trò (Role).
- `TokenStore()`: Constructor riêng tư để thực hiện singleton.
- `static TokenStore &instance()`: Trả về tham chiếu duy nhất của đối tượng TokenStore (singleton).
- `store(const std::string &token, Role role)`: Lưu token kèm vai trò vào bản đồ.
- `getRole(const std::string &token)`: Lấy vai trò tương ứng với token.
- `contains(const std::string &token)`: Kiểm tra token có tồn tại trong bản đồ hay không.

3.3 Authorization

Authorization Sau khi đã biết bạn là ai, hệ thống sẽ xác định bạn được quyền truy cập hoặc làm những gì.

Ví dụ thực tế: Khi bạn đã đăng nhập vào Facebook:

- Nếu là **người dùng thường**, bạn có thể đăng bài, bình luận.
- Nếu là **quản trị viên**, bạn có thêm quyền xóa bài viết, ban tài khoản.

Tóm lại: Phân quyền = Kiểm soát quyền truy cập.

Role	/user	/admin
USER	✓	
ADMIN	✓	✓

Bảng 1: *Bảng phân quyền truy cập theo role*

4 Controller và Router

4.1 Controller

Controller: Là thành phần xử lý logic nghiệp vụ khi nhận request; nhận dữ liệu từ request, thực thi xử lý (ví dụ: truy vấn database, tính toán), và trả về response phù hợp.

```
class Controller
{
public:
    virtual Response handle(const Request &req) = 0;
    virtual ~Controller() = default;
};
```

4.2 Router

Router: Là bộ phận phân phối request tới controller tương ứng dựa trên thông tin trong request (ví dụ: URL, method); đảm bảo mỗi request được xử lý đúng nơi.

```
class Router
{
public:
    static Controller *getController(const Request &req);
};
```

5 Thực Hành

1. Hiện thực các file yêu cầu có `TODO`
2. Test dựa trên **test.txt**
3. Đọc hiểu code xem có chỉnh gì tối ưu hay không, phải hiểu toàn bộ code
4. Đăng kí thêm một Controller mới
5. Đăng kí thêm một role mới và Controller với role mới đó
6. Đăng kí một xác thực mới là **API Key Authentication** tạo một **Key** và nếu truyền vào **token** có thể chạy được không cần login
7. Chụp anh gửi lên nhóm trong kênh nộp bài