Fast Ethernet Simulation project

switch.h

```
#ifndef SWITCH_H
#define SWITCH_H

#include "common.h"
#include <queue>
#include <map>
#include <mutex>
#include <string>

class Station; // Forward declaration of Station

class Switch {
private:
    std::queue<Frame> dataQueue;
    std::mutex mutex;
    std::map<std::string, Station*> stations;

public:
    // Register stations
    void registerStation(const std::string& stationId, Station* station);

    // Process frames
    void processFrames();

    // Forward frames
    void forwardFrame(const Frame& frame);
```

```
    // Logging
    void logActivity(const std::string& message) const;

    // Accessors
    std::mutex& getMutex();
    std::queue<Frame>& getDataQueue();
};

#endif // SWITCH_H
```

station.h

```
#ifndef STATION_H
#define STATION_H

#include "common.h"
#include "switch.h"
#include <queue>
#include <string>

class Station {
private:
    std::string id;
    std::queue<Frame> sendQueue;
    std::queue<Frame> receiveQueue;

public:
    explicit Station(const std::string& stationId);

    // Send frames
```

```
    void sendFrame(const Frame& frame, Switch& ethernetSwitch);

    // Receive frames
    void receiveFrame(const Frame& frame);

    // Process send
    void processSend(Switch& ethernetSwitch);

    // Logging
    void logActivity(const std::string& message) const;
};

#endif // STATION_H
```

station.h

```
#ifndef STATION_H
#define STATION_H

#include "common.h"
#include "switch.h"
#include <queue>
#include <string>

class Station {
private:
    std::string id;
    std::queue<Frame> sendQueue;
    std::queue<Frame> receiveQueue;
```

```
public:
    explicit Station(const std::string& stationId);

    // Send frames
    void sendFrame(const Frame& frame, Switch& ethernetSwitch);

    // Receive frames
    void receiveFrame(const Frame& frame);

    // Process send
    void processSend(Switch& ethernetSwitch);

    // Logging
    void logActivity(const std::string& message) const;
};

#endif // STATION_H
```

station.cpp

```
#include "station.h"
#include <fstream>

Station::Station(const std::string& stationId) : id(stationId) {}

void Station::sendFrame(const Frame& frame, Switch& ethernetSwitch) {
    std::lock_guard<std::mutex> lock(ethernetSwitch.getMutex());
    ethernetSwitch.getDataQueue().push(frame);
    logActivity("Sent frame " + std::to_string(frame.sequenceNumber) + " to switch.");
}
```

```
void Station::receiveFrame(const Frame& frame) {
    receiveQueue.push(frame);
    logActivity("Received frame " + std::to_string(frame.sequenceNumber) + " from " + frame.source);
}

void Station::processSend(Switch& ethernetSwitch) {
    while (!sendQueue.empty()) {
        Frame frame = sendQueue.front();
        sendQueue.pop();
        sendFrame(frame, ethernetSwitch);
    }
}

void Station::logActivity(const std::string& message) const {
    std::ofstream logFile(id + "_log.txt", std::ios::app);
    if (logFile.is_open()) {
        logFile << message << std::endl;
        logFile.close();
    }
}
```

switch.cpp

```
#include "switch.h"
#include "station.h"
#include <fstream>

void Switch::registerStation(const std::string& stationId, Station* station) {
    stations[stationId] = station;
}
```

```
void Switch::processFrames() {
    while (!dataQueue.empty()) {
        std::lock_guard<std::mutex> lock(mutex);
        Frame frame = dataQueue.front();
        dataQueue.pop();

        logActivity("Forwarding frame " + std::to_string(frame.sequenceNumber) +
                    " from " + frame.source + " to " + frame.destination);

        if (stations.find(frame.destination) != stations.end()) {
            stations[frame.destination]->receiveFrame(frame);
        } else {
            logActivity("Destination " + frame.destination + " not found.");
        }
    }
}

void Switch::forwardFrame(const Frame& frame) {
    std::lock_guard<std::mutex> lock(mutex);
    dataQueue.push(frame);
}

std::mutex& Switch::getMutex() {
    return mutex;
}

std::queue<Frame>& Switch::getDataQueue() {
    return dataQueue;
```

```
}

void Switch::logActivity(const std::string& message) const {
    std::ofstream logFile("switch_log.txt", std::ios::app);
    if (logFile.is_open()) {
        logFile << message << std::endl;
        logFile.close();
    }
}
```

main.cpp

```
#include "station.h"
#include "switch.h"

int main() {
    Switch ethernetSwitch;

    Station station1("SP1"), station2("SP2"), station3("SP3");
    ethernetSwitch.registerStation("SP1", &station1);
    ethernetSwitch.registerStation("SP2", &station2);
    ethernetSwitch.registerStation("SP3", &station3);

    Frame frame1(1, "SP1", "SP2", "Hello SP2!", 1);
    Frame frame2(2, "SP2", "SP3", "Hello SP3!", 2);

    station1.sendFrame(frame1, ethernetSwitch);
    station2.sendFrame(frame2, ethernetSwitch);

    ethernetSwitch.processFrames();
```

```
    return 0;
}
#include "station.h"
#include "switch.h"

int main() {
    Switch ethernetSwitch;

    Station station1("SP1"), station2("SP2"), station3("SP3");
    ethernetSwitch.registerStation("SP1", &station1);
    ethernetSwitch.registerStation("SP2", &station2);
    ethernetSwitch.registerStation("SP3", &station3);

    Frame frame1(1, "SP1", "SP2", "Hello SP2!", 1);
    Frame frame2(2, "SP2", "SP3", "Hello SP3!", 2);

    station1.sendFrame(frame1, ethernetSwitch);
    station2.sendFrame(frame2, ethernetSwitch);

    ethernetSwitch.processFrames();

    return 0;
}
```