Documentatia **Implementation “XOR in the Air”**

Introduction

Wireless networks these days are present almost in every corner of every big city and not only there. They are tools created by humankind that are indispensable, they provide means for mobility, city-wide Internet connectivity, distributed sensing, and outdoor computing. Current wireless implementations, however, suffer from a severe throughput limitation and do not scale to dense large networks.

To give you a feel for how this technology works, we start with a fairly simple example. Consider the scenario in the image below, where we have two persons, Oscar and Cris, who want to send one pair of messages to each other via a router. In the actual configuration of the code, Oscar sends the package to the router (relay), which will be sent as well to Cris, also The message (package) from Cris will be sent in the same way to Oscar. This process requires 4 data transmissions. Now consider a coding approach to this problem. Let's consider that Oscar and Cris will send the packages to the relay, which will apply XOR operation for the 2 packages received and it will send back the processed version. Oscar and Cris can obtain their packages by applying once again an XOR operation with the received package and their package. This method only requires 3 data transmissions instead of 4. The saved transmission can be now utilized for sending new data, enhancing the wireless debit.

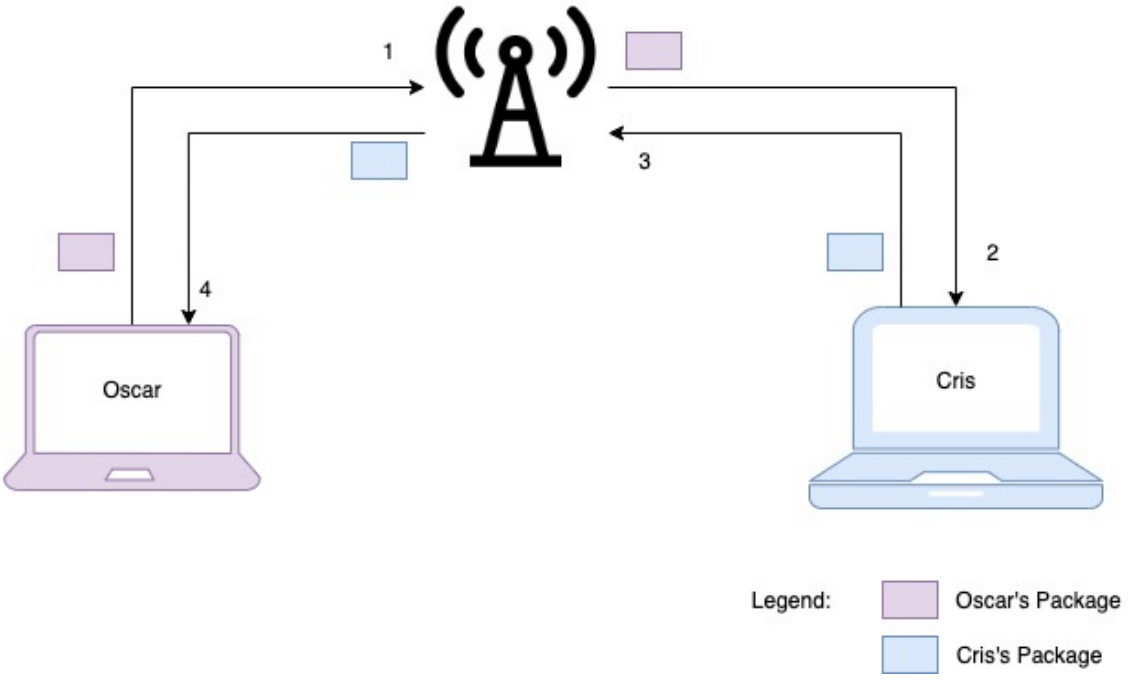

Figure 1

Implementation

Imports

```
#include <string>
#include <iostream>
#include <vector>
```

Persons Structs

```
struct FirstPerson {
    // The first person that's gonna send through a message
    std::string nume = "Oscar";
    std::string data;

    std::vector<std::pair<std::string, std::string>> received_data;

    void send(Relay& r);
    void read_user_data() {
        std::cout << "Add the data that Oscar's gonna send to Cris \n";
        std::cout << "Oscar: ";
        std::getline(std::cin, data);
    }
};


struct SecondPerson {
    // The second person that's gonna send through a message
    std::string nume = "Cris";
    std::string data;

    std::vector<std::pair<std::string, std::string>> received_data;

    void send(Relay& r);
    void read_user_data()
    {
        std::cout << "Add the data that Cris's gonna send to Oscar \n";
        std::cout << "Cris: ";
        std::getline(std::cin, data);
    }
};
```

Relay Struct

```
struct Relay {
    // methods send_first_person and send_second_person deals with the
    // selective sending of an answer from the relay to either
    // Oscar or Cris, for the simulation of xor in the air

    // data_relay contains the messages the relay receives
    std::vector<std::pair<std::string, std::string>> data_relay;
```

```
    // send_first_person
    // The data Oscar's receive from Cris
    // Xb = Xa + Xb - Xa
    // we parse the map and select only the elements that were send from Cris
    // for the simulation of the above ecuation
    void send_first_person(FirstPerson& a, SecondPerson& b) {
        for (auto pair : data_relay) {
            if (pair.first.find(b.nume) != std::string::npos)
            {
                a.received_data.push_back(std::make_pair(pair.first, pair.second));
                std::cout << "Relay sends a message to " << a.nume << " from " <<
b.nume << "\n";
            }
        }
    }

    // send_second_person
    // alike the above solution
    void send_second_person(FirstPerson& a, SecondPerson& b) {
        for (auto pereche : data_relay) {
            if (pereche.first.find(a.nume) != std::string::npos) {
                b.received_data.push_back(std::make_pair(pereche.first,
pereche.second));
                std::cout << "Relay sends a message to " << b.nume << " from " <<
a.nume << "\n";
            }
        }

    }
};
```

The send function for the persons

```
// The two methods of FirstPerson and SecondPerson deals with sending the data
// to the relay respectively Xa + Xb where Xa are the data from FirstPerson
// and Xb are the data from SecondPerson they are stored into a variable
// of type std::map from the relay
void FirstPerson::send(Relay& r) {
    r.data_relay.push_back(std::make_pair(nume, data));
    std::cout << "\n The relay received a message from: " << nume << "\n";
};

void SecondPerson::send(Relay& r) {
    r.data_relay.push_back(std::make_pair(nume, data));
    std::cout << "\n The relay received a message from: " << nume << "\n";
};
```

Main

```
int main() {
    FirstPerson a;
    SecondPerson b;
```

```
    Relay r;
    char stop = 'y';
    while (stop == 'y') {
        a.read_user_data();
        a.send(r);

        b.read_user_data();
        b.send(r);

        std::cout << "\n\t If you want to continue the conversation hit the key y else
n: ";

        stop = std::cin.get();
        std::cin.ignore();

    }
    r.send_first_person(a, b);
    r.send_second_person(a, b);
    std::cout << "\n Oscar received: " << "\n";
    for (auto pereche : a.received_data)
        std::cout << pereche.first << ": " << pereche.second << "\n";

    std::cout << "\n Cris received: " << "\n";
    for (auto pereche : b.received_data)
        std::cout << pereche.first << ": " << pereche.second << "\n";
}
```

Student: Gal Oscar
Specialization: Tehnologii Multimedia
Class: Tehnici avansate de codare și control al fluxului de date în rețelele de telecomunicații
Year: 2021