Anton Christoffersson

antchr-4@student.ltu.se

Laborationsrapport

D0036D, Nätverksprogrammering

Labb 4

8 juni 2016

Innehållsförteckning

[Problemspecifikation 2](#_Toc453327349)

[Användarhandledning 3](#_Toc453327350)

[Algoritmbeskrivning 4](#_Toc453327351)

[Klient 4](#_Toc453327352)

[Server 4](#_Toc453327353)

[Systembeskrivning 5](#_Toc453327354)

[Klient 5](#_Toc453327355)

[Server 5](#_Toc453327356)

[Allmänt 5](#_Toc453327357)

[Lösningens begränsningar 6](#_Toc453327358)

[Diskussion 7](#_Toc453327359)

[Testkörningar 8](#_Toc453327360)

# Problemspecifikation

Den här labben var uppgiften att implementera en spelklient med grafiskt gränssnitt och en server som kan hantera flera spelklienter i C++ eller C. Kopplingen mellan klient och server skulle vara TCP och i slutet av labben skulle all data som skickades emellan klient och server vara krypterad med RC4. För att genomföra labben skulle vi använda Winsock utan någon api som hanterar detta. Servern skulle även kompileras och köras på en Linux dator.

# Användarhandledning

Projektet är uppdelat i tre stycken mapper:

* Labb4ServerLinux – Innehåller alla .cpp och .h filer samt en make fil för att bygga projektet på Linux och en redan byggd version av servern vid namn ”MyServerShaggy”.
* Labb4Server – Innehåller ett Visual studio projekt med servern på då den både fungerar på Windows och Linux. Bara att starta och köra som ett vanligt VS projekt.
* Labb4Klient – Innehåller ett Visual studio projekt med klienten. För att köra klienten utan kryptering för test mot t.ex. den servern vi fick till kursen så behöver man kommentera ut krypteringen på 3 ställen i koden:
  1. myWindow.cpp: myWindow::sendMove() Rad 203.
  2. networklistener.cpp: NetworkListener::listener() rad 35.
  3. networklistener.cpp: NetworkListener::join() rad 162.

Seden när klienten ska köras så startar man först en och kollar i kommando fönstret så att man ser att den har lyckats ansluta till servern och när man startar den andra klienten är det viktigt att man väntar till att båda klienterna får ”opponent joined” meddelandet i konsolen innan man rör någon av klienterna.

# Algoritmbeskrivning

Har använd Doxygen för att generar klassdiagram och info. Dom ligger i respektive mapp. Bara att öppna index.html.

## Klient

På klient sidan så används treornas grafikmotor SM4K för det grafiska gränssnittet. Fungerar väldigt bra för den här labben och är väldigt lätt att sätta sig in i och snabbt få ett fungerande spel. Skulle rekommendera denna till nästa år studenter då QT kan vara lite krångligt att komma in i.

Förutom SM4K användes inga speciella lösningar eller algoritmer för den här labben. Det som händer när klienten startas är att den försöker ansluta till servern på den angivna ip:n och porten och om detta går så startas en tråd med en loop som lyssnar på och hanterar inkommande meddelanden ifrån servern.

## Server

Servern fungerar snarlikt klienten förutom att den sätter upp winsock och sedan lyssnar på inkommande anslutningar. Vid anslutning så skapar den en ny tråd som lyssnar på den inkommande anslutningens socket och hanterar dess meddelanden. Den lägger även alla anslutna klienter i en lista så att dom lätt kan kommas åt i programmet.

# Systembeskrivning

## Klient

**Struct Client**; är en strukt som används för att hålla koll på klientens id, seq\_nr och koordinater. Det finns två på varje klient. En för sig själv och en för fiende klienten.

**Class NetworkListener**; Startas på en tråd för att lyssna på inkommande meddelanden och hantera dom. Flyttar även spelaren via Client strukten vid accepterat move meddelande ifrån servern. Tar även hand om att skicka join meddelandet vid anslutning till server.

**Class myWindow**; Den viktiga funktionen i den här klassen är initializeNetwork som sätter upp kryptering och winsock för anslutning till servern.

## Server

**Class Sever**; Sätter upp winsock och lyssnar på inkommande klienter. Skapar en ny tråd av Class Connecitons för varje ansluten klient och lägger till objektet i en std::vector som håller koll på alla anslutna klienter. Tar även hand om att ta bort objekten ifrån vectorn om en klient skulle avsluta sin anslutning.

**Class Connecitons**; Har en funktion Reciver() som lyssnar efter och hanterar inkommande meddelanden. En SendMsg() funktion som krypterar och skickar angivet paket. En OnJoin() som hanterar vad som ska hända när servern får ett Join meddelande ifrån en klient.

## Allmänt

**Class RC4;** Både servern och klienten har en klass som heter RC4 som hanterar all kryptering och dekryptering med funktionerna Encrypt() och Decrypt() som i sin tur använder funktionen PRGA()**.** Klassen har även en funktion KSA som används när en ny klient skapas för att generera den första krypteringen av nyckeln som används i krypteringen.

# Lösningens begränsningar

Begränsningar på lösningen är att den inte är så använder vänlig då båda spelarna startar på samma ställe, det gör det lite svårt att ser när man börjar röra sig.

En annan begränsning är att sleep har använts för att inte skicka move meddelanden för ofta då det orsakade problem med krypteringen om en klient inte fått ett meddelande och hamnar efter i krypteringsprocessen då detta gör att den inte längre kan läsa av dom krypterade meddelandena som den får.

# Diskussion

Det har varit en väldigt krävande labb och jag hade egentligen velat ha mer tid till den originellt då det var svårt att hinna med labben under tiden som den var avsedd att göras.

Jag tycker även att labben var lite för öppen då det var lite svårt att bestämma sig för vad man skulle använda för grafiskt gränssnitt och det tog onödigt mycket tid att försöka lära sig QT istället för att man fick en plattform eller inte behövde göra det grafiskt.

Tycker även att göra ett grafiskt spel var lite för krävande för labben tillsammans med kryptering. Hade hellre koncentrerat mig på nätverks och krypteringsdelen av labben mer istället för att lägga ner tid på en ”spelidé” och implementera den grafiskt. Men det var helt klart lärorikt och roligt att göra men det satta ganska mycket press.

Väldigt kul att lära sig djupare i nätverksprogrammering då det är ett av de ämnen jag är mest intresserad av och kul att använda en kryptering man känner igen sen tidigare och som faktiskt använts i verkliga livet.

# Testkörningar

Hänvisas till redovisning.