Project Telecom/Gedistribueerde Systemen

Bart Braem – Johan Bergs – Tom De Schepper

2015-2016

1 Introductie

In het geïntegreerde project van de vakken Telecommunicatiesystemen en Gedistribueerde Systemen moet een gedistribueerde applicatie gebouwd worden bovenop een netwerk waarbij je een protocol implementeert. Dit laatste gebeurt aan de hand van de Click Modular Router. Dit document bespreekt zowel de opgave voor het Telecommunicatiesystemen deel als dat voor het luik Gedistribueerde Systemen.

2 Telecommunicatie Systemen

2.1 Opgave: RSVP in Click

Implementeer in userlevel Click RSVP volgens de geannoteerde RFCs 2205 en 2210 die beschikbaar zijn op Blackboard. Houd je daarbij aan onderstaande richtlijnen en alle vereiste opmerkingen uit de RFC. Houd er rekening mee dat *admission control* niet moet worden ondersteund.

2.2 Richtlijnen

2.2.1 Elementen

Je elementen moeten verantwoord *push*, *pull* of *agnostic* zijn. Je moet ons kunnen uitleggen waarom je welke keuze maakte. Je elementen moeten bovendien compileren en werken als ze in */elements/local* worden geplaatst op een standaard Click 2.0.1 distributie.

2.2.2 Vragen

In geval van vragen over Click stel je die NIET op de Click mailinglist. Op Blackboard staat een forum waarop je vragen kan stellen en een FAQ kan nalezen.
We sturen mededelingen enkel via Blackboard. Lees dus je studentenmail.

Als je voor een vraag liever even langskomt, stuur dan op voorhand een mail naar ons.

2.3 Evaluatie

2.3.1 Minimumvereisten om te slagen

Er zijn een aantal *minimale vereisten* waaraan jullie project moet voldoen om te kunnen slagen voor dit deel van het practicum. Als aan één van deze vereisten niet is voldaan, kan je nooit de helft van de punten halen:

- Je code moet compileren op het referentieplatform.
- Je code moet draaien op het referentieplatform door de gecompileerde click binary op te roepen met als argument het scriptje, voorzien voor de evaluatie (*ipnetwork.click*).
- Je elementen worden in de map */elements/local* (of in een submap daarvan) geplaatst en moeten op die manier compileren en werken. Elementen die je hebt gewijzigd of toegevoegd buiten deze map, worden sowieso verwijderd of vervangen door de originele elementen.
- Al je pakketten dienen correct te zijn volgens Wireshark. Met "correct" wordt bedoeld dat op zijn minst:
  - **Alle checksums** correct zijn.
  - Er geen pakketten zijn die geheel of deels niet door wireshark worden herkend.
  - De pakketformaten overeen komen met de specificaties in de RFC.
- In je code komen geen hardcoded IP, MAC of andere adressen voor.

2.3.2 Waar moet je verder op letten?

Als je project voldoet aan de minimumvereisten, wordt er voornamelijk op de volgende zaken gelet om je resultaat te beoordelen:

- Heb je de RFC correct gevolgd? Dit wil zeggen: volg je de RFC en heb je geen eigen versie ervan geïmplementeerd? Als er een bepaald gedrag in de RFC wordt vermeld, houd je je daar dan aan?
- Zijn alle velden/vlaggen/... in al je pakketten correct gezet (zie ook vorig punt)?
- Heb je inzicht in de RFC en in je code? Kan je ons uitleggen waarom iets is zoals je het hebt geïmplementeerd?

2.3.3 Tussentijdse evaluatie

We verwachten dat je volgende features hebt geïmplementeerd en kan demonstreren:

- het sturen van alle benodigde RSVP berichten met behulp van handlers (je hoeft dus nog geen timers te starten of onderhouden)
- op pakketten kan de TOS byte gezet worden met een handler
- een apart Click script toont hoe priority scheduling is geïmplementeerd met enkel bestaande Click elementen

Bijkomend verwachten we dat je (beknopt) redelijke waarden voorstelt voor de TSpec en RSpec parameters, om het gevraagde scenario te ondersteunen. Uiteraard mag je al extra functionaliteit insturen en tonen, als je al verder staat.

2.3.4 Eindevaluatie

We verwachten dat je, bovenop de vereisten voor de tussentijdse evaluatie, alle opgegeven stukken uit de RFC hebt geïmplementeerd en volgende features kan demonstreren:

- Alle berichten triggeren nu ook de relevante timers
- De datastream die quality of service vereist krijgt dat ook
- DiffServ gebeurt enkel op basis van klassen, gedefinieerd door de TOS byte in de IP headers maar gereserveerd op basis van poort en protocolnummer.

3 Gedistribueerde Systemen

3.1 Opgave: Chat applicatie met Apache Avro

3.1.1 Overzicht

Het doel van dit project is het ontwerpen en implementeren van een gedistribueerde video chat applicatie (vergelijkbaar met een vereenvoudigde versie van Skype of Facebook Messenger). De applicatie moet het mogelijk maken dat gebruikers verbinden met een chat server en eens ze verbonden zijn kunnen communiceren met anderen gebruikers.

Concreet moet een gebruiker een overzicht kunnen krijgen van de andere personen die verbonden zijn, de publieke chat room (e.g. groepsgesprek) kunnen joinen of een privé conversatie beginnen met één andere gebruiker. Binnen deze privé conversaties tussen twee gebruikers moet het ook mogelijk zijn om video te streamen naar elkaar. Alle communicatie tussen client en server en tussen clients onderling dient te gebeuren door gebruik te maken van Apache Avro. Verder wordt een zekere foutentolerantie verwacht.

3.1.2 Server

Volgende functionaliteit wordt verwacht voor de server:

- Een client moet kunnen registreren bij de server en de server kent een unieke id/gebruikersnaam toe.
- De server houdt een lijst bij van alle verbonden gebruikers en hoe deze bereikt kunnen worden.
- De server staat in voor het afhandelen van het joinen van de publieke chat room en het verzenden van berichten binnen dit groepsgesprek.
- De server staat in voor de verzending van uitnodigingen voor privé conversaties en het opstarten van zo'n gesprekken.
- De server staat **niet** in voor het verzenden van berichten of video tijdens een privé gesprek, deze communicatie verloopt direct tussen twee clients.
- Wanneer een client de applicatie verlaat, moeten zijn gegevens uit de lijst van verbonden gebruikers worden verwijderd.

3.1.3 Client

Volgende functionaliteit wordt verwacht voor de client:

- Automatische registratie bij de server.
- Nadat een client geregistreerd is, kan hij commando's in geven.
- Een gebruiker kan een lijst opvragen van andere online gebruikers.
- Een gebruiker kan besluiten de publieke chat room te joinen en boodschappen uit te wisselen met andere personen in deze chat room.
- Een gebruiker kan een uitnodiging sturen naar een andere gebruiker voor het opstarten van een privé gesprek. Omgekeerd moet een gebruiker een uitnodiging kunnen ontvangen en deze accepteren of weigeren. Indien een gebruiker de uitnodiging accepteert, moet er een privé gesprek worden aangemaakt en *zal de communicatie tussen de twee clients direct gebeuren en niet meer langs de server lopen.* Bij de creatie van een privé gesprek moet de server dus de correcte referenties bezorgen

aan de clients zodat deze directe verbinding mogelijk wordt. Het moet mogelijk zijn om de uitnodigingen voor privé gesprekken altijd te versturen of ontvangen, ook wanneer de gebruiker reeds in een gesprek of de publieke chat room zit.

- Het is **niet** mogelijk om twee of meer privé conversaties op het zelfde moment te voeren of om een privé conversatie te voeren terwijl je ook aanwezig bent in de publieke chat room. Wanneer een gebruiker een uitnodiging tot een privé conversatie accepteert terwijl hij of zij in de chat room zit, wordt de publieke chat room afgesloten.
- Binnen een privé gesprek moet het mogelijk zijn om aan video streaming te doen. Meer info hier over volgt in Sectie 3.1.6.
- Tenslotte moet het mogelijk zijn voor een client om de publieke chat room te verlaten, een privé gesprek te stoppen of de chat applicatie te verlaten.

3.1.4 Communicatie

In de theorielessen en in de introductie tot Apache Avro werd besproken dat er twee soorten van communicatie mogelijk zijn tussen client en server: synchroon en asynchroon. We verwachten dat jullie inzicht hebben in hoe deze communicatievormen juist werken en dat jullie de methodes dan ook correct toepassen binnen deze opdracht.

Hou ook rekening met het feit dat het Avro RPC protocol in essentie client-driven is (i.e., pull-based communicatie), zoals uitgelegd tijdens de introductie. In deze opdracht zal er eveneens gebruik moeten worden gemaakt van push-based communicatie (het oproepen van methodes op de client). Hiervoor verwachten we dat clients een eigen server starten, zodat er een Avro protocol object beschikbaar wordt voor invocaties.

Verder verwachten we dat een server en een client op verschillende computers kunnen functioneren en met elkaar communiceren. Hiervoor kan er best iedere keer bij het opstarten van de server en clients steeds gevraagd worden om een ip adres met poort nummer in te geven. Hou er hierbij ook rekening mee dat meerdere `SaslSocketServers` op hetzelfde machine een verschillende poort moeten gebruiken.

3.1.5 User interface

Voor de interactie met de gebruiker is het niet nodig om een GUI (grafische user interface) te voorzien maar mag er gebruikt worden gemaakt van commando's die worden ingegeven in een terminal. Hiervoor kan er - optioneel - gebruik gemaakt worden van de *Cliche* library. Dit is een Java library die op een heel eenvoudige manier de creatie van CLIs (command-line user interface) mogelijk maakt. Documentatie omtrent deze library kan hier gevonden worden: `https://code.google.com/p/cliche/wiki/Manual`.

Hoewel niet nodig is het uiteraard wel toegestaan om toch zelf een eenvoudige GUI te voorzien, in de plaats van gebruik te maken van een CLI. Voorbeelden van libraries die hierbij gebruikt kunnen worden zijn AWT, Swing, JavaFX of QT Jambi.

3.1.6 Video streaming

In een privé gesprek moet het mogelijk zijn om aan video streaming te doen. Aangezien de computers in het labo en de computerklassen niet uitgerust zijn met een webcam, zal in de plaats hiervan een video file worden verstuurd. Het is dus het idee dat de eerste gebruiker een video file inleest (de locatie van dit bestand mag hard coded in

de code staan) en frame per frame doorgeeft aan de tweede gebruiker. Deze gebruiker moet de ontvangen frames steeds weergeven waardoor je het idee van video streaming krijgt. Informatie over het weergeven van afbeeldingen in Java kan o.a. hier gevonden worden: `https://docs.oracle.com/javase/tutorial/2d/images/`. Op het internet zijn er echter ook andere methodes te vinden die gebruik kunnen worden.

3.1.7 Foutentolerantie

Er wordt verwacht dat er een zekere foutentolerantie wordt geïmplementeerd en er geen crashes kunnen worden veroorzaakt:

- Indien de server offline gaat, moeten de gebruikers een boodschap te zien krijgen in hun terminal. Clients die bezig waren met een privé gesprek moeten dit gesprek kunnen verder zetten zonder problemen. Gebruikers die in de publieke chat room zaten of nog geen gesprek voerden, zullen niets meer kunnen doen. Indien de server weer wordt opgestart moeten de verschillende clients vanzelf weer verbinden met de server.
- Indien een client wegvalt uit een privé gesprek krijgt de andere gebruiker een boodschap en wordt dit gesprek stop gezet.
- In de andere gevallen hoeft er geen speciale boodschap worden getoond indien een client offline gaat (bv. uit groepsgesprek).
- Hou er rekening mee dat de lijst van verbonden clients regelmatig moet worden geüpdatet, zodat deze accuraat blijft.

3.2 Evaluatie

3.2.1 Minimumvereisten om te slagen

Er zijn een aantal *minimale vereisten* waaraan jullie project moet voldoen om te kunnen slagen voor dit deel van het practicum. Als aan één van deze vereisten niet is voldaan, kan je nooit de helft van de punten halen:

- Je code moet compileren op het referentieplatform. Dit wilt ook zeggen dat alle nodige libraries aanwezig dienen te zijn.
- Je code moet kunnen worden uitgevoerd op het referentieplatform. Om dit mogelijk te maken dient bij de inzendingen steeds een manual te worden toegevoegd.
- Voor communicatie tussen de verschillende componenten dient gebruik te worden gemaakt van Apache Avro en mag niet gebruik worden gemaakt van een andere technologie.

Let op: het halen van deze minimumvereisten betekent niet onmiddellijk dat je geslaagd zal zijn voor dit project.

3.2.2 Waar moet je verder op letten?

Als je project voldoet aan de minimumvereisten, wordt er voornamelijk op de volgende zaken gelet om je resultaat te beoordelen:

- In de eerste plaats wordt er beoordeeld op de functionaliteit die geïmplementeerd is. Dit willen zeggen dat er wordt gekeken of alle functionaliteiten zijn voorzien en deze correct werken.

- Verder is het ook belangrijk om inzicht te hebben in de gebruikte technologiën en concepten en in je code. Zo moet je kunnen uitleggen waarom iets is zoals je het hebt geïmplementeerd.

3.2.3 Tussentijdse evaluatie

We verwachten dat je volgende features hebt geïmplementeerd en kan demonstreren:

- Een client moet kunnen registreren bij de server en de applicatie correct kunnen verlaten.
- Een client moet een lijst kunnen opvragen van de verbonden clients.
- Een client moet de publieke chat room kunnen joinen en daarin boodschappen versturen.

Uiteraard mag je al extra functionaliteit insturen en tonen, als je al verder staat.

3.2.4 Eindevaluatie

We verwachten dat je, bovenop de vereisten voor de tussentijdse evaluatie, alle opgegeven functionaliteit hebt geïmplementeerd en kan demonstreren:

- Een client moet kunnen registreren bij de server en de applicatie correct kunnen verlaten.
- Een client moet een lijst kunnen opvragen van de verbonden clients.
- Een client moet de publieke chat room kunnen joinen en daarin boodschappen versturen.
- Er kan een privé gesprek worden gevoerd tussen twee gebruikers.
- In een privé gesprek moet er aan video streaming kunnen worden gedaan.
- Fault tolerance.

4 Integratie

De integratie van beide delen gebeurt als volgt: in het deel Gedistribueerde Systemen zal een video streaming sessie moeten worden opgezet. Deze videosessie moet, via het gedeelte in Click, via RSVP een padreservatie opzetten. Deze reservatie moet succesvol verlopen, zodat de video stream prioriteit krijgt. Het is **niet** de bedoeling dat je dit manueel, buiten de applicatie van Gedistribueerde Systemen opzet. Het is **wel** de bedoeling dat het gedistribueerde systeem op één of andere manier aan Click doorgeeft dat deze reservatie moet worden opgezet.

Zodra dit alles in orde is, moet je met behulp van achtergrondverkeer (bijv. via het `iperf` commandox) aantonen dat het Click gedeelte naar behoren werkt: de video krijgt prioriteit ten opzichte van het achtergrondverkeer.

5 Praktische richtlijnen

5.1 Groepswerk

Dit project los je per twee op. Post je namen en studentennummers in de samenwerkingsthread van het vak Telecommunicatie Systemen op het Blackboard forum vóór de deadline. Problemen met de samenwerking meld je zo snel mogelijk, anders kunnen we daar geen rekening mee houden.

5.2 Referentieplatform

Het referentieplatform zijn de computers in het computerlabo op de tweede verdieping. Op de evaluatie wordt verwacht dat jullie code compileert en draait op deze VM.

6 Inzending en deadlines

6.1 Inzending

We verwachten dat je bij elke inzending een manual of README meelevert waarin duidelijk staat hoe we jou code moeten gebruiken om aan te tonen dat die doet wat wordt verwacht. Voor het Click project mag je uiteraard zelf geschreven scriptjes mee indienen als dat het aantonen van bepaalde functionaliteit vergemakkelijkt. Naast een manual moet voor het Avro project ook een beknopte beschrijving van de architectuur en de gebruikte technologiën en libraries worden toegevoegd.

De code moet tegen de deadline via de uploadzone van het vak Telecommunicatie Systemen op Blackboard worden opgestuurd. Zorg ervoor dat alle benodigde elementen, scripts, files en libraries aanwezig zijn, want enkel en alleen deze code wordt gebruikt op je evaluatie. Test alles op voorhand!

6.2 Deadlines

Voor dit project zijn er volgende deadlines:

- **Zondag 22 november 2015 23u59**: Tussentijdse evaluatie.
- **Zondag 3 januari 2016 23u59**: Finale inzending.

De eindevaluatie vindt plaats in de examenreeks. Jullie maken zelf een afspraak voor een evaluatie nadat de lijst met mogelijke datums is doorgegeven.