Integration Project

Kimberly Hengst *1305093*
Tim Sonderen *1465252*
Kevin Hetterscheid *1490443*
Martijn de Bijl *1470108*

15 april 2014

Voor gebruik

Onze applicatie maakt gebruik van een externe library. Om te code te laten werken, moet deze waarschijnlijk aan de projectpath worden toegevoegd. De jar is al met onze source meegeleverd, dus de instellingen van het project hoeven alleen maar angepast te worden, zodat de jar goed geimporteerd wordt.

Inhoudsopgave

1 Inleiding

Dit is ons verslag van het project van de derde module. Voor het project hebben wij een chat applicatie gemaakt. In deze applicatie kan een gebruiker in een ad-hoc netwerk met andere gebruikers berichten uitwisselen. De applicatie ondersteunt maximaal vier gebruikers. Een gebruiker kan ervoor kiezen om met n persoon, of drie personen te chatten. De berichten zijn beveiligd met behulp van encryptie. Hierdoor kunnen alleen de andere gebruikers de berichten uitlezen. Als twee gebruikers alleen willen chatten, is dit voor de andere gebruikers gencrypt. Het netwerk waarop de applicatie draait is een ad-hoc netwerk. Dit betekent dat het netwerk geen router of server nodig heeft. De computers maken een eigen netwerk via Wi-Fi.
In het verslag zullen we het proces van het ontwerpen van de applicatie, en de keuzes die we daarbij gemaakt hebben uitleggen. Als eerste bekijken we het ontwerp van de applicatie, hier zal de basis uitgelegd worden. Daarna zal de implementatie van de beveiliging uitgelegd worden. Als laatste zal het test proces beschreven worden.

2 Ontwerp van de applicatie

De gebruiker kan in een user interface berichten typen en versturen. Deze berichten worden dan in een pakket opgeslagen, en het pakket wordt verstuurd. Het versturen gebeurt via een multicast socket. Dit betekent dat het bericht naar iedereen verstuurd wordt. Als een socket een pakket ontvangt, wordt deze in de GUI weergeven.

2.1 Pakketjes

Voor het versturen van de pakketjes gebruikt multicast socket een Datagram packet. Deze gebruiken wij ook voor het versturen van data. Een datagram packet heeft een byte array met data, een IP adres van de ontvanger, en een poort van de ontvanger. In mulitcast wordt een pakketje dus naar iedereen verstuurd, het IP adres is dan ook het adres van een multicast network, een IP van de IP-klasse D. De poort is een standaard poort waar wij voor hebben gekozen. In de data van de datagram packet worden nog enkele andere velden aangewezen. De eerste byte bevat de sequencenummer, de tweede byte de teller voor het aantal hops, de derde tot zesde byte bestaan uit het IP-adres van de verzender, de zevende tot tiende byte bestaan uit het IP-adres van de ontvanger. We slaan de IP-adressen ook in de header op, omdat het IP-adres in een datagram packet veranderd volgens de context. Bij een ontvangen bericht is het IP-adres die van de verzender, bij een pakket wat verzonden wordt, is het IP-adres die van de ontvanger.

Het pakket ziet er dan als volgt uit:

Byte array data	lengte van data	IP-adres verzender of ontvanger	Poort nummer

Het byte array van de data ziet er dan als volgt uit:

Byte array data	
Header	Bericht

De header ziet er dan als volgt uit:

Header				
Sequence nummer	Hop teller	IP-adres verzender	IP-adres ontvanger	Lengte bericht

We gebruiken pakketten niet alleen voor het verzenden van berichten uit de GUI, maar ook voor een aantal andere functionaliteiten. De functionaliteiten worden dan in het bericht van de data van het packet toegevoegd. Als het bericht dan met een bepaalde string begint, wordt hiermee een functionaliteit uitgevoerd.

De volgende berichten gebruiken wij:

Bericht	Functionaliteit
$[FILE]$	Geeft aan dat het bericht een byte array van een bestand is
$[BROADCAST]$	Geeft aan dat het bericht een broadcast is, een bericht dat elke seconde verstuurd wordt om aan te geven dat een persoon er nog is
$[NAME\_IN\_USE]$	Geeft aan dat een naam al in gebruik is, en niet nog een keer gebruikt mag worden
$[PRIV\_MESSAGE]$	Geeft een privé bericht aan. Het bericht zal dan alleen aan de ontvanger worden weergeven
$[NACK]$	Geeft aan dat het bericht een NACK is
$[TOO\_LATE]$	Geeft aan dat een bericht waarvoor een NACK is gestuurd niet meer in de buffer voorkomt
$[EOF]$	Geeft het einde van een bestand aan

We hebben voor deze header van de pakket gekozen, omdat de methode om het IP-adres van een pakket op de vragen verschilt afhankelijk van de context. Door het IP-adressen in de header weten we zeker dat we de goede adressen hebben. Daarnaast gebruiken we het sequence nummer om te kijken of we alle pakketten hebben binnen gekregen. De hop teller gebruiken we om te zorgen dat pakketjes niet oneindig doorgestuurd worden. Het laatste veld gebruiken we om het bericht goed te ontcijferen.

2.2 Bestanden verzenden

Het verzenden van bestanden gaat ook via Datagram pakketjes. Echter is nu de array van bytes niet een bericht van een gebruiker, maar een deel van een bestand. Bij het versturen wordt een bestand over meerdere pakketjes verdeeld. Dit doen we omdat een bestand niet een pakket zou passen. De verzender kiest een bestand via de GUI die hij wil versturen. Deze wordt dan verzonden, en zodra alle pakketjes bij de ontvanger zijn aangekomen, kan hij deze ergens opslaan.

2.3 Pakket verlies

Doordat het network via Wi-Fi werkt, is de kans dat een pakket verloren gaat veel groter dan via de kabel. Om te zorgen dat deze pakketjes opnieuw worden verstuurd, gebruiken we sequence nummers en NACK's. Als een pakketje verloren gaat, ziet de applicatie dat pas als het volgende pakketje aankomt met een hoger sequence nummer dan verwacht. Als de applicatie een ander sequence nummer binnen krijgt dan verwacht, stuurt de applicatie een negative

aknowledgement, een NACK. Omdat er een lange tijd tussen twee berichten kan zitten, zitten er in broadcast berichten ook een sequence nummer, waardoor elke seconde gekeken wordt naar het sequence nummer en het verwachte sequence nummer.
We doen het om deze manier, omdat er alleen een bericht gestuurd wordt, als er een bericht mist. Als voor elk pakket een bericht van ontvangst zouden sturen, zou de kans op opstopping groter worden.

2.4 Privé chat

Een gebruiker kan ervoor kiezen om berichten maar naar één andere gebruiker te sturen. Een gebruiker kan hiervoor kiezen door in de GUI /w en dan de naam van de ontvanger te typen, of op een naam in de lobby te klikken waardoor een privé chat begint. Pakketten die alleen voor een bepaalde gebruiker bedoeld zijn worden alleen in zijn GUI weergeven. Andere gebruikers krijgen die pakketten wel binnen, maar ze worden niet standaard uitgelezen.

2.5 User interface

De gebruiker kan via de user interface met het programma communiceren. De interface bestaat uit een aantal elementen:

- Tekstveld
- Typveld
- Een knop voor het verzenden van tekst en bestanden
- Een knop voor privé chat
- Een knop voor het afsluiten van het applicatie

Via deze interface is het makkelijk met de applicatie communiceren. Alle functionaliteiten kunnen via de interface aangeroepen worden.

3 Beveiling

Om te zorgen dat niet iedereen de inhoudt van de pakketten kan lezen, gebruiken we encryptie.

3.1 Encryptie

Voor het encryptie gebruiken we symetrische sleutel encryptie. Dit betekent dat de clients een sleutel delen, waarmee ze hun berichten kunnen coderen en decoderen.

4 Test plan

Voor een aantal eisen voor de applicatie hebben we test cases geschreven en uitgevoerd. De volgende eisen hebben we getest:

- Verzenden berichten
- Multi-hop
- Pakket verlies opvangen
- Beveiliging
- Bestanden versturen

4.1 Test cases

4.1.1 Verzenden berichten

We verwachten van de applicatie dat elk bericht wat door een gebruiker in de GUI vestuurd wordt, ook verzonden wordt, en aankomt bij de andere gebruikers. Om dit te testen, hebben we een klasse geschreven, die een aantal berichten stuurt. Deze berichten

4.1.2 Multi-hop

We verwachten dat de applicatie berichten doorstuurt, die binnen komen.
Om dit te testen proberen we ons te verspreiden, zodat twee individuelen nodes elkaars berichten niet meer ontvangen, maar als een node er tussen in gaat staan, die de berichten doorstuurt, de andere twee elkaars berichten weer binnenkrijgt. Doordat de middelste node de berichten doorstuurt.

4.1.3 Pakket verlies opvangen

We verwachten dat de applicatie verloren pakketjes detecteerd, en ervoor zorgt dat deze weer opnieuw verzonden worden.

4.1.4 Beveiliging

We verwachten van de applicatie dat de berichten of data uit pakketten versleuteld worden.

4.1.5 Bestanden versturen

We verwachten van de applicatie dat bestanden verstuurd kunenn worden, en indien groter dan wat er in één pakket, dat het bestand verdeeld wordt over meerdere pakketten.