Advies Centrale Bank

Paul Wondel

28 mei 2018

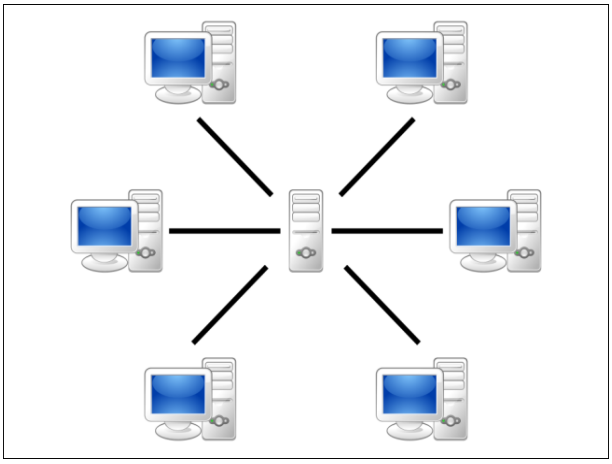
Samenvatting

In opdracht voor project 4 heb ik dit verslag opgesteld. In dit verslag worden er de kwaliteitseisen en functionaliteitseisen beschreven voor de inrichting van de Centrale Bank. De vraag die gesteld wordt is: "Hoe richt ik efficient een centrale bank op?". In dit verslag wordt er antwoord gegeven op deze vraag.

Inhoudsopgave

1 Analyse: Inrichting Centrale Bank

1.1 Server Communicatie

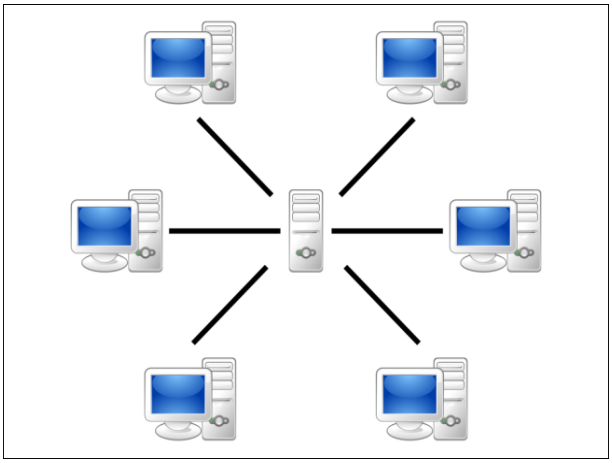


Centrale Server De Centrale Bank zal fungeren als een zogenaamd tussenpersoon. De Centrale Bank is verbonden met alle lokale banken. Als een lokale bank data nodig heeft van een andere lokale bank, kan hij de gegevens opvragen door een aanvraag naar de centrale bank te sturen. De centrale bank vervolgens stuurt de aanvraag naar de geadresseerde lokale bank. Die stuurt, op zijn beurt, de gegevens naar de centrale bank waardoor de centrale bank de gegevens terug kan sturen de lokale bank die de gegevens had aangevraagd. Het sturen van data duurt wel iets langer dan peer to peer netwerken, maar het verwerken van de data is veel efficienter bij de lokale banken. Dit vanwege het feit dat het meeste werk belast valt bij de centrale server. Bij het gebruikerseinde is dit veel fijner om te zien, want dan zijn de ATMs niet te langzaam bij het tonen van informatie.

Deze gang van zaken vindt plaats d.m.v. servers. Elke lokale bank heeft een database waarin alle gegevens van hun klanten in staan. Deze databases staan op een server van de lokale bank (dus een lokale server). De servers van de lokale banken zullen verbonden zijn aan een centrale server van de centrale bank. De databases zijn SQL-type databases. Om data uit de databases te halen moeten er query commandos verstuurd worden. Door middel van Message queuing of MQTT kunnen er een aantal queries tegelijk verstuurd worden. De connectie tussen de lokale server en de centrale server moeten wel beveiligd zijn. Hiervoor kunnen er protocollen toegepast worden, namelijk TLS (nu SSL). SSL zorgt ervoor dat de data die verstuurd wordt over deze connectie beveiligd is d.m.v. encryptie. De encryptie maakt de data onleesbaar voor externe afluisteraars. Door het gebruik van een centrale server is er geen probleem in welke

programmeer taal de clients van de lokale banken geschreven zijn. De applicatie die draait op de centrale server vertaald de quaries die de lokale banken sturen en stelt dan zijn eigen query op voor waarmee hij dan een aanvraag aan maakt. Dit wordt dan gedaan d.m.v. certificaten.

Mocht er een nieuwe lokale bank ontstaan die wenst mee te doen aan dit systeem, dan is het helemaal niet moeilijk om hem aan te sluiten aan het netwerk. Zijn programmeertaal is niet van belang omdat de server alle talen aankan. Ook is het makkelijker omdat de nieuwe lokale bank alleen verbonden hoeft te zijn aan de centrale server. Via de centrale server kan hij in contact komen met de andere lokale banken.



Peer-to-Peer Network Een peer to peer netwerk is een netwerk waarbij verschillende servers met elkaar verbonden zijn. In het figuur hierboven is een schets van het netwerk model gemaakt. In dit model zijn alle lokale banken met elkaar verbonden. Als een lokale bank data nodig heeft van een andere bank, dan stuurt hij een aanvraag rechtstreeks naar de lokale bank waarvan hij data nodig heeft. De andere lokale bank stuurt dat gewoon de data naar de lokale bank die de aanvraag verstuurd heeft. Met het peer to peer netwerk gaat het versturen van data veel sneller dan data versturen via een centrale server.

In dit model horen alle lokale banken wel in dezelfde programmeertaal geschreven te worden. Dit is om te voorkomen dat de aanvragen die verstuurd worden onleesbaar zijn. Dus het versturen van data gaat vrij snel omdat er geen vertaling plaats hoeft te vinden Het verwerken van data is ook langzamer aan het gebruikerseinde dan bij het gebruik van een centrale server. De lokale banken zijn verbonden met alle lokale banken in de regio. Dit brengt veel werk belast met zich mee. Omdat de banken een constante connectie moeten openhouden met elkaar. Hiervoor moeten er veel resources gebruikt worden om dit stand te houden. Hiernaast hoort hij constant data te verwerken om aan

de eindgebruiker te tonen. Al dit werkbelast maakt het gebruik bij de ATM niet prettig voor de klant. De klant zal langer moeten wachten om de gegevens tevoorschijnt te krijgen.

Bij deze connectie hoort er ook encryptie te zijn. De encryptie zal hier helaas per connectie moeten geschieden. Als er bijvoorbeeld 50 banken in de regio zijn, zal elke bank 49 connecties open hebben staan. Alle 49 connecties horen dan ook met ge-encrypt te zijn met b.v. SSL/TLS. Dit neemt dan ook veel resources van de bank.

1.2 Certificaten

Certificaten gebruiken bij het aanvragen van data is een goede manier om data te versturen tussen verschillende clients in verschillende programmeertalen. Een certificaat is data opsturen in een bepaald formaat om bepaald data uit het database te krijgen. De centrale server vertaald dit certificaat in SQL queries en stuurt die dan naar de databases. De databases sturen dan de data op naar de centrale bank. De centrale bank stuurt dan de data weer in een certificaat formaat zodat de lokale bank de gegevens kan lezen. Het cerficiaat kan b.v. bestaan uit een bankpasnummer en een pincode. Bij het inloggen bij een bank wordt de cerficitaat opgestuurd naar de centrale bank. Na het verwerken van de data met het database, geeft de centrale bank een response. Aan de hand van het response kan de lokale bank verder gaan met het verwerken van de data die hij ontvangen heeft.

De certificalen worden op de centrale server gemaakt en vertaald door een applicatie. Deze applicatie is in Java geschreven. Deze applicatie vertaald de certificaten in SQL queries en stuurt de data, verkregen het database, terug naar de lokale banken. De banken moet er voor zorgen dat er in hun eigensysteem het sturen en verwerken van certificaten mogelijk is.

1.3 Message Protocols

1.3.1 MQ: Message Queuing

Berichten worden door verschillende programmas gebruikt om data met elkaar te delen tussen een zender en ontvanger. Programmas gebruiken veel berichten na elkaar maar niet alle berichtn kunnen tegelijk verwerkt worden. Daarom is er een queue. Een gueue is een lijst met wachtende onderwerpen. Message Queuing zorgt ervoor dat de messages in een volgorde verstuurd worden.

Message Queuing is een asynchroon communicatie protocol. Dat houdt in dat het systeem een bericht in een queue lijst plaatst en dat hoeft niet per direct verwerkt te worden. Email is een voorbeeld van asynchroon communicatie.

Bij message queuing wordt er gebruik gemaakt van decoupling. Dit is om de afzender en ontvanger te van elkaar te scheiden. Decoupling is een proces dat delen van een systeem die afhankelijk zijn van elkaar scheidt en zelfstandig maakt.

Message queuing werkt als volgt. Een message producer(een applicatie bv) maakt een message aan met data erin. De message stuurt hij dan naar een 'Message Broker'. Een Message Broker houdt de message queue bij. Wanneer de producer al zijn messages naar de message broker gestuurd heeft stuurt de message broker de message queue naar de message consumer. De message consumer kan nu een voor een de messages in de message queue verwerken en hoeft zo niet alle messages tegelijk te behandelen. De messages hebben allemaal een topic. Aan de hand van de topic reageren de message clients op de message queue die door de message broker verstuurd wordt.

MQ brengt vele voordelen met zicht mee. MQ heeft redundantie via persistentie om data te blijven behouden. Je kan batches met messages zodat niet elke message appart wordt verstuurd, maar dat je in 1 keer een aantal messages stuurt. Dit zorgt voor de efficientie. MQ kan ook in verschillende talen worden geschreven. Berichten kunnen meerdere keren worden verstuurd. Er kunnen ook meerdere message clients zijn per topic.

MQ heeft ook een aantal nadelen. Bij MQ weet de message producer niet wie zijn clienten zijn. Dus de message broker stuurt gewoon de message queue naar alle verbonden apparaten. Een message client moet verbonden zijn om een bericht te kunnen ontvangen anders wordt hij niet naar hem toegestuurd.

Toepasselijkheid op de Centrale Bank Op dit moment stuurt de client van de lokale bank berichten na mekaar steeds naar de lokale server. Er is geen regeling of controle bij het sturen van de berichten. Op kleine schaal is dat niet van belang. Op een grotere schaal is komt dit wel in conflict met de efficientie van de server connectie tussen de centrale bank en de lokale banken. De centrale bank is verbonden met tientallen servers van de lokale banken. De lokale banken sturen ook constant berichten naar de centrale bank. De centrale bank kan dit niet in een keer verwerken. Dus om dit probleem op te lossen maken we gebruik van MQ. Message Queuing zorgt er voor dat de berichten(met queries voor de database) die per client gestuurd worden naar de centrale bank eerst in een lijst worden gedaan om in een keer gestuurd te worden naar de centrale bank. Dit zorgt ook voor minder data belast. De server kan dan gerust per queue de berichten een voor een verwerken.

1.3.2 MQTT: Message Queuing Telemetry Transport

MQ en MQTT hebben veel overeenkomsten maar zij verschillen toch van elkaar. Hier is het de bedoeling dat de berichten naar een message client verstuurd worden, i.p.v. meerdere message clients. Dit protocol is heel licht. Dus het is geen zwaar process dat veel resources neemt van de server en applicaties. MQTT wordt vaak gebruikt bij IOT (Internet of Things). Dus het is afhankeljk dat de apparaten met MQTT verbonden zijn binnen hetzelfde netwerk.

MQTT werkt op dezelfde manier als MQ. Maar i.p.v. topics maakt MQTT gebruik van channels. De message clients kunnen zich dan subscriben (inschrijven) bij zo een kanaal om de berichten te ontvangen van een bepaald apparaat.

Dus gewoonlijk een message producer, message broker en een message client. Maar bij MQTT wordt de producer de publisher genoemd, en de client wordt een subscriber genoemd. Een publisher stuurt berichten en een subscriber ontvangt hen op een kanaal. Een publish(bericht) kan je naar meerdere subscribers sturen. MQTT heeft wel de mogelijkheid om een message queue te maken maar dat hoeft niet.

In MQTT is er een mogelijkheid om QoS (Quality of Service) te hebben. Hiermee kan er aangegeven worden welke berichten belangrijk zijn, en welke weer niet. QoS bestaat uit 3 niveaus:

QoS-0 (niveau 1): Het bericht wordt een aantal keer verstuurd en niet opgeslagen. Er wordt ook niet gecontroleerd of het bericht aankomt.

QoS-1 (niveau 2): De publisher stuurt een bericht tenminste 1 keer en verwacht een response van de broker als conformatie dat de broker het bericht naar alle subscribers gestuurd heeft.

QoS-2 (niveau 3): De publisher stuurt maar 1 bericht naar de broker. De broker bevestigd aan de publisher dat het bericht is ontvangen. Daarna stuurt de publisher een bevestiging terug dat hij de bevestiging van de broker ontvangen heeft.

Helaas is MQTT is niet veilig omdat de gegevens gewoon als plain text verstuurd worden. Hiervoor is SSL/TLS geschikt om toe te passen.

Toepasselijkheid op de Centrale Bank MQTT kan gebruikt worden bij het sturen van de certificaten naar de centrale bank. Elke bank kan op een appart kanaal ge-subscribed zijn. Hierdoor is het ook te voorkomen dat er berichten verloren gaan. De kanalen zorgen voor de zekerheid dat de berichten tussen de lokale en centrale bank worden ontvangen. Als er veel lokale banken verbonden zijn aan de centrale bank, dan is het handig om gebruik te maken van MQTT. Er moet gebruikt gemaakt worden van QoS-1 om het versturen van efficient en veilig te maken.

2 Kwaliteitseisen

2.1 Security

Encryptie Als we het hebben over encryptie, dan hebben wij het gewoon over het versleutelen van data. SSL en Hashing zijn hier goede voorbeelden van. Hashing is een

TLS/SSL

Hashing

Firewall

Arduino beveiliging

2.2 Uitbreidbaarheid

2.3 Effectiviteit

2.4 Bruikbaarheid

3 Advies: Inrichting Centrale Bank

4 Project Management

4.1 Kwaliteitseisen

4.2 Code Beheer

4.3 Issue & Risico

Issue Tracking

Risico Log