Versleijen,Lowie L.W.M.

l.versleijen@student.fontys.nl

Server communicatie protocol

Onderzoek

Contents

[1. Introductie 2](#_Toc115161723)

[2. Onderzoeksvragen 3](#_Toc115161724)

[2.1. Hoofdvraag 3](#_Toc115161725)

[2.2. Deelvragen 3](#_Toc115161726)

[3. Transport layer 4](#_Toc115161727)

[3.1. Sockets 4](#_Toc115161728)

[3.2. TCP 4](#_Toc115161729)

[3.3. UDP 4](#_Toc115161730)

[4. Application layer 5](#_Toc115161731)

[4.1. Berichten 5](#_Toc115161732)

# Introductie

Tijdens de proftaak in het derde semester werkt de groep aan het project Battlebots. In dit project moeten een aantal protocollen en soorten hardware gebruikt worden. Aan het begin van het semester wordt er onderzoek gedaan door de projectgroep naar de verschillende mogelijkheden en wat het beste gebruikt kan worden in dit semester.

In dit document wordt onderzoek gedaan naar de mogelijkheden die er zijn op het vlak van de communicatie tussen de robots en de server. Het gaat om welke protocollen gebruikt worden om een bericht van A naar B te krijgen, en welke afspraken er gemaakt worden tussen de verschillende devices.

Aan het eind van het document komt een conclusie met een advies over hoe de protocollen eruit moeten zien in het project.

# Onderzoeksvragen

## Hoofdvraag

Welke protocollen kunnen het beste gebruikt worden tussen de server en de robots in het Battlebots project?

## Deelvragen

* Welk protocol kan het beste gebruikt worden op transport niveau?
* Welk protocol kan het beste gebruikt worden op application niveau?

# Transport layer

De communicatie tussen de robots en de server zal gaan over internet. Welk protocol hieroverheen gebruikt wordt, wordt hieronder uitgewerkt.

## Sockets

Bij het gebruik van socket communicatie maakt de server een socket aan op een bepaald address. De client kan connecten met de server en berichten sturen en ontvangen. De server kan ook berichten sturen naar en ontvangen van sockets die verbonden zijn met de server.

Sockets worden heel breed gebruikt in de praktijk. Dit heeft het voordeel dat er veel over te vinden is en dat veel problemen al opgelost zijn. Veelgebruikte protocollen die gebruik maken van sockets zijn TCP, UDP en HTTP. Het gebruik van sockets belast de processor minimaal, dit is handig bij het gebruik van embedded systemen waar processor tijd vaak gelimiteerd is.

Het gebruik van sockets is bij veel projecten vaak de eerste en ook de beste keuze. Dit omdat veel systemen zijn aangepast op het communiceren met sockets.

## TCP

Tcp is een protocol dat gebaseerd is op een connectie. Alle berichten die verzonden worden kunnen alleen verstuurd worden als er een connectie aanwezig is. Verder wordt er ook nog altijd gecontroleerd of het bericht ontvangen is en of het ontvangen bericht ook klopt. Doordat er zoveel gecheckt word is het wel een vrij zwaar protocol. Veel bekende protocollen maken ook gebruik van TCP als er zekerheid moet zijn dat alles aankomt.

## UDP

UDP is een protocol dat vaak naast TCP wordt gelegd. Ze maken beide gebruik van sockets om de communicatie te verzekeren. Het grote verschil tussen de 2 protocollen is dat UDP niet op een connectie gebaseerd is. Dit wil zeggen dat voor het sturen van een UDP bericht niet perse een connectie nodig is. Ook worden de berichten die verstuurd worden niet verder gecontroleerd. De enige controle die plaatsvind is een checksum. Het is een korte checksum die niet met recovery werkt. TCP doet dit wel. UDP is een stuk minder zwaar op de processor dan TCP.

## Bluetooth

Het gebruik van bluetooth is een goede optie als de afstand tussen de apparaten niet heel groot is. Een voordeel daarvan is dat de verbinding lastig te onderbreken is van buitenaf. Een nadeel is dat de verbinding makkelijk kan worden onderbroken en dat de afstand vaak niet heel erg groot kan zijn. Voor het prototype is het makkelijker om gebruikt te maken van TCP/IP omdat deze ook makkelijker getest kan worden vanuit een laptop.

# Application layer

Welke berichten worden er gestuurd en hoe vaak worden deze gestuurd. Door deze vragen te beantwoorden kan er een keuze gemaakt worden over welk protocol er gebruikt gaat worden.

## Protocol server-robot mogelijkheden

Het protocol tussen de server en de robot kan op verschillende manieren aangepakt worden. De server kan werken als wedstrijdleider/scheidsrechter, dit wil dan zeggen dat de robots verbinden met de server, vervolgens zegt de server tegen de robots dat het spel is gestart. Tijdens het spel geven de robots constant updates naar de server toe. Als er een robot is uitgeschakeld of de tijd is voorbij dan zet de server het spel stil.

Een andere optie is dat de server alleen helpt bij het opstarten met de spellen. In dit geval verbindt de robot ook eerst met de server. Maar op het moment dat het spel start, dan stuurt de server de verbindingsinformatie naar beide robots, zodat de robots onderling het spel kunnen regelen.

Deze opties hebben beide hun voor- en nadelen. Door de server als scheidsrechter te gebruiken kunnen er makkelijk meerdere spelvormen toegevoegd worden zonder dat de code van de robots hiervoor moet worden aangepast. Ook ligt er dan meer verantwoordelijkheid bij de server wat dus makkelijker bij te houden is voor de beheerder.

Door de server alleen te gebruiken bij het opstarten van het spel ligt er meer verantwoordelijkheid bij de robot zelf. Hierdoor is er een extra soort verbinding nodig en dus een extra plek waar het mis kan gaan. Een voordeel is dat de server niet heel erg belast zal gaan worden. Hierdoor is het dus mogelijk om veel meer robots tegelijk te laten spelen.

## Protocol

Voor dit project is het handiger om de server te gebruiken als scheidsrechter. Dit omdat het dan makkelijker is om verschillende spelvormen toe te voegen. Ook zal het project niet zo groot worden dat de server overbelast raakt door te veel robots die verbonden zijn. Waar de grens ligt van het aantal robots dat mogelijk kan verbinden wordt later nog getest.

De verschillende stappen die gezet worden in dit protocol zijn als volgt:

* De robots verbinden met de server.
* Op de server wordt een spel gestart en de verbonden robots kunnen worden geselecteerd.
* De geselecteerde robots krijgen een melding dat het spel is gestart, met eventueel spelregels erbij.
* De server vraagt regelmatig een ping van de robot om ervoor te zorgen dat deze nog verbonden zijn.
* De robots geven door aan de server als ze geraakt zijn.
* Als het spel is afgelopen, door tijd of een robot heeft gewonnen, dan stuurt de server dit naar alle robots.
* De server blijft verbonden met de robots nadat het spel is afgelopen. Op dit moment kan er een nieuw spel gestart worden.

# Conclusie

Door rekening te houden met de eisen van het project is het beste om TCP/IP te gebruiken. Dit heeft te maken met de het feit dat de robots draadloos moeten kunnen communiceren met de server en dat het voor het prototype belangrijk is dat het goed getest is.