## De master

De master is het brein van de webcrawler. De master communiceert met de crawler(s) en de parser(s) en slaat alle data op die binnenkomt in een database. Als de master wordt opgestart moet er een port meegeven worden. Deze port wordt gebruikt door de crawler en parser om met de master te communiceren. De master maakt vervolgens een socket aan deze gaat luisteren naar inkomende aanvragen voor verbindingen. Voor elke binnenkomende verbinding wordt een thread gestart. Deze verbinding wordt open gehouden tot de crawler of parser de verbinding sluit.

De master gaat vervolgens wachten op binnenkomende pakketjes van de crawler of parser, ook wel een thywinpacket. Een thywinpacket bevat een methode, een type en content. Er wordt gecontroleerd of de methode van het pakket een GET = 1, PUT = 2 of RESPONSE = 3 is. Vervolgens wordt er gecontroleerd welk type het pakket heeft. URI = 1, DOCUMENT = 2, DOCUMENTVECTOR = 4 en URIVECTOR = 5 zijn de pakket types. De nummertjes geven de waarde aan die uitgelezen kunnen worden. Als de master een GET URI binnenkrijgt van een cliënt wordt er een thywinpacket gemaakt met een URI. Deze URI wordt gehaald uit de URI queue en dit is het eerste element daarin. Dit element wordt daarna uit de URI queue verwijderd. Het gecreëerde thywinpacket heeft een methode RESPONSE en een type URI. De content van het pakket bevat de URI en de relevantie. Dit response pakket wordt terug gestuurd naar de cliënt. Dit gebeurt hetzelfde met een GET DOCUMENT. Het pakket dat terug gestuurd wordt heeft een methode RESPONSE en een type DOCUMENT. De content van het pakket bevat dan een URI en een DOCUMENT. De PUT methode wordt aangeroepen door de crawler en de parser, dit wordt later beschreven hoe deze pakketten gemaakt worden. Als de master een PUT methode binnen krijgt, dan weet de master dat hij iets moet opslaan. Dit wordt gedaan op basis van het pakket type. Bijvoorbeeld een PUT URI pakket slaat een URI op in de URI queue in de database. Dit gebeurt hetzelfde voor een PUT DOCUMENT. Voor een PUT URI wordt ook de relevantie van de source pagina, dus waar de link gevonden is, opgeslagen en de relevantie die gevonden wordt tijdens het parsen van een document. Ook wordt er opgeslagen hoe vaak een URI in de URI queue wordt gezet. Dit is om te voorkomen dat er dubbele URI’s in de queue komen te staan.

## Thywinlib

Een thywinpacket wordt gemaakt door functies in de thywinlib. De master, parser en crawler gebruiken de thywinlib. De thywinlib bevat ook het algoritme voor het vergelijken van document vectoren en het stemmings algoritme. Ook bevat de thywinlin een logger class.

Zoals eerder beschreven bevat een thywinpacket een methode, een type en een content. Hoe weten we wat de methode en het type is en wanneer de content begint? Hiervoor is protocol gemaakt. Dit protocol maakt gebruikt van tekens die niet uit te printen zijn, maar die wel gelezen kunnen worden door het programma. Een thywinpacket ziet er als volgt uit:

METHODE | HEADER\_SEPERATOR | TYPE | HEADER\_SEPERATOR | CONTENT | END\_OF\_PACKET

Een voorbeeld van een pakket voor het menselijke oog:  
11http://www.google.nl/

Stel dat we bijvoorbeeld meerder URI’s opsturen, hoe ziet dat er dan uit?

METHODE | HEADER\_SEPERATOR | TYPE | HEADER\_SEPERATOR | CONTENT | CONTENT\_SEPERATOR | CONTENT | CONTENT\_SEPERATOR | CONTENT | CONTENT\_SEPERATOR | END\_OF\_PACKET

Een voorbeeld van een pakket met meerdere URI’s voor het menselijke oog:

15http://www.google.nl/http://www.tweakers.nl/http://www.facebook.com/http://www.han.nl/

Hoe zorgen we ervoor dat bijvoorbeeld documentvectoren of getallen goed overkomen in een pakket en dat de master, de crawler of de parser het pakket goed kan uitlezen, maar ook goed versturen. Hiervoor gebruiken wij serialize en deserialize. Serialize en deserialize wordt gebruikt op alle content van een thywinpacket.

#### Serialize

Serialize is het plat slaan van je object en dit omzetten in bytes. Wij doen dit in de vorm van een string. De serialize functies voegen de CONTENT\_SEPERATOR, de HEADER\_SEPERATOR en de END\_OF\_PACKET toe aan bericht dat uiteindelijk verstuurd wordt. Als bijvoorbeeld serialize op een vector aangeroepen wordt, maakt de serialize functie hier een string van door steeds langs elke element te lopen en deze aan het einde van de string toe te voegen met een separator.

#### Deserialize

Deserialize doet het tegenovergestelde van serialize. Hier worden bytes omgezet naar objecten. Het omzetten naar objecten wordt gedaan op basis van een separator. Er wordt steeds gelezen tot een separator.

## De crawler

Als de crawler wordt opgestart moet er het aantal processen, het IP-adres en een port meegeven worden. Vervolgens gaat de crawler het aantal processen dat meegeven is aanmaken. Dit wordt gedaan met behulp van een fork. Fork creëert een exacte kopie van het huidige proces (parent proces) dit verder gaat in het zo genoemde child proces. Als er bijvoorbeeld tien wordt meegeven worden er tien crawler processen gestart. Elk proces gaat vervolgens crawlen.

Het eerst wat een crawler proces doet is een URI opvragen bij de master. Dit wordt gedaan door een thywinpacket te sturen naar de master. Dit is een GET URI pakket zonder content. Vervolgens wacht de crawler op een URI pakket van de master. De URI uit het pakket wordt vervolgens gebruikt om te crawlen. Tijdens het crawlen wordt er gebruikt gemaakt van CURL. CURL is een proces dat gestart wordt en de http header en de content kan downloaden.

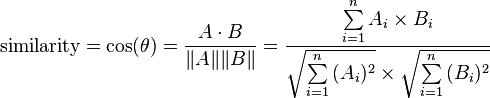
Het crawler proces verwerkt deze data vervolgens. De http header wordt uitgelezen. Hier wordt eerst uitgelezen of de http status code geen 3xx Redirection bevat. Is dit wel het geval dan wordt er gezocht naar de redirect URI. Deze redirect URI wordt terug gestuurd naar de master met een PUT URI pakket. De content van dit pakket bevat de redirect URI en de relevantie van de initiële URI die verkregen is van het GET URI pakket. Als dit niet het geval is, wordt er gecontroleerd of de http header het goede content type bevat. Er is voor gekozen om alleen “text/html” te accepteren. Als het content type hier niet gelijk aan is, wordt er niks gedaan met de URI. Als het content type wel correct is wordt er een thywinpacket met de waardes PUT DOCUMENT gemaakt. De content van dit pakket bevat de URI en het DOCUMENT. Dit pakket wordt verstuurd naar de master en daar opgeslagen in de document queue. Hierna begint het crawlen weer opnieuw, de crawler vraagt een nieuwe URI aan en verwerkt deze volgens bovenstaand proces.

## De parser

Als de parser wordt opgestart moet er het IP-adres en een port meegeven worden. Vervolgens maakt de parser een document vector van het onderwerp ("probabilistic logic discrete event simulation"). De parser stuurt een thywinpacket met GET DOCUMENT en zonder content naar de master. De master stuurt vervolgens een URI en DOCUMENT terug. Dit is hetzelfde proces als bij de crawler maar dan met een document.

Het document wordt vervolgens verwerkt in de parser. Eerst worden alle URI’s eruit gehaald. Alle URI’s worden eruit gehaald, ook relatieve URI’s en URI’s in lagere mappen. Hierna worden alle HTML tags eruit gefilterd, aangezien deze irrelevant zijn. De tekst die overblijft wordt gebruikt om een document vector te maken. Een document vector is nu al twee keer genoemd, hier volgt een kleine uitleg van de document vector. Een document vector wordt gecreeerd uit een lange string van woorden. Steeds wordt er een woord uitgehaald en deze wordt in een map(vector) gestopt. Voor het volgende woord wordt er gekeken of deze al voorkomt in de map. Is dit het geval dan wordt er een getalletje opgehoogd van dat woord. Is dit niet het geval dan wordt het woord toegevoegd aan de map. Bijvoorbeeld een string als dit: Lorem ipsum dolor sit amet, lorem lorem amet. wordt dit: <lorem, 3> <ipsum, 1> <dolor, 1> <sit, 1> <amet, 2>. Ook maakt de document vector gebruik van stemming. Dit betekend dat de stam van een woord wordt genomen. Voor stemming wordt het porteralgoritme gebruikt. Een voorbeeld: working = work, calling = call, call = call, index = index, bobby = bobbi, party = parti

Als er van het onderwerp en het document een vector zijn kunnen deze vergeleken worden. Dit wordt aan de hand gedaan van een cosine similarity algoritme, zoals hieronder beschreven.



Dit algoritme levert een getal op, dit is de relevantie van het document tegenover het onderwerp. Als de relevantie 1 is het document 100% gelijk aan het onderwerp en als de relevantie 0 is komt het onderwerp niet voor in het document.

Een document vector is eigenlijk een index van het document. De index wordt opgestuurd naar de master met een thywinpacket. Dit pakket bevat een PUT DOCUMENTVECTOR en de content bestaat uit de URI, de relevantie en de index van het document. Vervolgens worden alle gevonden URI’s naar de master gestuurd. Nu wordt er een thywinpacket gemaakt met de waarde PUT URIVECTOR. De content bevat dan een lijst met URI’s

# Searchengine

Als de searchengine opgestart wordt moeten er 2 parameters meegegeven worden. Een poort en het maximaal aantal connecties die de searchengine mag hebben. De poort wordt gebruikt door de website zodat deze met elkaar kunnen communiceren. Tijdens het opstarten van de searchengine wordt er een socket aangemaakt die gaat luisteren naar inkomende berichten van de cliënt (website). Voor elke binnenkomende connectie wordt er een thread aangemaakt. Een thread gaat wachten tot deze een string ontvangt van de cliënt. De string wordt gebruikt voor het zoeken in de database. Van de string, ook wel de zoekterm van de gebruiker, wordt een documentvector gemaakt. Voor elk element in de documentvector van de zoekterm worden alle keywords met bijbehoren URI en occurence opgehaald waar het keyword gelijk is aan het huidige element. Dit wordt opgehaald uit de indices tabel. Nu is er een lijst met URI’s met bijbehorende keyword en occurence. Eerst wordt er een documentvector gemaakt van alle keywords die bij een gevonden URI horen. De documentvector van de keywords representeert de volledige content van een URI. Deze wordt dan vergeleken met de documentvector van de zoekterm. Op basis van de hoogste relevantie wordt er steeds een URI toegevoegd aan een thywinpacket. Dit gebeurt tot er 25 URI’s in het pakket zitten. Dit pakket wordt vervolgens gestuurd naar de cliënt. De website laat vervolgens een lijst zien met de zoekresultaten. De cliënt sluit na het ontvangen van de URI’s de verbinding met de searchengine en de thread wordt beëindigd door de searchengine.