— Verslag P&O CW 2019–2020 Taak ST1.1 —

Department Computerwetenschappen – KU Leuven 25 februari 2020

Team 12

Frédéric Blondeel
Dirk Vanbeveren
Bert Van den Bosch

Inhoudsopgave

1 Introductie

Het synchroniseren van de slave devices is essentieel om video of animatie op een correcte manier weer te geven. Het eerste deel van de opgave kijkt naar het absolute verschil tussen cliënt- en referentietijd. Doordat de gebruikte server gesynchroniseerd is met een atomische wereldklok, kan het verschil in tijd worden gemeten. De testen gebruiken verscheidene besturingssystemen en browsers. Het verslag licht ook het verschil tussen UDP en TCP, twee verbindingprotocols, toe.

2 opstelling

In de opstelling is er een server gebruikt waaraan alle clienten zich aansluiten via een socket protocol. Met een knop op de pagina worden alle klokken gesynchroniseerd met de server door het tijdsverschil en de vertraging (ping) tussen de laatste twee te berekenen. Het tijdsverschil wordt namelijk 10 keer herberekend om daar uit mogelijke verbindingsfluctuaties uit te kunnen filteren (meer hierover in sectie [IN TE VULLEN]).

3 Testen

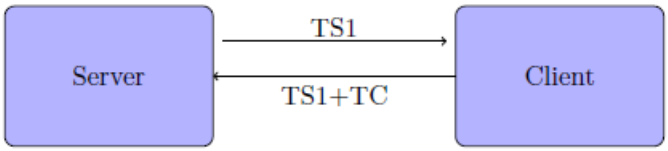
3.1 Algemeen concept

De testen maken gebruik van een simpele server waar clients op kunnen verbinden. De client kan vervolgens het absolute tijdverschil meten met een atomische wereldklok.

Eerst en vooral is er een vertraging tussen een aangesloten client en de server, de *ping*. Dit is gemeten in milliseconden. De testen meten de vertraging door een bericht met de actuele tijd te verzenden van de server naar de client, en terug. De vertraging is dan de verzonden tijd afgetrokken van de actuele tijd waarmee de ping verkregen is. In figuur 1 is de informatieoverdracht zichtbaar. De server berekent de servertijd (TS1) door middel van een API die de exacte wereldtijd teruggeeft, verzonden naar de client en teruggekregen. De uiteindelijke ping is:

$$ping = TS2 - TS1$$

met TS2 de actuele tijd berekent in de server TS2



Figuur 1: Ping

Het is niet gegarandeerd dat de klokken van de clients allemaal gesynchroniseerd zijn met de server. Bij het terug verzenden van de client naar de server wordt de clienttijd (TC) bij het bericht gezet. Met deze TC en de berekende *ping* is het mogelijk het tijdsverschil tussen de client en de server te bepalen ($DeltaTime$).

$$DeltaTime = (TC + ping/2) - TS2$$

Het verslag zal deze data bekijken vergelijken tussen verschillende browsers en besturingssystemen.

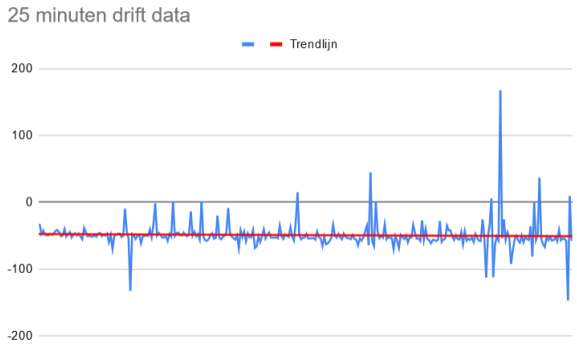
3.2 UDP vs TCP

UDP heeft in tegenstelling tot TCP geen verbinding nodig tussen bijvoorbeeld server en client. TCP zal garanderen dat data correct aankomt door middel van foutopsporing en zal ook in de goede volgorde binnenstromen. Om geen pakketten te verliezen zal TCP deze in een *receive buffer* steken en zal de applicatie de ontvangen data pas lezen als ze er klaar voor is. Tegenover UDP waar de data continu zal binnenstromen, ontvangen of niet. Deze zal ook niet aan foutopsporing doen en de juiste volgorde is niet gegarandeerd. Het is duidelijk dat UDP veel sneller is doordat deze minder stappen bevat. Dit is ook de reden dat het NTP protocol UDP zal gebruiken in plaats van TCP. Het is logisch dat voor een simpele synchronisatie tussen client en server geen complex protocol nodig is. Socket.io gebruikt het TCP protocol.

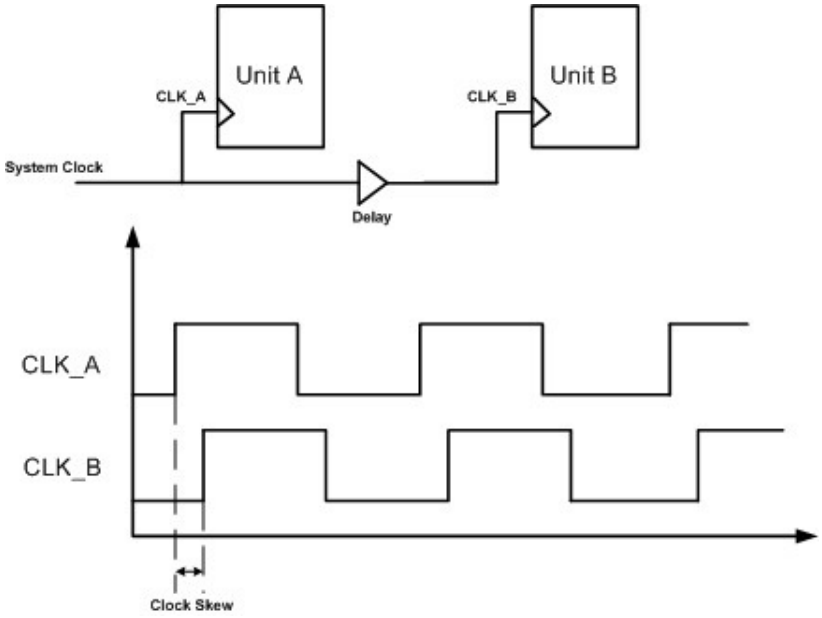
3.3 Drift en skew

Drift zal ervoor zorgen dat een klok uit sync valt met een referentie. Windows lost dit op met een wekelijkse resync (het tijdsverschil zal dus op een sawtooth diagram lijken) terwijl Mac OSX rekening zal houden met de klok skew om zo beter in sync te blijven. Klok skew is het verschil van tijd van een klok signaal tussen 2 componenten (zie figuur 3 [**?**]).

Het tijd verschil van een windows computer en een atoom clock is gedurende 25 minuten gemeten (zie figuur 2). De trendlijn is door de grafiek getrokken en het is duidelijk dat er al dan niet geen effect ervan te zien is. Over langere tijd periode zal de drift groter worden maar voor de animatie zal dit verwaarloosbaar zijn aangezien het onwaarschijnlijk is dat iemand dagenlang een deze zal laten spelen.



Figuur 2: Drift

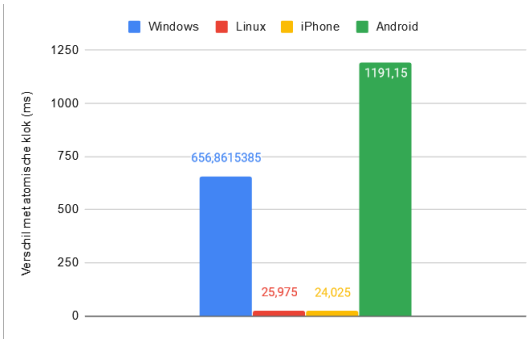


Figuur 3: Skew

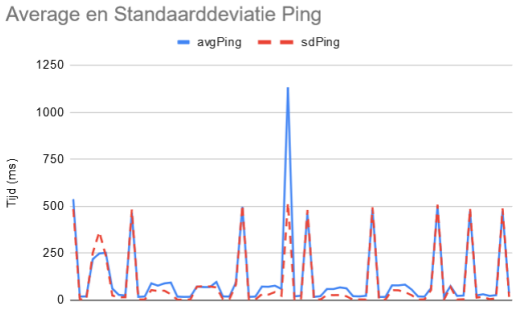
Klok skew kan problemen geven voor correct timings, en zoals eerder vermeld zal OSX dit tegen gaan via software. We zullen niet dieper gaan over de problemen van skew en OSX.

3.4 Analyse van de data

Het is duidelijk dat er op elk soort systeem een afwijking gemeten wordt tegenover de referentieklok. Bij sommige operating systems al een groter verschil dan de andere, maar de reden waarom is niet altijd te verklaren of vrijgegeven en zullen hier dus niet verder op in gaan. Elk systeem wordt door het operating system zelf voldoende gesynchroniseerd met een server-klok als referentie en zou in principe maar een paar miliseconden mogen afwijken van deze referentieklok. De fout wordt pas waargenomen in de metingen van het verschil tussen de twee klokken over een netwerk. In deze situatie wordt er een poging gedaan om het verschil tussen device-klok en server-klok te vergelijken door de ping mee in rekening te brengen. De drift van beide klokken gaan over deze tijdsspanne minimaal zijn en gaan geen invloed hebben op deze verschillen. Bij de testen werd ook de gemiddelde ping en standaarduitwijking van de ping per test opgeslagen. Uit deze waarden valt af te leiden dat er bij grote ping fluctuaties ook een grote standaarduitwijking mee gepaard ging en wijst naar inconsistenties binnen het netwerk van ofwel de client ofwel de belasting op de server en is er bijgevolg meer kans op een grotere fout binnen de vergelijking van de klokken.



(a) Mediaan van tijdsverschil van elk device.



(b) Gemiddelde en standaarduitwijking van de ping met de server.

Figuur 4: Resultaten van de testen

4 Besluit

Referenties