
Team 12

Frédéric Blondeel
Dirk Vanbeveren
Bert Van den Bosch

Inhoudsopgave

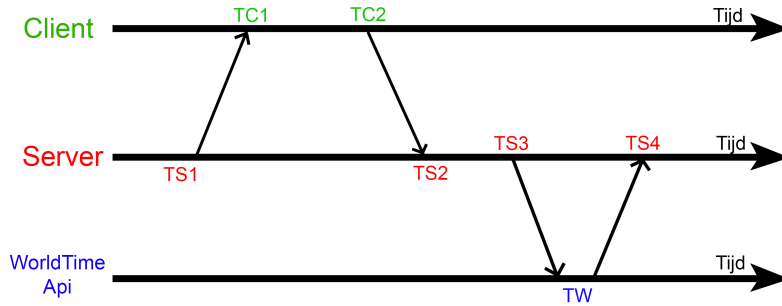
1	Introductie	2
2	opstelling	2
3	Testen	3
3.1	Algemeen concept	3
3.2	UDP vs TCP	3
3.3	Drift en skew	4
3.4	Analyse van de data	5
4	Besluit	5

1 Introductie

Het synchroniseren van de slave devices is essentieel om video of animatie op een correcte manier over meerdere schermen 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 absolute verschil in tijd worden gemeten. De testen gebruiken verscheidene besturingssystemen, apparaten en browsers. Het verslag licht ook het verschil tussen UDP en TCP, twee verbindingprotocols, toe.

2 opstelling

In de opstelling om de testen uit te voeren is er een server gebruikt waar alle clienten zich op aansluiten aan de hand van 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 daaruit mogelijke verbindingsfluctuaties uit te kunnen filteren (meer hierover in sectie 3).



Figuur 1: NTP Protocol

Voor de synchronisatie is het NTP protocol geïmplementeerd. Het schema over het NTP protocol is te vinden in figuur 1. TSx en TCx zijn de tijden gemeten door de server en respectievelijk de client. Het tijdsverschil wordt gemeten door.

$$timeDiff = \frac{(TC1 - TS1) + (TS2 - TC2)}{2}$$

De vertraging voor de server om een bericht te verzenden en terug te ontvangen is:

$$ping = (TS2 - TS1) - (TC2 - TC1)$$

Hiermee is het makkelijk de servertijd te berekenen door de vergelijking:

$$timeServer = timeClient - timeDiff$$

Om vervolgens de tijdsverschil tussen de server- en de wereltijd te berekenen is er gebruikt gemaakt van een externe API, worldtimeapi.org. Het tijdsverschil is als volgt berekend:

$$serverWorldOffset = (TW - TS4) + \frac{TS4 - TS3}{2}$$

Uiteindelijk is de volgende informatie opgeslagen:

- lijst van ping tussen client en server
- tijdsverschil tussen client en server tijd
- tijdsverschil server en wereld tijd
- tijdsverschil client en wereld tijd

3 Testen

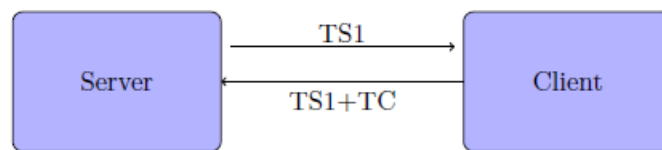
3.1 Algemeen concept

De testen maken gebruik van een simpele server waar clients op kunnen verbinden. De client kan vervolgens het absolute tijdsverschil 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 2 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 2: 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*).

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

Deze data wordt vervolgens tussen verschillende browsers en besturingssystemen vergeleken.

3.2 UDP vs TCP

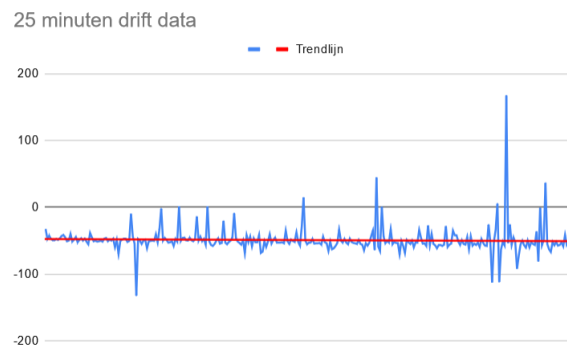
UDP heeft in tegenstelling tot TCP geen gevestigde verbinding nodig tussen bijvoorbeeld server en client. TCP zal garanderen dat data correct aankomt door middel van foutopsporing en zal ook in de goede volgorde binnenstromen. Om geen pakketten te verliezen zal TCP deze in een *receive buffer* steken 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 niet garanderen. Het is duidelijk dat UDP veel sneller is doordat deze minder stappen en controle 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 in de browser voor veiligheidsredenen.

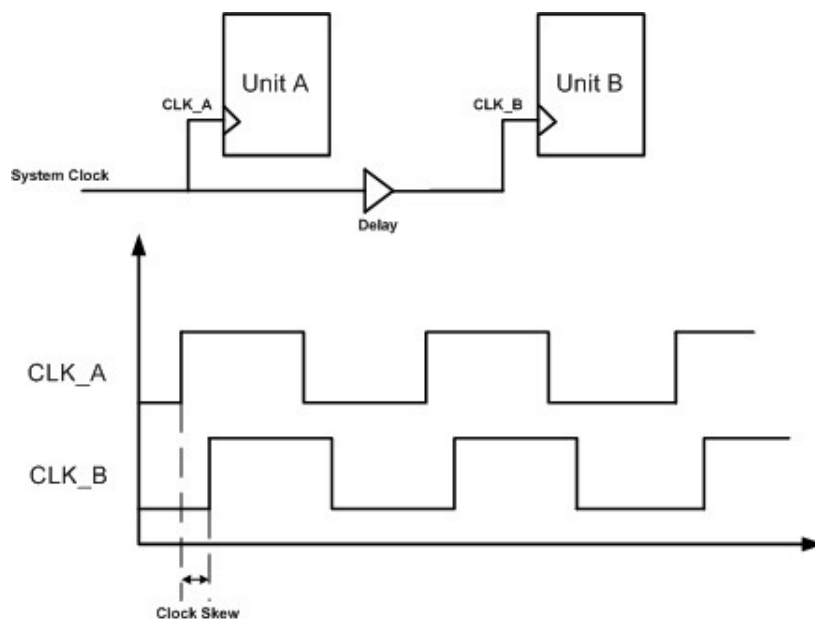
3.3 Drift en skew

Drift zal ervoor zorgen dat een klok niet meer synchroon met zijn oorspronkelijke referentie loopt. 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 en andere hardware invloeden om zo beter synchroon te blijven. Klok skew is het verschil van tijd van een kloksignaal tussen 2 componenten binnen een systeem(zie figuur 4 [?]).

Het tijdsverschil van een windows computer en een atoomklok is gedurende 25 minuten gemeten (zie figuur 3). De trendlijn is door de grafiek getrokken en het is duidelijk dat er geen effect ervan te zien is. Over langere tijdsperiode zal de drift groter worden, maar voor de animatie zal dit verwaarloosbaar zijn aangezien het onwaarschijnlijk is dat iemand dagenlang deze zal laten afspelen.



Figuur 3: Drift



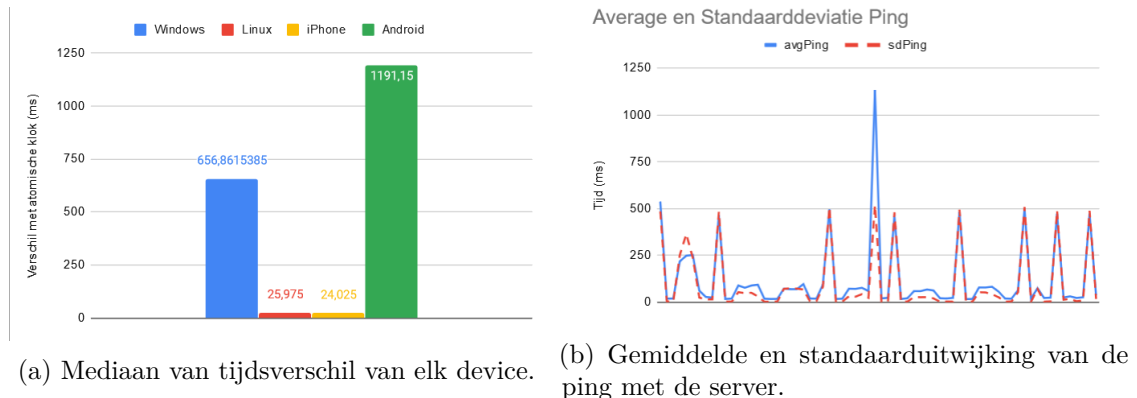
Figuur 4: Skew

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 pingfluctuaties 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.



Figuur 5: Resultaten van de testen

In figuur 5b is het duidelijk zichtbaar dat er een pieken zijn met hoge standaardafwijkingen. Dit wilt zeggen dat er heel grote fluctuaties zijn in de ping tijdens het synchronisatie proces met 10 opmetingen. Hiermee is de gesynchroniseerde tijd een afwijking hebben met de werkelijke tijd tot $(maxPing - minPing)$.

4 Besluit

Uit de testen is bewezen dat er een groot verschil kan zijn tussen de klok van de clients en de wereltijd. Bijgevolg moet er dus rekening mee gehouden worden bij de synchronisatie van apparaten. In het project van pno is dit probleem opgelost door het NTP-protocol in te voeren waarmee deze verschillen gecorrigeerd worden.

Vervolgens zijn er ook fluctuaties in het netwerk dat er voor kunnen zorgen dat de synchronisatie een slechte benadering heeft dat zelfs tot een halve seconde kan oplopen. Dit kan dan opgelost worden door 10 keer de ping te berekenen, vervolgens de standaarddeviatie

af te leiden en dan zal de client zich opnieuw moeten synchroniseren wanneer de berekende standaarddeviatie zich boven een vooraf gekozen waarde bevindt.

Uit de tests is het dan ook concludeerbaar dat de klokdrift weinig impact heeft over een korte tijdsspanne en is daardoor verwaarloosbaar.

Referenties