Project Telecommunicatiesystemen: VM Howto

Johan Bergs – Jeremy Van den Eynde

2020 - 2021

1 Gebruik van de VM

1.1 Introductie

Om het project voor het vak telecommunicatiesystemen tot een goed einde te brengen, voorzien wij een Virtualbox virtuele machine. Deze virtuele machine is het referentieplatform voor het project. Dat wil zeggen dat jouw inzending wordt getest op deze virtuele machine. Je oplossing moet hierop dus werken. We kijken **niet** of je oplossing ergens anders (bijv. je persoonlijke laptop) wel werkt!

Je kan de Virtualbox virtualisatiesoftware gratis downloaden op https://www.virtualbox.org. De virtuele machine zelf kan je downloaden op https://student.idlab.uantwerpen.be/telecom/.

1.2 Gebruikersnaam - paswoord

Wanneer je de virtuele machine in je virtualbox start, word je automatisch ingelogd als de *student* gebruiker. Het paswoord van deze gebruiker is *mvkbj1n*. De student gebruiker heeft *sudo* rechten, dus je kan, als je wilt, bijkomende software en dergelijke installeren. Hou er wel rekening mee dat de evaluatie zal doorgaan op de ongewijzigde VM!

2 Click installatie

2.1 Click instanties

Wanneer je inlogt op de virtuele machine als de student gebruiker, zal je in je home directory een aantal mappen terugvinden:

- `click`: bevat een reeds gecompileerde versie van click van https://github.com/johanbergs/click. Het is de bedoeling dat je hier, in de map `elements/local/igmp/` je eigen elementen toevoegt.
- `opgaven`: bevat alle opgaven. Voor installatie van de IGMP scripts voer je het volgende uit:

```
rm -rf $HOME/click/scripts/
cp -r $HOME/opgaven/igmp/scripts $HOME/click/scripts
```

- `referentie-oplossingen`: bevat alle referentie oplossingen. Voor installatie van de IGMP oplossing, voer je het volgende uit:

```
rm -rf $HOME/click-reference
cp -r $HOME/referentie-oplossingen/igmp $HOME/click-reference
```

De *click-reference* map bevat nu de referentie-implementatie voor het project. Deze referentie-implementatie is een oplossing in click, geïmplementeerd volgens de geannoteerde RFC. Verder is deze implementatie uitgebreid met een aantal handlers om zo "foutief" gedrag uit te lokken om je eigen implementatie tegen te testen.

2.2 Setup

Om gemakkelijk componenten uitwisselbaar te maken tussen de referentie en je eigen implementatie, is er voor gekozen om elke "entiteit" in het netwerk in een aparte click executable te laten draaien. Dit wil zeggen dat de server, de router en de vier client hosts elk in hun eigen click instantie draaien. Bijkomend is er een laatste click instantie voorzien die deze zes "aan elkaar hangt".

Om deze hele setup te laten werken, wordt er gebruik gemaakt van `tap` devices: virtuele netwerkinterfaces. Deze moeten, telkens de VM is herstart éénmalig worden aangemaakt. Dit doe je door in `click/scripts/` het scriptje `setup.sh` uit te voeren. Dit moet als root gebeuren, dus:

```
sudo /home/student/click/scripts/setup.sh
```

Dit creëert de nodigde tap interfaces `tap0..6`. Je kan altijd m.b.v. het `ifconfig` commando controleren of deze interfaces correct zijn aangemaakt. **Eens deze tap interfaces zijn aangemaakt, blijven ze bestaan zolang de VM niet wordt herstart! Je moet het setup.sh scriptje dus énkel na een herstart opnieuw gebruiken!**

2.3 Click starten

2.3.1 De referentie-implementatie

Om de (volledige) referentie-implementatie te starten, volstaat het om het volgende uit te voeren in de map `/home/student/click-reference/solution`:

```
sudo ./start_click.sh
```

De server zal vanaf dat moment UDP pakketten beginnen sturen naar het multicast adres 225.1.1.1. Aangezien initieel geen van de clients geïnteresseerd zijn in deze pakketten, komen deze niet aan bij de clients.

2.3.2 Je eigen implementatie

Om (volledig) je eigen implementatie te starten, doe je iets gelijkaardigs aan hierboven: je gaat naar de map `/home/student/click/scripts` en je voert daar `./start_click.sh` uit.

2.3.3 Stukjes van beide implementaties

Om een combinatie van beide implementaties te draaien, bijvoorbeeld al jouw eigen elementen, maar de referentie implementatie van de router, volstaat het om in jouw startup scriptje de regel voor router in commentaar te zetten en in

het referentie startup scriptje het omgekeerde te doen, dus alles in commentaar behalve de regel voor router. **Let er vooral op dat elk element maar één keer wordt gestart!**

3 Handlers

3.1 Write handlers voor de eigen implementatie

Om het voor ons makkelijker te maken om de implementatie te testen, moeten de clients (zoals in de referentie implementatie) minstens volgende write handlers voorzien op een object `igmp`:

- `join GROUP` *`multicast-group`*
- `leave GROUP` *`multicast-group`*

Dus om bijvoorbeeld client22 te laten joinen op een multicast group met adres 225.1.1.1, moet volgende call gebruikt kunnen worden:
`write client22/igmp.join 225.1.1.1`

3.2 Speciale handlers

De router en client referentie-implementaties hebben een aantal handlers om het gedrag ervan te veranderen om zo bijvoorbeeld timers, error handling e.d. uit te testen. Hieronder vind je al deze handlers opgelijst.

- **robustness** ROBUSTNESS v: wijzigt `Robustness` naar v.
- **query_interval** QUERY_INTERVAL v: wijzigt `Query Interval` naar v.
- **max_response_time** MAX_RESPONSE_TIME v: wijzigt `Max Response Time` naar v deciseconden.
- **unsolicited_report_interval** UNSOLICITED_REPORT_INTERVAL v: wijzigt `Unsolicited Report Interval` naar v seconden.
- **last_member_query_interval** LAST_MEMBER_QUERY_INTERVAL v: wijzigt `Last Member Query Interval` naar v deciseconden.
- **last_member_query_count** LAST_MEMBER_QUERY_COUNT v: wijzigt `Last Member Query Count` naar v.
- **set_igmp_type** QUERY q, REPORT r: als $q \neq 0$, dan wordt het IGMP type van uitgaande Queries veranderd naar q. Als $r \neq 0$, dan wordt het IGMP type van uitgaande Reports veranderd naar r.
- **set_igmp_address** QUERY q, REPORT r: als $q \neq 0.0.0.0$, dan wordt het IGMP multicast address van uitgaande Queries veranderd naar q. Als $r \neq 0.0.0.0$, dan wordt het IGMP multicast address van uitgaande Reports veranderd naar r.
- **set_igmp_group_record_type** TYPE t: als $t \neq 0$, dan wordt van uitgaande IGMP Reports het type van elk Group Record veranderd naar t.

- **invalid_igmp_checksum** QUERY q, REPORT r: als q=true, dan wordt de IGMP checksum van uitgaande Queries verkeerd gezet. Als r=true, dan wordt de IGMP checksum van uitgaande Reports verkeerd gezet.

Hieronder volgen enkele voorbeelden:

- **write router/igmp.set_igmp_type QUERY 67** zorgt ervoor dat de router ongeldige IGMP Queries uitstuurt.
- **write client22/igmp.invalid_igmp_checksum REPORT true** zorgt ervoor client22 IGMP Reports uitstuurt met een foutieve checksum.