Project Telecommunicatiesystemen: VM Howto

Johan Bergs – Jeremy Van den Eynde

2017 - 2018

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 twee click instanties terugvinden: *click* en *click-reference*. De eerste bevat een reeds gecompileerde versie van click. Deze versie is degene die je van de github repository kan downloaden. Het is de bedoeling dat je hier, in de map `elements/local/` je eigen elementen toevoegt, bij voorkeur gebundeld in een mapje (bijv. `elements/local/mobileip`).

De *click-reference* map bevat 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 HA, FA, MN en CN elk in hun eigen click instantie draaien. Bijkomend is er een vijfde click instantie voorzien die deze vier "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 ./setup.sh
```

Dit creëert de nodigde tap interfaces `tap0..5`. 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 `click-reference/solution`:

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

Je zal zien dat er een ping begint te lopen van de CN naar de MN. De MN is initieel in zijn thuisnetwerk. Om de MN te verhuizen naar het foreign netwerk, is een scriptje voorzien `to_foreign.sh`. Met het scriptje `to_home.sh` verhuis je de MN terug naar zijn thuisnetwerk.

2.3.2 Je eigen implementatie

Om (volledig) je eigen implementatie te starten, doe je iets gelijkaardigs aan hierboven: je voert het startup scriptje nu uit vanuit de map `click/scripts`. Ook hier zijn weer de scriptjes voorzien om de MN van netwerk te verhuizen.

2.3.3 Stukjes van beide implementaties

Om een combinatie van beide implementaties te draaien, bijvoorbeeld al jouw eigen elementen, behalve de FA, volstaat het om in jouw startup scriptje de regel voor de FA in commentaar te zetten en in het referentie startup scriptje het omgekeerde te doen, dus alles in commentaar behalve de regel voor de FA. **Let er vooral op dat elk element maar één keer wordt gestart!**

3 Handlers

De MN, HA en FA 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, per instantie, de handlers opgelijst.

3.1 Mobile Node

3.1.1 mobile_node/mobile_node

- **force_lifetime** (boolean): in de RFC staat vermeld dat de lifetime die in een registration request wordt gevraagd, niet groter mag zijn dan een bepaalde waarde (welke, dat staat in de RFC...). Met deze handler kan je er voor zorgen dat de mobile node in zijn requests toch steeds de lifetime vraagt die je hebt meegegeven in zijn configuratie.

3.1.2 mobile_node/AgentSolicitationSender@17

- **code** (integer): zet het "code" veld van de agent solicitation op de gevraagde waarde (standaard: 0)
- **interval** (integer): zet het interval, in seconden, tussen agent solicitation berichten (standaard: 5)
- **invalid_checksum** (boolean): indien `true`, zal de checksum van het agent solicitation bericht foutief worden gezet (standaard: `false`).
- **length** (integer): zet de lengte van het agent solicitation pakket. Standaard is de lengte net goed.
- **sending** (boolean): indien `false`, stuurt de mobile node geen agent solicitation berichten uit (standaard: `true`).

3.2 Home Agent

3.2.1 home_agent/adv

- **code** (integer): zet het "code" veld van de agent advertisement op de gevraagde waarde (standaard: 0)
- **invalid_checksum** (boolean): indien `true`, zal de checksum van het agent advertisement bericht foutief worden gezet (standaard: `false`).
- **sending** (boolean): indien `false`, stuurt de mobile node geen agent advertisement berichten uit (standaard: `true`).

3.2.2 home_agent/mobility

- **invalid_checksum** (boolean): indien `true`, zal de checksum van het agent advertisement bericht foutief worden gezet (standaard: `false`).
- **invalid_id** (boolean): indien `true`, zal het ID veld van het registration reply bericht foutief worden gezet (standaard: `false`).
- **invalid_node** (boolean): indien `true`, zal het home address van het registration reply bericht foutief worden gezet (standaard: `false`).
- **invalid_port** (boolean): indien `true`, zal het registration reply bericht naar de foutieve UDP poort worden gestuurd (standaard: `false`).
- **sending** (boolean): indien `false`, stuurt de mobile node geen registration reply berichten uit (standaard: `true`).

- **zero_checksum** (boolean): indien `true`, zal de UDP checksum van het registration reply bericht op '0' worden gezet (standaard: `false`).

3.3 Foreign Agent

3.3.1 foreign_agent/adv

- **code** (integer): zet het "code" veld van de agent advertisement op de gevraagde waarde (standaard: 0)
- **invalid_checksum** (boolean): indien `true`, zal de checksum van het agent advertisement bericht foutief worden gezet (standaard: `false`).
- **sending** (boolean): indien `false`, stuurt de mobile node geen agent advertisement berichten uit (standaard: `true`).

3.3.2 foreign_agent/visitors

- **invalid_checksum** (boolean): indien `true`, zal de checksum van het agent advertisement bericht foutief worden gezet (standaard: `false`).
- **invalid_id** (boolean): indien `true`, zal het ID veld van het registration reply bericht foutief worden gezet (standaard: `false`).
- **invalid_port** (boolean): indien `true`, zal het registration reply bericht naar de foutieve UDP poort worden gestuurd (standaard: `false`).
- **sending** (boolean): indien `false`, stuurt de mobile node geen registration reply berichten uit (standaard: `true`).
- **zero_checksum** (boolean): indien `true`, zal de UDP checksum van het registration reply bericht op '0' worden gezet (standaard: `false`).