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/rsvp/` je eigen elementen toevoegt.
- `opgaven`: bevat alle opgaven. Voor installatie van de RSVP scripts voer je het volgende uit:

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

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

```
rm -rf $HOME/click-reference
cp -r $HOME/referentie-oplossingen/rsvp $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 sender host, twee routers en receiver host 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..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 `click-reference/solution`:

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

Er zullen vanaf dat moment drie verschillende flows data beginnen sturen van host1 naar host2. Initieel wordt er nog geen Quality of Service toegepast.

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`.

2.3.3 Stukjes van beide implementaties

Om een combinatie van beide implementaties te draaien, bijvoorbeeld al jouw eigen elementen, behalve router2, volstaat het om in jouw startup scriptje de regel voor router2 in commentaar te zetten en in het referentie startup scriptje het omgekeerde te doen, dus alles in commentaar behalve de regel voor router2. **Let er vooral op dat elk element maar één keer wordt gestart!**

3 Handlers

3.1 Write handlers voor de eigen implementatie

In de RFC is er een RSVP Application interface voorzien. Een versimpelde versie van deze interface is vereist in de eigen implementatie, en zal gebruikt worden voor de evaluatie. De volgende write handlers moeten in de implementatie voorzien worden:

- session ID *sessionId*, DST *address*, PORT *port*
- sender ID *sessionId*, SRC *address*, PORT *port*
- reserve ID *sessionId*, CONF *confirmation*
- release ID *sessionId*

3.1.1 Typisch scenario

Het scriptje hieronder toont een typisch scenario.

```
#!/bin/sh

HOST1=192.168.1.1
HOST2=192.168.2.1
echo "write host1/rsvpHost.session ID 1, DST $HOST1, PORT 2222" \
        | telnet localhost 10001
echo "write host2/rsvpHost.session ID 1, DST $HOST1, PORT 2222" \
        | telnet localhost 10004
echo "write host1/rsvpHost.sender  ID 1, SRC $HOST2, PORT 7" \
        | telnet localhost 10001

sleep 10; echo "write host2/rsvpHost.reserve ID 1, CONF 1" \
        | telnet localhost 10004
sleep 10; echo "write host1/rsvpHost.release ID 1" \
        | telnet localhost 10001
```

3.2 Speciale handlers

De host en router 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. Alle handlers verwachten als eerste de parameter TYPES. Dit is een lijst van RSVP bericht types waarop de handler zal toegepast worden.

- **refresh_period** *milliseconds*: verandert de refresh period van de node naar *milliseconds* ms. (standaard: 10000).
- **block_messages** TYPES *types*, BLOCK *block*: indien *block*=`true`, dan worden binnenkomende RSVP berichten niet verwerkt, en uitgaande RSVP berichten niet verstuurd. (standaard: `false`)
- **return_err** TYPES *types*, CODE *code*, VALUE *value*: deze handler geldt enkel voor Path en Resv berichten. Als een node zo een bericht ontvangt, wordt het bericht niet verwerkt en een *PathErr* respectievelijk *ResvErr* bericht teruggestuurd vanuit de node met waardes *code* en *value*. Een *code*=`0` herstelt het gewone gedrag. (standaard: *code*=`0`, *value*=`0`)

- **invalid_rsvp_checksum** TYPES *types*, INVALID *invalid*: indien *invalid*=`true`, zal de RSVP checksum van buitengaande pakketten verkeerd gezet worden. (standaard: `false`)
- **invalid_message_type** TYPES *types*, INVALID *invalid*: indien *invalid*=`true`, zal het bericht type van buitengaande RSVP pakketten naar een ongeldig type veranderd worden. (standaard: `false`)
- **invalid_message_length** TYPES *types*, INVALID *invalid*: indien *invalid*=`true`, zal het lengte veld van het RSVP bericht verminderd worden. (standaard: `false`)
- **invalid_object_class_num** TYPES *types*, INVALID *invalid*: indien *invalid*=`true`, zal een willekeurig object uit het RSVP bericht een ongeldig class_num toegekend krijgen. (standaard: `false`)
- **invalid_hop_object** TYPES *types*, INVALID *invalid*: indien *invalid*=`true`, zal bij RSVP berichten die een Hop object hebben, een ongeldig adres ingesteld worden. (standaard: `false`)
- **invalid_session_object** TYPES *types*, INVALID_ADDRESS *address*, INVALID_PORT *port*: indien *address* of *port* op true staan, zal het adres en/of de poort van het Session object op een ongeldige waarde ingesteld worden. (standaard: *address*=`0.0.0.0`, *port*=`0`)

Hieronder volgen enkele voorbeelden:

- **write router1/rsvpRouter.block_messages TYPES "PATH RESV", BLOCK true** blokkeert alle RSVP Path en Resv berichten in router1.
- **write router2/rsvpRouter.invalid_invalid_object_class_num_object TYPES "*", INVALID true** zorgt ervoor dat in router2 alle RSVP berichten het Hop object veranderd wordt, als het aanwezig is.
- **write host1/rsvpHost.return_err TYPES "RESV", CODE 6, VALUE 0** stuurt een ResvErr bericht naar host2 als het een Resv bericht ontvangt. De reservatie is dan ook niet geldig.