Practicum Computernetwerken (TI2400) – opgave 1

May 21, 2013

OPGAVE 1

Doel:
Inzicht verkrijgen in het gedrag van de verschillende protocollen en mechanismen.

Systeem:
De opdracht kan worden uitgevoerd op een PC voorzien van het linux operating systeem. Er wordt gebruik gemaakt van een netwerk simulator en voorgeprogrammeerde protocollen zoals beschreven in het boek Computer Networks (4th edition) van A.S. Tanenbaum. De C-code voor de simulator en de protocollen zijn verkregen via de Website horende bij het boek. Ten behoeve van het practicum zijn er enkele wijzigingen aangebracht in deze code. De gewijzigde source code is te downloaden van blackboard.

Compileren:
de simulator wordt gecompileerd door
`gcc -m32 -c simulator.c -o simulator.o`
Een protocol (b.v. protocol4) wordt wordt gecompileerd en meteen gelinkt aan de simulator door:
`gcc -m32 -o protocol4 p4.c simulator.o`

Simulatie:
Een simulatierun (van b.v. protocol4) wordt gestart door:
./protocol4 *ticks timeout pct_loss pct_chksum debug_flag*
Hierbij is:

ticks:	Het aantal kloktikken dat de simulatie duurt
timeout:	De waarde waarop de timer wordt gezet bij het verzenden van een bericht
pct_loss:	percentage van de frames dat onderweg verloren gaat
pct_chksum:	percentage van de aangekomen frames dat een checksum error heeft
debug_flags:	geeft aan welke debugging informatie wordt afgedrukt.

Voor meer details zie de README file bij de simulator

De uitvoer:
Tijdens een simulatierun zijn er 3 processen actief. De twee communicerende processen p0 en p1 die het protocol uitvoeren en een proces pM dat voor de aansturing zorgt. Elk van deze processen heeft een eigen logfile, aangeduid met resp log0, log1 en logM. Daarnaast schrijven alle processen debugging informatie en ook statistiek naar dezelfde standaard output, hetgeen wel eens leidt tot verlies van enige data.

De logging informatie:
p0 en p1 schrijven allerlei logging informatie naar log0 en log1. Voor het reconstrueren van hetgeen zich heeft afgespeeld, is het nuttig hieruit een selectie te maken en de gegevens uit beide files te combineren in één overzicht. Iedere regel in de logfile begint daarom met een kenmerkende character combinatie waaruit de herkomst en de aard van de data in die regel blijkt. Voor het nagaan van welke frames in welke volgorde werden uitgewisseld is het nodig alle regels die beginnen met XX uit beide files te selecteren en deze in de juiste volgorde te mengen. Voor het selecteren gebruiken we het linux commando grep (tweemaal), gevolgd door sort voor het in de juiste volgorde mengen.

Voorbeeld:

```
grep ^XX log0 > log0.xx
grep ^XX log1 > log1.xx
sort -n -k2,3 log0.xx log1.xx > log2.xx
```

Het resultaat in de file log2.xx, zal er, na enkele identificerende regels, ongeveer als volgt uitzien. (de nummering boven de kolommen ontbreekt in de file)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
XXX	0	S	0	0	Data	0	1	–>					
XXX	10	R	1						–>	0	Data	0	1
XXX	10	S	1						<–	0	Data	0	0
XXX	80	R	0	0	Data	0	0	<–					
XXX	80	S	0	1	Data	1	0	–>					
XXX	140	R	1						–>	1	Data	1	0
XXX	140	S	1						<–	1	Data	1	1

De betekenis van de kolommen is als volgt.

1	soort en oorsprong van het log record (als gevolg van de bij grep gebruikte ^XX beginnen deze allemaal met XX)
2	tijd (in tikken maal 10)
3	Sent of Received
4	log record afkomstig uit p0 of uit p1
5,11	inhoud van het frame (de simulator gebruikt hiervoor de reeks 0, 1, 2, 3, etc.)
6,12	soort frame (data, ack of nak)
7,13	het sequence nummer
8,14	het ack nummer
9,10	de richting van het frame. Door bij elkaar horende pijltjes door een lijn te verbinden krijg je een goed inzicht over wat er precies gebeurt

opdracht 1a

1. copieer simulator.c, simulator.h, protocol.h, p3.c, README naar je eigen systeem

2. lees de README file

3. compileer de simulator en compileer protocol3 (source: p3.c)

4. doe een simulator run met protocol3 d.m.v.
   ./protocol3 100 10 0 0 15
   waarschijnlijk zijn er timeouts opgetreden. Als dat niet het geval is, herhaal dan de run desnoods met een kleinere waarde voor de timeout, totdat er wel één of meer timeouts optreden.

5. Analyseer het verkeer door het creëren van een file als log3.xx, zoals beschreven in de paragraaf over de uitvoer. Druk deze file af. Ga na of de timeouts wel op het juiste moment optreden. Geef in de uitvoer aan welke timeouts correct zijn en welke niet.

6. Corrigeer p3.c zodanig dat de timeouts wel correct zijn.

In te leveren:
1a.1 de afgedrukte log3.xx met daarin aangegeven de juiste en onjuiste timeouts
1a.2 de aangepaste versie van p3.c met daarin duidelijk aangegeven de wijziging(en).

opdracht 1b

1. copieer nu ook p4.c naar je eigen systeem

2. compileer protocol4 (source: p4.c)

3. Voer de volgende run met protocol4
   ./protocol4 100 20 0 0 15
   maak een file als log2.xx zoals aangegeven in de paragraaf over de output, en verklaar aan de hand hiervan de uitkomst van de efficiency

4. Pas p4.c zodanig aan dat de efficiency van protocol4 in de buurt van de 100controleer dit d.m.v. een simulatie run.

5. Voer elk van de volgende runs 3 maal uit. (het verschil tussen de runs is dus een factor 10 in het aantal tikken)

   ./protocol4 100 7 0 0 15
   ./protocol4 1000 7 0 0 15
   ./protocol4 10000 7 0 0 15

   bepaal de gemiddelde efficiency bij 100 tikken, bij 1000 tikken en bij 10000 tikken

6. Verklaar de oorzaak

In te leveren:
1b.1 De afgedrukte log3.xx, met daarin aangegeven waarom de efficiency geen 100
1b.2 een afdruk van de verbeterde p4.c code
1b.3 het antwoord op vraag 5 op een aparte bladzijde

opdracht 1c

1. Copieer nu ook p5.c en p6.c naar je eigen systeem en compileer protocol5 en protocol6.

2. Vergelijk de performance van beide protocollen door runs uit te voeren met de volgende parameters.

   ticks = 10000
   timeout = 40
   pct_loss achtereenvolgens: 5, 10, 20,30, 40,50, 60, 70, 80, 90
   pct_chksum = 0

   Registreer en bewaar voor beide processen steeds de volgende waarden:
   Total frames sent
   Data Frames lost
   Frames retransmitted
   Good data frames received
   Payloads accepted
   Timeouts
   Efficiency

3. Teken (of plot) een grafiekje waarin voor beide protocollen de efficiency (verticaal) is uitgezet tegen pct-loss (horizontaal)
   Verklaar het verschil tussen de grafiek van protocol5 en die van protocol6. Licht je antwoord toe aan de hand van de geregistreerde waarden, b.v. van Data Frames lost, Frames retransmitted, Timeouts.

In te leveren:
1c.1 De gemeten waarden
1c.2 De grafiek
1c.3 De gevraagde verklaring