

Objectif de test3:

Déterminer si pour un fichier de taille plus grande (dans une des deux dimension), notre protocole peut aussi fonctionner. La taille de tableau doit changer correspondamment.

Déterminer si la crash de la carte vient de l'asynchronisation des paquets entre la carte et PC.

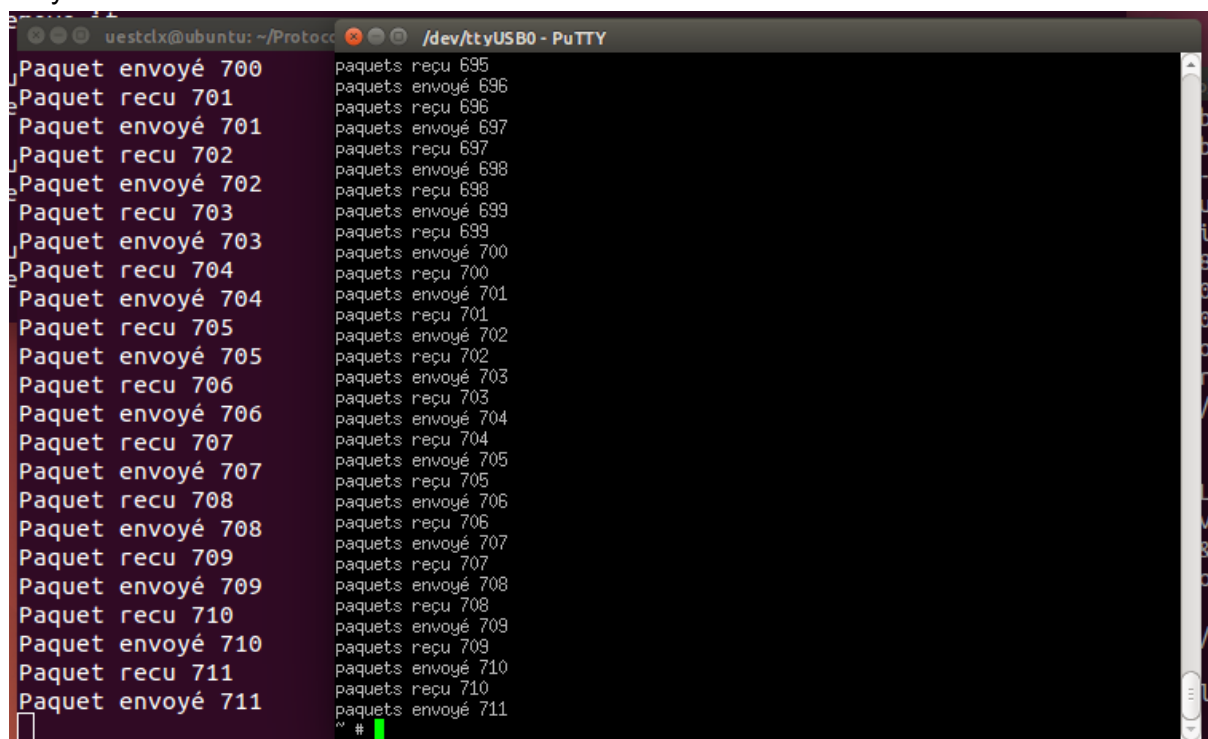
Essayer de faire fonctionner notre protocole en utilisant un tableau de taille moins grande, et utiliser le d'une manière récursive.

Pour vérifier si le problème vient de l'asynchronisation de paquet, j'ai modifié la valeur d'entervalle entre deux envois du côté client. La modification est basée sur la deuxième version de test où le serveur ne fait pas le débrutage. J'ai mis usleep(1000000) pour que le serveur puisse avoir assez de temps de recevoir et de renvoyer le paquet.

Mais malheureusement, la carte crush ou se termine juste après la réception d'un sept centaine paquet.

Pour être sûr, j'ai essayé avec plusieurs valeurs d'usleep, mais le résultat ne change pas beaucoup: soit la carte crush, soit le programme se termine plus tôt. J'ai aussi essayé l'autre manière de compilation, mais le résultat reste pareil.

Donc, maintenant on peut dire que c'est peu probable que le problème vient de l'asynchronisation.



```
uestclx@ubuntu: ~/Protoc... /dev/ttyUSB0 - PuTTY
Paquet envoyé 700    paquets reçu 695
Paquet reçu 701      paquets envoyé 696
Paquet envoyé 701    paquets reçu 696
Paquet reçu 702      paquets envoyé 697
Paquet envoyé 702    paquets reçu 697
Paquet reçu 703      paquets envoyé 698
Paquet envoyé 703    paquets reçu 698
Paquet reçu 704      paquets envoyé 699
Paquet envoyé 704    paquets reçu 699
Paquet reçu 705      paquets envoyé 700
Paquet envoyé 705    paquets reçu 700
Paquet reçu 706      paquets envoyé 701
Paquet envoyé 706    paquets reçu 701
Paquet reçu 707      paquets envoyé 702
Paquet envoyé 707    paquets reçu 702
Paquet reçu 708      paquets envoyé 703
Paquet envoyé 708    paquets reçu 703
Paquet reçu 709      paquets envoyé 704
Paquet envoyé 709    paquets reçu 704
Paquet reçu 710      paquets envoyé 705
Paquet envoyé 710    paquets reçu 705
Paquet reçu 711      paquets envoyé 706
Paquet envoyé 711    paquets reçu 706
~ #
```

```
uestcdx@ubuntu: ~/Pro /dev/ttyUSB0 - PuTTY
Paquet envoyé 694 paquets envoyé 699
Paquet reçu 695 paquets reçu 699
Paquet envoyé 695 paquets envoyé 700
Paquet reçu 696 paquets reçu 700
Paquet envoyé 696 paquets envoyé 701
Paquet reçu 697 paquets reçu 701
Paquet envoyé 697 paquets envoyé 702
Paquet reçu 698 paquets reçu 702
Paquet envoyé 698 paquets envoyé 703
Paquet reçu 699 paquets reçu 703
Paquet envoyé 699 paquets envoyé 704
Paquet reçu 700 paquets reçu 704
Paquet envoyé 700 paquets envoyé 705
Paquet reçu 701
Paquet envoyé 701
Paquet reçu 702
Paquet envoyé 702
Paquet reçu 703
Paquet envoyé 703
Paquet reçu 704
Paquet envoyé 704
Paquet reçu 705
Paquet envoyé 705

KERNEL: fault at 0x0a373538 [pc=0x0a373538, sp=0xa033ddf0]
Undefined Instruction

Pid: 29, comm: client_opt_1s
CPU: 0 Not tainted (2.6.33-arm1 #5)
pc : [<0a373538>] lr : [<101dd227>] psr: 4100000b
sp : a033ddf0 ip : 00000010 fp : a00cbe40
Code dump at pc [0a373538]:
e7bfefef f8bfded7 ffe2dfbf fd6bfeee
r10: a00cbe40 r9 : a033c000 r8 : a3419ba0
r7 : 000000a0 r6 : 00000f00 r5 : a033de08 r4 : 0a373537
r3 : 00000f00 r2 : a033df54 r1 : a3419ba0 r0 : a033de08
Flags: nZcv IRQs on FIQs on Mode UK11_26 ISA unknown Segment kernel
Kernel panic - not syncing:
Rebooting in 10 seconds..

U-Boot 2010.03-linux-cortexm-1.12.0 (Nov 25 2013 - 15:25:13)
```

Maintenant on va voir si le problème vient de la taille de tableau statique qu'on utilise dans notre client.

J'ai copié trois fois les valeurs dans le fichier car.dat, maintenant le fichier car.dat a 2197620 valeurs et une taille de 16.6MB. Le programme client.c possède 3 threads, un threads s'occupe l'envoi de données et l'autre deux s'occupent la réception et l'écriture de donnés. Le threads d'envoi va lire paquet par paquet des données depuis le fichier car.dat et l'envoyer ver le serveur; le threads de réception va recevoir les données qui viennent du serveur et stocker dans un buffer de taille 1000*960 d'une manière récursive, c'est à dire que le 1001eme paquet va être stocké dans la premier ligne; le threads de l'écriture va lire les données dans ce buffer et les écrire dans un fichier sur la carte.

Le résultat est étonnant considérant le fait que la carte envoie et recevoir plus de 5800 paquets avant qu'elle s'appuie.

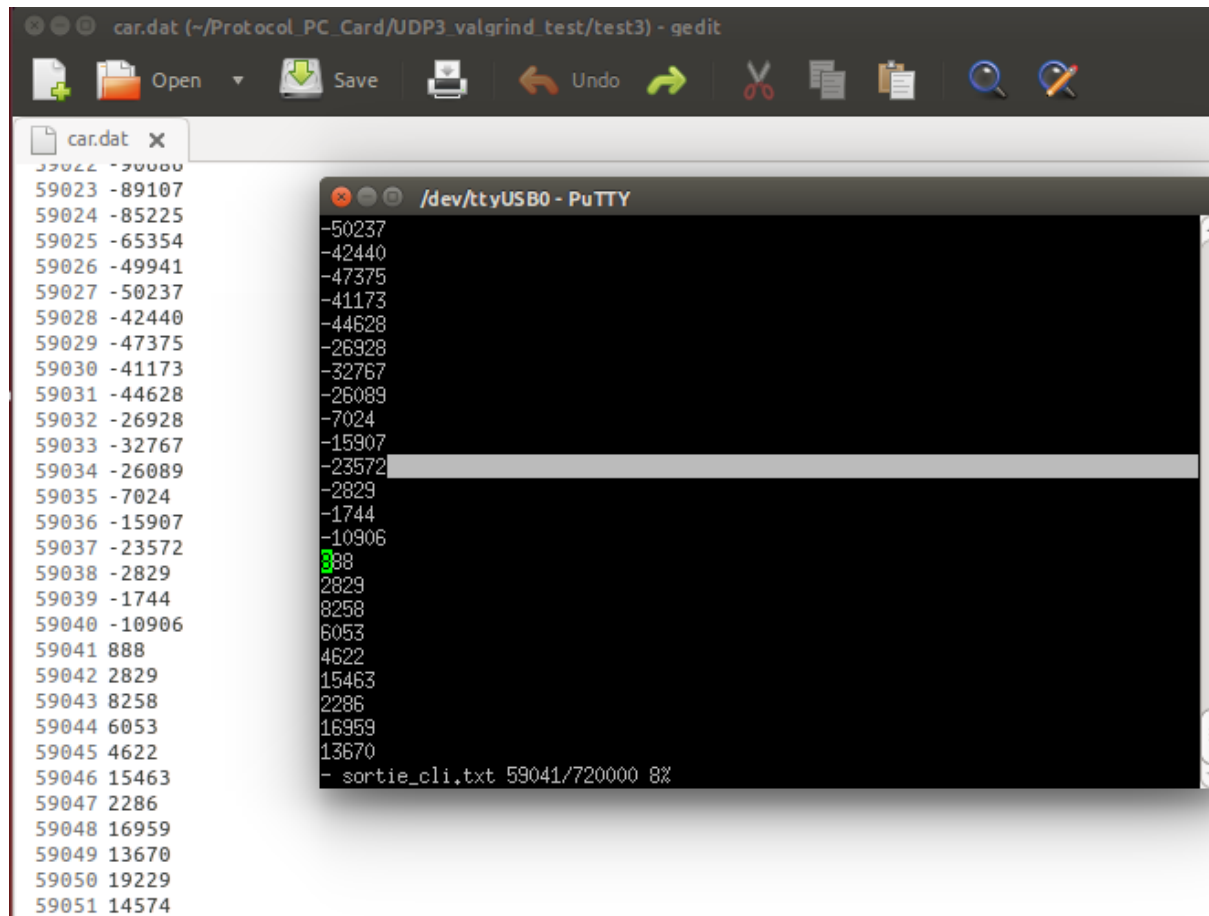
```
/dev/ttyUSB0 - PuTTY
Mem-info:
Normal per-cpu:
CPU 0: hi: 0, btch: 1 usd: 0
active_anon:0 inactive_anon:0 isolated_anon:0
active_file:0 inactive_file:0 isolated_file:0
unevictable:13678 dirty:0 writeback:0 unstable:0
free:254 slab_reclaimable:44 slab_unreclaimable:279
mapped:0 shmem:0 pagetables:0 bounce:0
Normal free:1016kB min:1016kB low:1268kB high:1524kB active_anon:0kB inactive_anon:0kB active_file:0kB inactive_file:0kB unevictable:54712kB isolated(anon):0kB isolated(file):0kB present:65024kB mlocked:0kB dirty:0kB writeback:0kB mapped:0kB shmem:0kB slab_reclaimable:176kB slab_unreclaimable:1116kB kernel_stack:152kB pagetables:0kB unstable:0kB bounce:0kB writeback_tmp:0kB pages_scanned:5 all_unreclaimable? yes
lowmem_reserve[]: 0 0
Normal: 0*4kB 1*8kB 1*16kB 1*32kB 1*64kB 1*128kB 3*256kB 0*512kB 0*1024kB 0*2048kB 0*4096kB 0*8192kB = 1016kB
13680 total pagecache pages
16384 pages of RAM
273 free pages
488 reserved pages
323 slab pages
239 pages shared
0 pages swap cached
Out of memory: kill process 38 (test3_7500_960) score 12 or a child
Killed process 38 (test3_7500_960) vsz:3984kB, anon-rss:0kB, file-rss:0kB
test3_7500_960 invoked oom-killer: gfp_mask=0x200d2, order=0, oom_adj=0
Backtrace: invalid frame pointer 0x00000040
Mem-info:
Normal per-cpu:
CPU 0: hi: 0, btch: 1 usd: 0
active_anon:0 inactive_anon:0 isolated_anon:0
active_file:0 inactive_file:0 isolated_file:0
unevictable:13678 dirty:0 writeback:0 unstable:0
free:254 slab_reclaimable:44 slab_unreclaimable:279
mapped:0 shmem:0 pagetables:0 bounce:0
Normal free:1016kB min:1016kB low:1268kB high:1524kB active_anon:0kB inactive_anon:0kB active_file:0kB inactive_file:0kB unevictable:54712kB isolated(anon):0kB isolated(file):0kB present:65024kB mlocked:0kB dirty:0kB writeback:0kB mapped:0kB shmem:0kB slab_reclaimable:176kB slab_unreclaimable:1116kB kernel_stack:152kB pagetables:0kB unstable:0kB bounce:0kB writeback_tmp:0kB pages_scanned:5 all_unreclaimable? yes
lowmem_reserve[]: 0 0
Normal: 0*4kB 1*8kB 1*16kB 1*32kB 1*64kB 1*128kB 3*256kB 0*512kB 0*1024kB 0*2048kB 0*4096kB 0*8192kB = 1016kB
13680 total pagecache pages
16384 pages of RAM
273 free pages
488 reserved pages
323 slab pages
239 pages shared
0 pages swap cached
Out of memory: kill process 40 (test3_7500_960) score 10 or a child
Killed process 40 (test3_7500_960) vsz:3984kB, anon-rss:0kB, file-rss:0kB
```

Donc on va essayer de reduire la taille de buffer et l'utilise deux threads pour la réception l'écriture.

J'ai changé la taille de buffer et le nombre de paquet à envoyer à 50 et 750 respectivement.

Compiler le programme en utilisant optimisation -O2.

Exécuter le programme et ça marche! Le fichier sortie_cli.txt contient 720 000 valeur, et j'ai comparé les valeurs de 61ème paquets et elles sont correctes. C'est-à-dire qu'il n'y a pas de problème de boucle et de collision, je pense que cela fonctionne bien.



```
car.dat (-/Protocol_PC_Card/UDP3_valgrind_test/test3) - gedit
59022 -50080
59023 -89107
59024 -85225
59025 -65354
59026 -49941
59027 -50237
59028 -42440
59029 -47375
59030 -41173
59031 -44628
59032 -26928
59033 -32767
59034 -26089
59035 -7024
59036 -15907
59037 -23572
59038 -2829
59039 -1744
59040 -10906
59041 888
59042 2829
59043 8258
59044 6053
59045 4622
59046 15463
59047 2286
59048 16959
59049 13670
59050 19229
59051 14574

/dev/ttyUSB0 - PuTTY
-50237
-42440
-47375
-41173
-44628
-26928
-32767
-26089
-7024
-15907
-23572
-2829
-1744
-10906
888
2829
8258
6053
4622
15463
2286
16959
13670
- sortie_cli.txt 59041/720000 8%
```

Les prochaines étapes:

Rajouter la partie débrutage.

Intégrer lecture d'entrée et l'écoute en sortie.