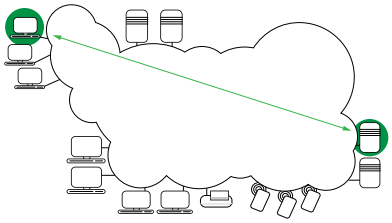
ARes - Lab n°4

Couche transport (1) : TCP et UDP



1 Rappels sur la couche transport (sans machine)

1. Un client web souhaite accéder à un document dont il connaît l'URL. L'adresse IP du serveur concerné est initialement inconnue. Quels protocoles de la couche application sont requis pour satisfaire cette requête ?

2. Quels protocoles de la couche transport sont nécessaires pour satisfaire cette même requête ?

1.1 Protocole en mode non connecté, UDP

1. Rappelez les caractéristiques d'un protocole en mode non connecté.

2. Indiquez les principales caractéristiques du protocole UDP.

3. Pour quelles raisons un développeur pourrait-il préférer le protocole UDP à un autre protocole de transport ?

4. A votre avis, une application peut-elle réaliser un transfert de données fiable si elle repose sur UDP ? Justifiez votre réponse.

1.2 Protocole en mode orienté connexion, TCP

1. Rappelez les caractéristiques d'un protocole en mode orienté connexion.

2. Explicitez les mécanismes nécessaires à la réalisation d'un transfert de données fiable.

3. Indiquez les principales caractéristiques du protocole TCP.

1.2.1 Gestion de connexion

1. Comment sont différenciés les rôles des segments dans le cadre de TCP ?

2. Représentez le diagramme d'établissement d'une connexion réseau. Discutez du nombre de messages nécessaires. Dans le contexte de TCP, pourquoi procéder à un échange en trois phases ?

3. Quelles sont les possibilités de numérotation des segments ? Dans le contexte de TCP, comment évoluent les numéros de séquence ? Deux segments successifs peuvent-ils contenir le même numéro de séquence ? Et en l'absence de données transmises, le numéro de séquence peut-il augmenter ?

4. Pourquoi ne pas commencer la numérotation de séquence à 0 ?

5. Quelles sont les possibilités de terminaison d'une connexion ? Représentez les diagrammes correspondants.

1.2.2 Gestion de la fiabilité

1. La fiabilisation requiert la connaissance de la remise des données. Quelles sont les deux principales techniques d'acquittement ? Précisez leurs intérêts respectifs selon l'importance du trafic contrôlé et précisez laquelle est utilisée par TCP.

2. En cas de perte de données, deux politiques de retransmission sont envisageables : décrivez-les et indiquez celle utilisée avec TCP.

1.2.3 Estimation du RTT d'une connexion

Lorsque l'on utilise le protocole TCP, le choix du RTT (*Round Trip Time*) est important puisque la détection de perte en découle directement et que les divers mécanismes de contrôle qui vont influer sur le débit d'émission en dépendent. Le calcul du RTT peut se faire à l'aide de la formule suivante $RTT = \alpha * RTT_{mesure} + (1 - \alpha) * RTT_{ancien}$ avec α le coefficient de lissage.

1. Comment TCP mesure-t-il le délai aller-retour (RTT_{mesure}) pour un segment donné ?

2. Montrez que l'effet d'une valeur mesurée pour le RTT se réduit exponentiellement avec le temps.

3. Quel est l'intérêt d'utiliser cette formule comparée à une moyenne mobile dans laquelle le RTT est la moyenne calculée sur une fenêtre de longueur L ?

4. Quelles sont les conséquences d'une valeur de α proche de 1 ou proche de 0 ?

5. Quelles sont les précautions à prendre lors de la mesure du délai aller-retour d'un segment donné ?

6. A votre avis, quelle est l'utilité de l'option TCP `timestamp` ? Pourquoi est-il conseillé d'utiliser cette option (on pourra consulter le RFC 1323 pour plus de détails) ?

1.2.4 Calcul du RTO de TCP

1. La première approche pour déterminer la valeur du temporisateur de retransmission RTO (*Retransmission TimeOut*) est $RTO = n * RTT$. Quelles sont les précautions à prendre quant au dimensionnement de n ?

2. La deuxième approche utilise $RTO = RTT + \delta D$ avec généralement $\delta = 4$.
   $D = \beta(|RTT_{mesure} - RTT_{ancien}|) + (1 - \beta)D_{ancien}$ avec généralement $\beta = 1/4$.
   Cette approche consiste à calculer la variance du RTT. Quelle est l'amélioration apportée ?

3. Comment calculer le RTO lorsqu'il y a des pertes ?

2 Observation de trafic UDP

Les analyses qui suivent ont pour but d'observer les mécanismes protocolaires du protocole UDP.

2.1 Caractéristiques d'un datagramme UDP (sans machine)

Voici la trace d'une trame à étudier :

```
0000  00 04 76 21 1b 95 00 01  02 a5 fb 88 08 00 45 00   ..v!.... ......E.

0010  00 30 00 00 40 00 40 11  6d 58 c2 fe a3 b1 c2 fe   .0..@.@. mX......

0020  a3 b6 06 9c 00 45 00 1c  e1 e4 00 01 75 6e 69 78   .....E.. ....unix

0030  62 6f 74 74 00 6e 65 74  61 73 63 69 69 00         bott.net ascii.
```

1. Analysez **manuellement** (sans `wireshark/tshark`) la trame présentée ci-dessus. Utilisez directement le support du lab pour entourer les différents champs sur les traces.
2. Quelles informations peut-on déduire des numéro de ports contenus dans le datagramme ci-dessus ?
3. UDP est dit minimaliste en terme de fonctionnalités. En observant les champs présents dans l'en-tête, pensez-vous que leur nombre soit réduit au maximum ?

2.2 Capture et analyse de datagrammes UDP

2.2.1 Capture d'un trafic UDP

Cette première capture a pour but de percevoir la nature du trafic UDP. Sur la plateforme d'expérimentation, la topologie 1 a été configurée (postes clients et serveur sur le même LAN). Générez du trafic UDP en utilisant TFTP. Réalisez la capture de ce trafic à l'aide du logiciel `wireshark` ou `tshark` :

- A partir du poste PPTI **N**, connectez-vous sur les 3 VM correspondantes de la plateforme à l'aide de 3 fenêtres textuelles
  - fenêtre 1, accédez à la vm**N**1 (hôte "client") via SSH vers `etudiant@10.0.7.`**N**1
  - fenêtre 2, accédez à la vm**N**2 (hôte "sonde") via SSH vers `etudiant@10.0.7.`**N**2 (utilisez `-Y` pour la redirection X11 de `wireshark`)
  - fenêtre 3, accédez à la vm**N**3 (hôte "serveur") via SSH vers `etudiant@10.0.7.`**N**3
- Vérifiez que le serveur TFTP tourne sur `10.0.7.`**N**3 (fenêtre 3)
  - recherchez le processus du serveur, (tapez `ps aux | grep tftp` ou vérifiez dans `inetd.conf`)
  - verifiez qu'un répertoire est configuré pour réaliser les transferts
  - visualisez la configuration des interfaces, en particulier celle sur le LAN d'étude (`/sbin/ifconfig eth1`) pour vérifier l'adresse IPv4 du serveur pour la connexion du client (devrait être `10.`**N**`.1.`**N**3)
- Démarrez la capture en lançant l'analyseur sur `10.0.7.`**N**2 (fenêtre 2)
  - lancez l'analyseur, tappez : `wireshark`
  - initiez la capture sur l'interface `eth1`, comme indiqué dans le **Lab n°1**
- Démarrez un client TFTP sur `10.0.7.`**N**1 (fenêtre 1)
  - lancez le client en établissant une connexion vers le serveur, tapez : `tftp 10.`**N**`.1.`**N**3

  - récupérez un fichier du serveur avec la commande `get`
  - terminez l'échange avec la commande `quit`

- Observez la capture dans la fenêtre de `wireshark`
- **Filtrez le trafic afin de ne conserver que celui relatif à UDP** (filtre = `udp`). Enregistrez la trace filtrée pour pouvoir la ré-utiliser ultérieurement. Ne quittez pas l'application afin de pouvoir démarrer les analyses qui suivent.

2.2.2 Analyse de l'échange UDP

1. Par rapport à la capture des échanges TFTP du **Lab n°2** (avec filtre = `tftp`), quelles différences observez-vous avec la trace **affichée** ici ?

2. Pouvez-vous identifier les rôles de client ou de serveur des applications impliquées ?

3. Comment est gérée l'association des deux applications impliquées ?

4. UDP n'intégrant pas de mécanismes de fiabilité, que pouvez-vous dire des mécanismes de protection mis en place par les applications ?

2.2.3 Sans la plateforme...

En cas de problème d'accès à la plateforme d'expérimentation ou si vous souhaitez réviser sur une autre machine, vous pouvez télécharger la trace `tme4-udp.dmp` (similaire à celle capturée précédemment) soit à partir du répertoire `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, soit sur la page web `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, puis l'analyser avec le logiciel `wireshark` (sans avoir besoin des droits d'administrateur).

3 Observation du trafic TCP

Cette seconde analyse a pour but d'observer les différents mécanismes protocolaires de TCP. Pour cela, nous nous appuierons sur des captures de segments TCP.

3.1 Mise en place de la connexion TCP (sans machine)

Voici la première trame échangée lors de l'ouverture d'une connexion :

```
0000  00 50 7f 05 7d 40 00 10  a4 86 2d 0b 08 00 45 00   .P..}@.. ..-...E.
0010  00 3c 17 96 40 00 40 06  6d f3 0a 21 b6 b2 c0 37   .<..@.@. m..!...7
0020  34 28 84 b3 00 50 b6 94  b0 b7 00 00 00 00 a0 02   4(...P.. ........
0030  16 d0 e8 23 00 00 02 04  05 b4 04 02 08 0a 00 6f   ...#.... .......o
0040  a7 21 00 00 00 00 01 03  03 00                     .!...... ..
```

1. Analysez **manuellement** (sans `wireshark`/`tshark`) la trame présentée ci-dessus. Utilisez directement le support du Lab pour entourer les différents champs sur les traces.
2. Quels sont les bits de contrôle (TCP *flags*) positionnés ? Que signifient-ils ?
3. Identifiez les hôtes impliqués dans cet échange. Quels vont être leurs rôles respectifs dans la suite ?
4. Quelles informations peut-on déduire des numéro de ports contenus dans les segments ci-dessus ?
5. Rappelez le fonctionnement des numéros de séquence de TCP. Justifiez les valeurs présentes dans ce segment.
6. Pouvez-vous observer des options dans l'en-tête TCP ? Si oui, que signifient-elles ?

3.2 Capture et analyse d'une connexion TCP

3.2.1 Capture d'un trafic TCP

Cette seconde capture a pour but de percevoir la nature du trafic TCP. Sur la plateforme d'expérimentation, toujours sur la topologie 1 (postes client et serveur sur le même LAN), réalisez la capture de trafic HTTP à l'aide du logiciel `wireshark` ou `tshark` :

- A partir du poste PPTI **N**, connectez-vous sur les 3 VM correspondantes de la plateforme à l'aide de 3 fenêtres textuelles (si ce n'est déjà fait).

- Vérifiez que le serveur HTTP (`apache2`) tourne sur `10.0.7.`**N**3 (fenêtre 3)
- Démarrer la capture en lançant l'analyseur sur sur l'interface `eth1` de l'hôte `10.0.7.`**N**2 (fenêtre 2)
- Démarrez un client HTTP sur `10.0.7.`**N**1 (fenêtre 1)
  - lancez le client de votre choix (`firefox`, `wget`...)
  - établissez une connexion vers le serveur en spécifiant l'URL suivante : `http://10.`**N**`.1.`**N**3 (affiche la page par défaut du serveur Apache)
- Observez la capture dans la fenêtre de `wireshark`
- **Filtrez le trafic afin de ne conserver que celui relatif à TCP** (filtre = `tcp`). Enregistrez la trace filtrée pour pouvoir la ré-utiliser ultérieurement. Ne quittez pas l'application afin de pouvoir démarrer les analyses qui suivent.

3.2.2 Analyse de l'échange TCP

1. Par rapport à la capture des échanges HTTP du **Lab n°2** (avec filtre = `http`, ou plutôt `tcp.port == 80 and tcp.len > 0` une fois que l'on s'est assuré que c'était bien le port adéquat), quelles différences observez-vous avec la trace **affichée** ici ?

2. Quels sont les bits de contrôle (TCP *flags*) positionnés dans les différentes trames ? Que signifient-ils ?

3. Vérifiez le fonctionnement des numéros de séquences de TCP. **Attention,** `wireshark` **utilise une numérotation relative.** Comparez les valeurs réelles lues dans la trame et celles données par `wireshark` pour les numéros de séquence et d'acquittement. Justifiez les valeurs présentés.

4. Que pouvez-vous dire du contrôle de flux pour les segments étudiés ?

5. Pouvez-vous trouvez d'autres options dans les entêtes TCP ? Si oui, que signifient-elles ?

3.2.3 Sans la plateforme...

En cas de problème d'accès à la plateforme d'expérimentation ou si vous souhaitez réviser sur une autre machine, vous pouvez télécharger la trace `tme4-tc1.dmp` (similaire à celle capturée précédemment) puis l'analyser avec le logiciel `wireshark` (sans avoir besoin des droits d'administrateur).

3.3 Analyse détaillée d'un échange TCP

Utilisez la trace de sauvegarde `tme4-tc1.dmp` proposée ci-dessus, puis analysez-la sur le poste PPTI avec le logiciel `wireshark` exécuté sans les droits d'administrateur.

Ci-dessous sont affichées les premières trames de cette trace partiellement décodée grâce à l'outil UNIX `tcpdump` (basé sur `libpcap`, la même bibliothèque de capture que `wireshark`/`tshark`, mais plus adaptée pour une présentation textuelle) :

```
00:00:00.000000 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [S], seq 3063197879, win 5840,
[A]                                    options [mss 1460,sackOK,TS val 7317281 ecr 0,nop,wscale 0], length 0

00:00:00.170558 IP 192.55.52.40.80 > 10.33.182.178.33971: Flags [S.], seq 610765288, ack 3063197880, win 64240,
[B]                                    options [mss 1402,nop,wscale 0,nop,nop,TS val 0 ecr 0,nop,nop,sackOK], length 0

00:00:00.170618 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [.], ack 1, win 5840,
[C]                                    options [nop,nop,TS val 7317298 ecr 0], length 0

00:00:00.170819 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [P.], seq 1:486, ack 1, win 5840,
[D]                                    options [nop,nop,TS val 7317298 ecr 0], length 485

00:00:00.370505 IP 192.55.52.40.80 > 10.33.182.178.33971: Flags [.], seq 1:1391, ack 486, win 63755,
[E]                                    options [nop,nop,TS val 19332362 ecr 7317298], length 1390

00:00:00.370560 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [.], ack 1391, win 8340,
[F]                                    options [nop,nop,TS val 7317318 ecr 19332362], length 0

00:00:00.381289 IP 192.55.52.40.80 > 10.33.182.178.33971: Flags [.], seq 1391:2781, ack 486, win 63755,
[G]                                    options [nop,nop,TS val 19332362 ecr 7317298], length 1390

00:00:00.381336 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [.], ack 2781, win 11120,
[H]                                    options [nop,nop,TS val 7317319 ecr 19332362], length 0

...
```

1. Tracez **précisément** le chronogramme correspondant à cet échange **(Respectez impérativement l'échelle des temps pour réussir à visualiser l'évolution des échanges)**.

2. Nous souhaitons étudier la demi-connexion correspondant à l'émission de données du serveur (`192.55.52.40.www`) vers le client (`10.33.182.178.33971`). Complétez les dernières lignes du tableau suivant (numéros de séquence relatifs) :

action	base de la fenêtre	pointeur d'émission	fin de la fenêtre	taille de la fenêtre	*commentaire*
réception `A`	—	—	—	5840	*win 5840*
émission `B`	0 (610765288)	1	5840	—	*SYN, +1 pas d'émission*
réception `C`	1	1	5841	5840	*ACK*
réception `D`	1	1	5841	5840	*ACK*
émission `E`	1	1391	5841	—	
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					

3. Commentez l'évolution des numéros de séquence.

4. Que pouvez-vous dire sur la gestion des tampons ?

5. Observez-vous de nouvelles options ? Pouvez-vous les expliquer ?

6. Que pouvez-vous dire à propos de la génération des acquittements par le récepteur ?

7. Comment se termine la communication ? Détaillez les échanges finaux.

4 Echanges TCP imbriqués

4.1 Capture et analyse d'un échange avec deux connexions TCP imbriquées

4.1.1 Capture du trafic de deux connexions TCP imbriquées

Cette dernière capture a pour but de percevoir l'entrelacement des deux connexions TCP associées à l'application FTP. Sur la plateforme d'expérimentation, toujours sur la topologie 1 (postes client et serveur sur le même LAN), réalisez la capture de trafic FTP à l'aide du logiciel `wireshark` ou `tshark` :

- A partir du poste PPTI **N**, connectez-vous sur les 3 VM correspondantes de la plateforme à l'aide de 3 fenêtres textuelles.
- Vérifiez que le serveur FTP (`ftpd`) tourne sur `10.0.7.`**N**3 (fenêtre 3)
- Démarrez la capture en lançant l'analyseur sur sur l'interface `eth1` de l'hôte `10.0.7.`**N**2 (fenêtre 2)
- Démarrez un client FTP sur `10.0.7.`**N**1 (fenêtre 1)
  - lancez le client en établissant une connexion vers le serveur, tapez : `ftp 10.`**N**`.1.`**N**3 (réseau d'expérimentation)
  - identifiez-vous avec le login `etudiant` et le mot de passe correspondant
  - choisissez un fichier et transférez le sur la machine client (commandes de l'interface utilisateur `get`)
  - terminez l'échange (commandes de l'interface utilisateur `quit`)
- Observez la capture dans la fenêtre de `wireshark`
- **Filtrez le trafic afin de ne conserver que celui relatif à TCP** (filtre = `tcp`). Enregistrez la trace filtrée pour pouvoir la ré-utiliser ultérieurement. Ne quittez pas l'application afin de pouvoir démarrer les analyses qui suivent.

4.1.2 Analyse du trafic de deux connexions TCP imbriquées

Retrouvez la correspondance entre les messages échangés sur le réseau, sur la connexion de contrôle de FTP et ceux affichés par l'application (interface utilisateur sur le client).

1. Observez l'échange capturé et expliquez les actions réalisées au niveau applicatif.

2. Tracez le chronogramme correspondant à ces échanges en utilisant une couleur différente par connexion.

3. Que pouvez-vous dire de l'utilisation du *flag* `PUSH` ?

4.1.3 Sans la plateforme...

En cas de problème d'accès à la plateforme d'expérimentation ou si vous souhaitez réviser sur une autre machine, vous pouvez télécharger la trace `tme4-tc2.dmp` (similaire à celle capturée précédemment) puis l'analyser avec le logiciel `wireshark` (sans avoir besoin des droits d'administrateur).

5 Avant de quitter la salle

- Si vous avez enregistré des captures sur la VM "sonde", n'oubliez pas de les rapatrier sur votre compte utilisateur de la PPTI. Tapez la commande suivante dans un terminal local du PC de la PPTI : `scp etudiant@10.0.7.N2:<trace> <dest>`
- Avant de terminer vos connexions sur les équipements de la plateforme, supprimez vos fichiers et modifications effectuées afin de retrouver l'état initial.