Protocole UDP

Introduction aux réseaux locaux

Plan de la présentation

- Rappel
- Détails du segment UDP
- Ordonnance des bits

Protocole UDP

Rappel

- UDP (User Datagram Protocol) est un protocole sans connexion, donc sans processus initial d'entente (handshaking).
- UDP:
  - ne garantie pas l'envoie et la réception
  - n'envoie pas d'accusé réception
  - ne garantie pas l'ordre d'arrivé des paquets
  - intègre un système de contrôle de somme
  - est plus léger que TCP et sert aux applications qui sont tolérantes aux fautes et erreurs de transmissions.

En-tête UDP

offset (bits)	0 – 15	16 – 31
0	Source Port Number	Destination Port Number
32	Length	Checksum
64+	Data	

- L’en-tête est composée de 64 bits (4 paquets de 2 octets) spécifiant:
  - Le port de la source et de la destination
  - La longueur du segment et la somme de contrôle
- Les données correspondent au message de l’application

En-tête UDP

- Source 16 bits:
  - Ce champ optionnel identifie le port de l'expéditeur (la source). Si il n'est pas utilisé, il doit contenir la valeur 0.
- Destination 16 bits:
  - Contient le port de destination.
- Longueur 16 bits
  - Détermine la longueur totale (en-tête et message) du segment.
- Somme de contrôle 16 bits
  - Optionnel si le transport est sur IPv4 ce champ contient le checksum du segment. Si non utilisé il doit être zéro,

Longueur maximale

- Étant donné que la longueur est exprimée sur 16 bits la longueur maximale théorique du message sera de
  - 2^16 = 65535 – 8 octets pour l’entête du segment = 65527 octets
- Par contre, en pratique, sur IPv4, une en-tête IP de 20 octets est ajouté au Datagram, ce qui limitera, en pratique, l’envoie de message de taille supérieure à 65507 octets.

Ordonnance des bits

Ordonnance des bits

- Lorsqu'un mot sur 32 bits est déposé en mémoire il peut être déposé de deux façons:
  - L'octet ayant le poids le plus fort en début
  - **Big endian**
  - L'octet ayant le poids le plus faible au début
  - **Little Endian**
- Les processeurs Intel™ utilise la nomenclature little-endian
- Les processeurs Sparc™ et Arm™ utilise du big endian.

Exemple

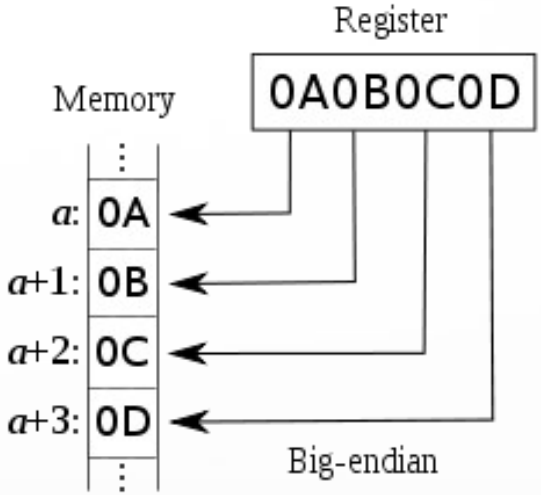
- En mémoire, si on suppose le mot de 32 bits (4 octets) suivants:
  - 0x0A010B02 (le nombre : 167 840 514)
- Sera représenté comme suit sur une architecture :
  - Little endian

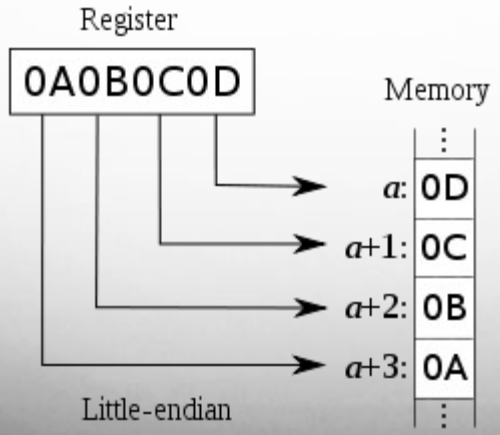
```
02  0B  01  0
                A
0   01  0B  02
A
```

  - Big endian :

Big vs little

- Octet avec le poids le plus élevé au début





- Octet avec le poids le plus faible au début

Endianness de la couche transport ?

- Étant donné que la couche transport (et les autres couches en dessous) assumerons du Big Endian, sur Intel (little endian), il faudra utiliser les fonctions de conversion suivantes pour vos spécifications de port et autres données lues:
  - htons() : "Host to Network Short"
  - htonl() : "Host to Network Long"
  - ntohs() : "Network to Host Short"
  - ntohl() : "Network to Host Long“
- Short = 16 bits et Long = 33 bits.

Exemple de traduction

```
int main(void)

{

 // Coté de l'expéditeur
 int valeurLI = 15000; //(0x00003A98)
 int valeurBI = htonl(valeurLI); // valeurBI vaut maintenant
0x983a0000

 // Coté du récepteur
 int valeurRecu = ntohl(valeurBI); // De retour à 0x00003A98,
15000

 return 0;

}
```

Quand utiliser la conversion?

- Lorsque vous spécifier un élément de la structure du segment (port, etc.)
- Lorsque vous passez, à l'intérieur du message, des valeurs numérique plus grandes que 8 bits (16 ou 32 bits) et que vous désirez être 'portable' entre les différentes architectures.
  - Si vous assumez que le programme roule seulement sous Intel, vous n'avez pas besoin de vous préoccuper de l'ordre des bits dans le message.
  - Même chose, si vous transférez du texte, l'ordre des octets est la même pour toutes les architectures.

À maitriser

- Les détails du segment UDP
- Différence entre Big et Little Endian.

Des questions?

Sources

- http://en.wikipedia.org/wiki/User_Datagram_Protocol
- http://en.wikipedia.org/wiki/Endianness