Couches hautes du réseau

- Généralités
- La couche session
  - gestion, synchronisation
- La couche présentation
  - chiffrement, compression, transcription

Couches hautes - Généralités

7	Application *Application*
6	Présentation *Presentation*
5	Session *Session*
4	Transport *Transport*
3	Réseau *Network*
LLC	Contrôle de lien logique *Logical Link Control*
MAC	Contrôle d'accès au médium *Medium Access Control*
1	Physique *Physical*

Couches hautes ➔ logicielles

couche 5 et 6, non obligatoires

➔ valeur ajoutée

Couches basses ➔ plus matérielles que logicielles

➔ permettent d'acheminer les données d'un point A à un point B

La couche session (1)

• Une session est un ensemble de transactions entre deux unités du réseau ou plus. Elle permet d'établir *une connexion logique* entre deux systèmes.

• Afin qu'une conversation se déroule correctement entre plusieurs individus, il est impératif de mettre en place des règles qui permettent de gérer le dialogue entre ces individus.

Cette notion de *contrôle du dialogue* est une des fonctionnalités de la couche session

Autres fonctionnalités : *synchronisation, gestion des activités*

→ Par conséquent, 3 fonctionnalités "importantes"

La couche session (2)

- Le rôle de la couche session est **d'ouvrir, gérer et fermer les sessions entre les applications**. Donc, il y a trois phases dans une session :
    - établissement d'une session
    - transfert des données (synchronisation du dialogue)
    - libération de la session (fermeture)

La couche session s'appuie sur la couche transport pour fonctionner. En cas de problème de la couche 4, la couche session peut relancer une connexion au niveau transport sans perdre la connectivité au niveau session.

La couche session (3)

- **Le contrôle du dialogue**

2 types de communication possible :

- bidirectionnelle simultanée

→ géré par les couches inférieures

- bidirectionnelle alternée

→ géré par la couche session

Utilisation d'un **jeton** pour avoir le droit de parole

•pas de collision possible au niveau dialogue

•divers types de jeton (data, activity major,…)

La couche session (4)

- **La synchronisation du dialogue**

  But : permettre aux individus communicants de marquer une pause pour sauvegarder la communication en cours et re synchroniser le dialogue

  (utile lors d'un problème au niveau couche 4, saturation disque dur,…)

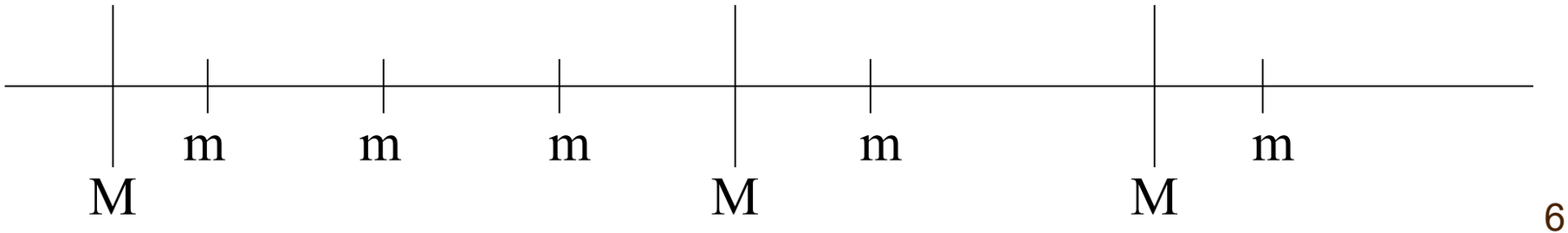
  ➡ Utilisation de "point de synchronisation"

→ sauvegarde les fichiers donnés, les paramètres réseau, les paramètres horloges

2 sortes de synchronisation :

- majeur : avec acquittement de l'autre entité
- mineur : sans acquittement

La retransmission peut reprendre à partir d'un point mineur, mais pas au-delà d'un point majeur.

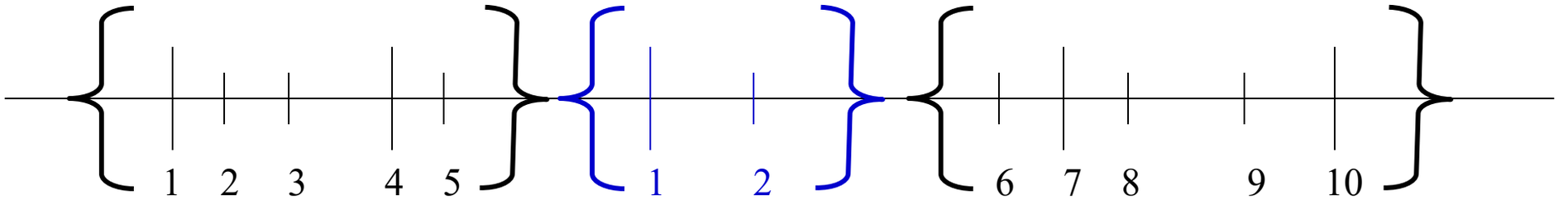


La couche session (5)

- **La gestion des activités**

But : gérer la communication simultanée entre plusieurs tâches (activités)

exemple : l'envoi de plusieurs fichiers,…



La couche présentation (1)

• But : présenter les informations dans un format que l'utilisateur est en mesure d'interpréter

• 3 fonctionnalités :

  • Compression des données

  • Chiffrement des données

  • Transcription (formatage) des données

La couche présentation (2)

- **Compression des données**

→ diminuer la quantité de données envoyée sur le réseau

(minimiser la bande passante utilisée)

2 sortes de compression : { - sans perte (souvent basé sur Huffman)
- avec perte

Quelques normes de compression :

- images :

TIFF (Tagged Image File Format), JPEG (Joint Photographic Experts Group), PICT (Mac), PNG (**Portable Network Graphics**), …

- son/vidéo :

MPEG 1 layer 3, MIDI (Musical Instrument Digital Interface), MPEG (Motion Picture Experts Group) 1, 2 ou 4, QuickTime, …

La couche présentation (3)

- **Chiffrement des données**

→ éviter que les données puissent être facilement lues

- chiffrement — $E_k(\text{message}) = \text{code}$
- déchiffrement — $D_{k'}(\text{code}) = \text{message}$

E	algorithme de chiffrement
D	algorithme de déchiffrement
k	clé de chiffrement
k'	clé de déchiffrement

- 2 sortes de chiffrement : symétrique/asymétrique

La couche présentation (4)

- **Transcription des données**

  → représentation des informations échangées entre systèmes

  • préservation de la sémantique des données échangées

  - même si on travaille sur des systèmes différents, les entiers, caractères,… doivent être traités de la même manière

  • négociation de la **syntaxe de transfert**

  - syntaxe identique entre les entités communicantes
  - quelques exemples : MIME, XDR, BER (Basic Encoding Rule)

  • accès des applications aux services de la couche session

Comment créer une syntaxe de transfert le plus simplement possible ?

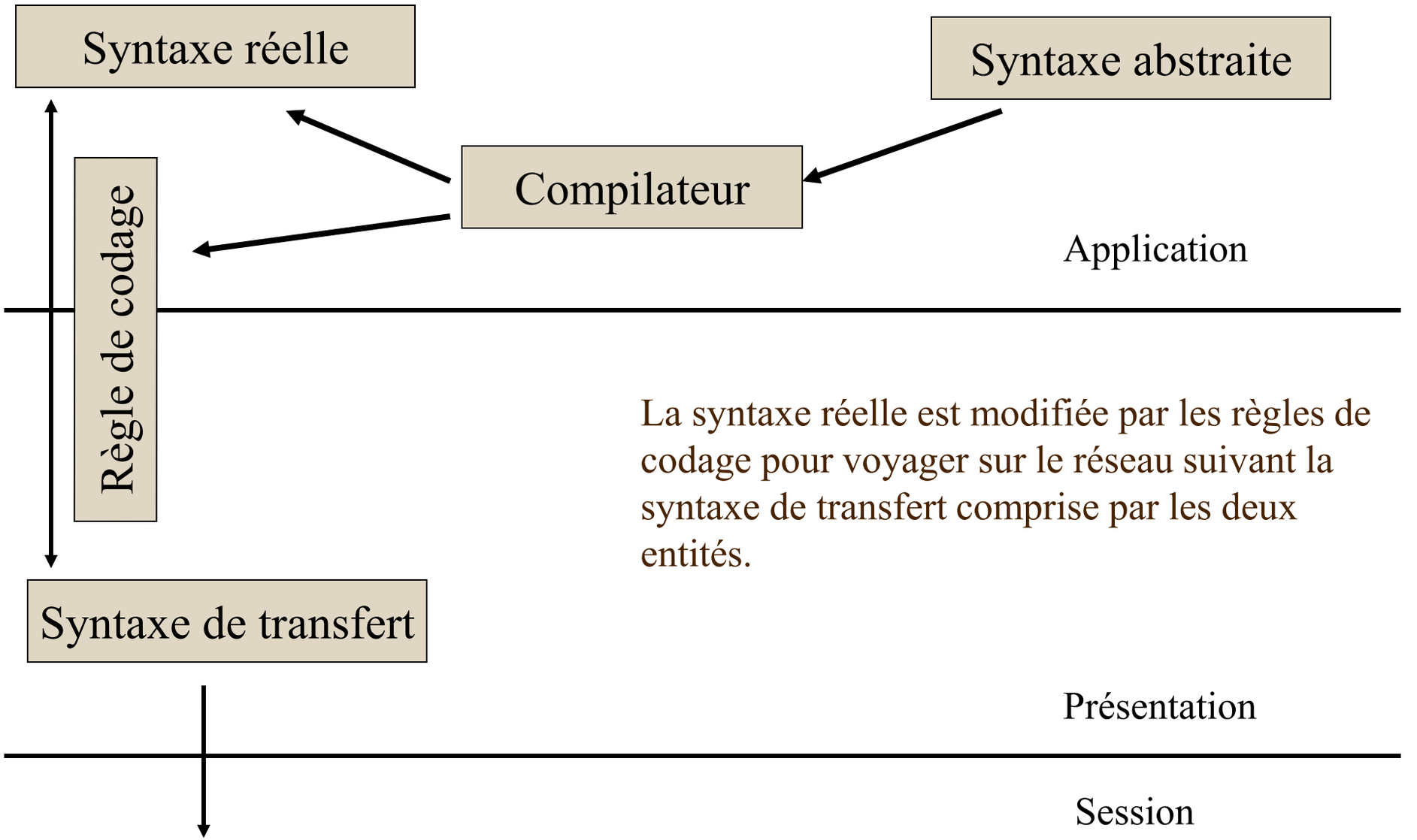
La couche présentation (5)

- La syntaxe abstraite
  - Définir un modèle supérieur décrivant l'application qui soit:
    - indépendant vis à vis du processeur
    - indépendant vis à vis du système d'exploitation
    - indépendant vis à vis du langage programmation
  - et qui permette après compilation de devenir une syntaxe de transfert

ASN.1 : Abstract Syntax Notation number 1

➔ permet de spécifier les données échangées dans un protocole de communications
(norme ISO 8824, ou ITU-T X.680)

La couche présentation (6)



La syntaxe réelle est modifiée par les règles de codage pour voyager sur le réseau suivant la syntaxe de transfert comprise par les deux entités.

La couche présentation (7)

Définitions

Une ***syntaxe abstraite*** est la représentation sous forme abstraite, indépendante de la représentation réelle, des données de chaque application; c'est une spécification de données de la couche application en appliquant des règles de notation indépendantes de la technique de codage utilisée pour représenter ces données.

Une ***syntaxe de transfert*** est un ensemble de règles de codage (et de décodage) permettant de générer un flot d'octets à partir d'une syntaxe abstraite ou réelle.

Un ***contexte de présentation*** est l'association d'une syntaxe abstraite et d'une syntaxe de transfert.

Rôle de la couche présentation : **choisir un contexte de présentation**

→ à une syntaxe abstraite peut-être associé plusieurs syntaxes de transfert

choisir une syntaxe de transfert commune aux n entités

ASN.1 (1)

- Types de base ou construit
  - types de base

			N° Tag
Integer	entier de taille quelconque	10; -5; 24526228,..	02
Boolean	booléen	true/ false	01
Bitstring	chaînes de bits de longueur quelconque	1010100010010	03
Octetstring	chaînes d'octets de longueur quelconque	0A29B6D56	04
IA5string	chaînes ASCII de longueur quelconque	Code	16

  - types construit ou structuré
    - utilisation de : Sequence {…} : collection ordonnée de divers types
      Sequence of {…} : collection ordonnée d'un seul type
      Set {…} : collection non ordonnée de divers types
      Set of {…} : collection non ordonnée d'un seul type
      Choice {…} : un parmi la liste proposée

ASN.1 (2)

- Exemple

```
Age  ::= INTEGER (0..7)
User ::= SEQUENCE {
         name    IA5String (SIZE(1..128)),
         age     Age,
         address IA5String OPTIONAL,
         ...
    }
```

- Divers :
  - le mot clé *implicit* permet d'éviter de spécifier dans la syntaxe de transfert le type de la donnée (il est toujours couplé avec la notion d'étiquetage)
    exemple : texte :: = implicit IA5string
  - L'étiquetage permet de faire correspondre un numéro à un type
    exemple : texte :: = [10] implicit IA5string

  ➡ cela permet de se référer seulement au type 10, qui correspond à une suite de caractères ASCII

Exemple utilisation ASN.1 (1)

- Description de la fonction GET de HTTP en C++ et ASN.1

en C++ :

```
struct GetRequest { int headerOnly; // booléen
                    int lock; // booléen
                    string url; // l'url
                    AcceptTypes* acceptTypes; } // liste des types acceptés

struct AcceptTypes { List<bitset>* standardTypes; // liste type standard
                                 List<string>* otherTypes; }// autre type
```

En ASN.1 :

```
GetRequest ::= [APPLICATION 0] IMPLICIT SEQUENCE {
                    headerOnly BOOLEAN,
                    lock BOOLEAN,
                     acceptTypes AcceptTypes OPTIONAL
                     url OCTET STRING, }

AcceptTypes ::= [APPLICATION 1] IMPLICIT SEQUENCE {
                    standardTypes [0] IMPLICIT SEQUENCE OF BIT STRING {html(0),plain-text(1),gif(2), jpeg(3)},
                    otherTypes [1] IMPLICIT SEQUENCE OF OCTET STRING OPTIONAL }
```

ASN.1 (3)

- Transformation de ASN.1 en BER (Basic Encoding Rule)
  - Codage T L V (option) : **Type Longueur Valeur** (fin de contenu –option)

Type sur 8 bits : 3 champs :

1. la classe (2 bits) : 00 = universel, 01 = application,
   10 = contextuel, 11 = privé
2. méthode de codage (1bit) : 0 codage de base, 1 sinon
3. N° étiquette (5 bits) : valeur binaire du tag ou étiquette

Longueur : 1 octet si longueur < 128, n sinon
(1er bit à 0, lg sur 1 octet, autrement , lg supérieure)
si octet = 128 , longueur indéfini -> voir la fin du décodage

Valeur : valeur de la variable

- Exemple:

(binaire)	00 0 00010	00000001	00001000	} entier valant 8
(Hexa)	02	01	08	

ASN.1 (3 bis)

- Transformation de ASN.1 en DER (Distinguished Encoding Rule)
  - Codage T L V
    - Sous-ensemble de BER (plus restrictif)
    - Une seule écriture possible pour une data (ex : booléen)
    - Longueur toujours définie
    - Etc…

Utilisation pour les certificats X509.

ASN.1 (4)

Evolution de ASN.1 en XER

- Xml Encoding Rule
  - Utilisation de tag

Exemple : `Age ::= INTEGER (0..7)`

`firstGrade Age ::= 6`

codage : <?xml version="1.0" encoding="UTF-8"?>

<Age>6<Age>

Product ::= SEQUENCE {

size INTEGER,

price REAL (WITH COMPONENTS {..., base(10)})

(ALL EXCEPT (-0 | MINUS-INFINITY | PLUS-INFINITY | NOT-A-NUMBER)) }

value Product ::= { size 44, price 19.95 }

codage : <?xml version="1.0" encoding="UTF-8"?>

<Product size="44" price="19.95"/>

Exemple utilisation ASN.1 (1)

- **En ASN.1 :**

```
GetRequest ::= [APPLICATION 0] IMPLICIT SEQUENCE {
                headerOnly BOOLEAN,
                lock BOOLEAN,
                 acceptTypes AcceptTypes OPTIONAL
                 url OCTET STRING, }

AcceptTypes ::= [APPLICATION 1] IMPLICIT SEQUENCE {
               standardTypes [0] IMPLICIT SEQUENCE OF BIT STRING {html(0),plain-text(1),gif(2), jpeg(3)},
               otherTypes [1] IMPLICIT SEQUENCE OF OCTET STRING OPTIONAL }
```

- **Requête à envoyer :**

```
GetRequest (headerOnly TRUE :lock FALSE :acceptTypes (AcceptTypes :standardTypes ((html) (plain-text)))
:url "/cours/index.html")
```

Type sur 8 bits : 3 champs : puis L et V

1. la classe (2 bits) : 00 = universel, 01 = application, 10 = contextuel, 11 = privé
2. méthode de codage (1bit) : 0 codage de base, 1 sinon
3. N° étiquette (5 bits) : valeur binaire du tag ou étiquette

Exemple utilisation ASN.1 (2)

• Requête à envoyer :

GetRequest (headerOnly TRUE :lock FALSE :acceptTypes (AcceptTypes :standardTypes ((html) (plain-text))) :url "/cours/index.html")

En encodage BER :

GetRequest -> 01100000 -> 0x60

Longueur -> … (0x80 indéfini ???)

GetRequest.headerOnly -> 0x01 0x01 0x01

GetRequest.lock -> 0x01 0x01 0x00

GetRequest.types -> 0x61 0x05

types.standardTypes -> 0xa0 0x03

liste des types -> 0x03 0x01 0xC0

GetRequest.url -> 0x04 0x11 [/cours/index.html]

Taille totale : 33 octets dont 17 pour l'url

Exemple utilisation ASN.1 (3)

• Autre encodage possible : **PER**
Packed Encoding Rule → permet de diminuer la quantité de données envoyées

On met directement certaine valeur…

GetRequest (headerOnly TRUE :lock FALSE :acceptTypes (AcceptTypes :standardTypes ((html) (plain-text)))
:url "/cours/index.html")

[1] --booleen headeronly true
[0] –booleen lock false
[1] – accept type est present
[1] – standard type est present
[0] – other type est absent
[000] – alignement au niveau octet
[00000010] – nombre de type accepté
[1000] – type html
[0100] – type plain-text
[00010001] – taille de l'url
/cours/index.html taille totale : 21 octets dont 17 pour l'url