SÉANCE 17

COMMUNICATION INTERPROCESSUS

SUJETS

Introduction à la Communication Interprocessus

Unicast et Multicast

Passage de Message (Synchrone et Asynchrone)

Représentation des Données Communiquées

Protocoles de Communication

- Exemple de Protocole: HTTP

COMMUNICATION INTERPROCESSUS

uOttawa
L'Université canadienne
Canada's university

Échange de données entre deux ou plus processus/threads indépendants

Le système d'exploitations fournit plusieurs mécanismes pour la Communication InterProcessus (CIP):

- Files
- Sémaphores
- Mémoire partagée

Ces mécanismes sont utilisés pour réaliser la CIP à un plus haut niveau d'abstraction (ex. *send* et *receive*)

CIP – UNICAST ET MULTICAST

uOttawa
L'Université canadienne
Canada's university

L'informatique distribuée implique deux ou plusieurs processus engagés dans la CIP

- Elle utilise un protocole prédéfini

Un processus peut agir comme expéditeur (sender) à un moment et récepteur (receiver) à un autre

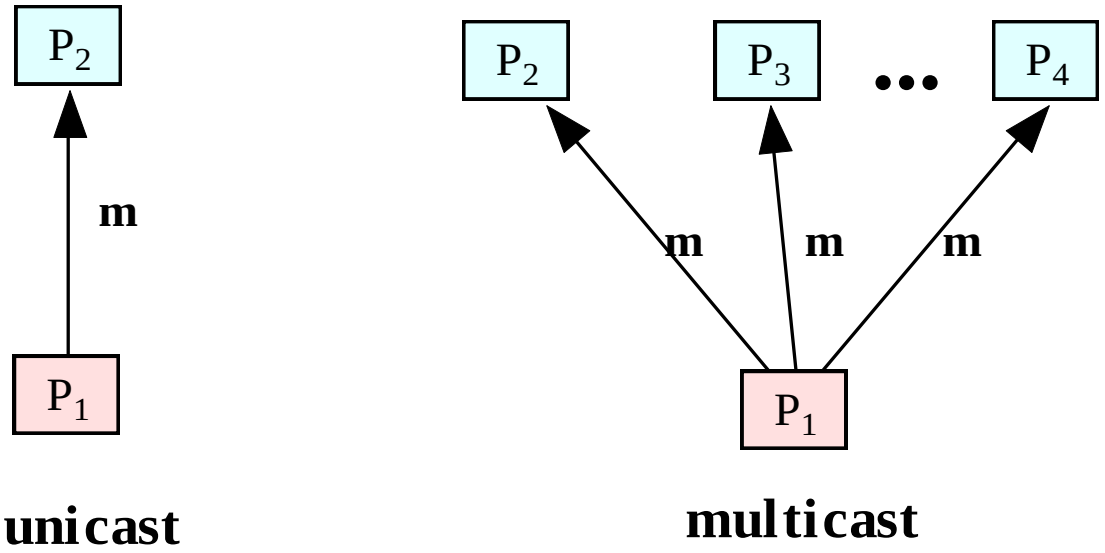
Unicast: communication entre deux processus

- Ex., communication de *socket*

Multicast: communication entre un processus et un groupe de processus

- Ex., modèle de communication publisher/subscriber

UNICAST VS. MULTICAST



OPÉRATIONS TYPIQUES D' UN API DE CIP

uOttawa
L'Université canadienne
Canada's university

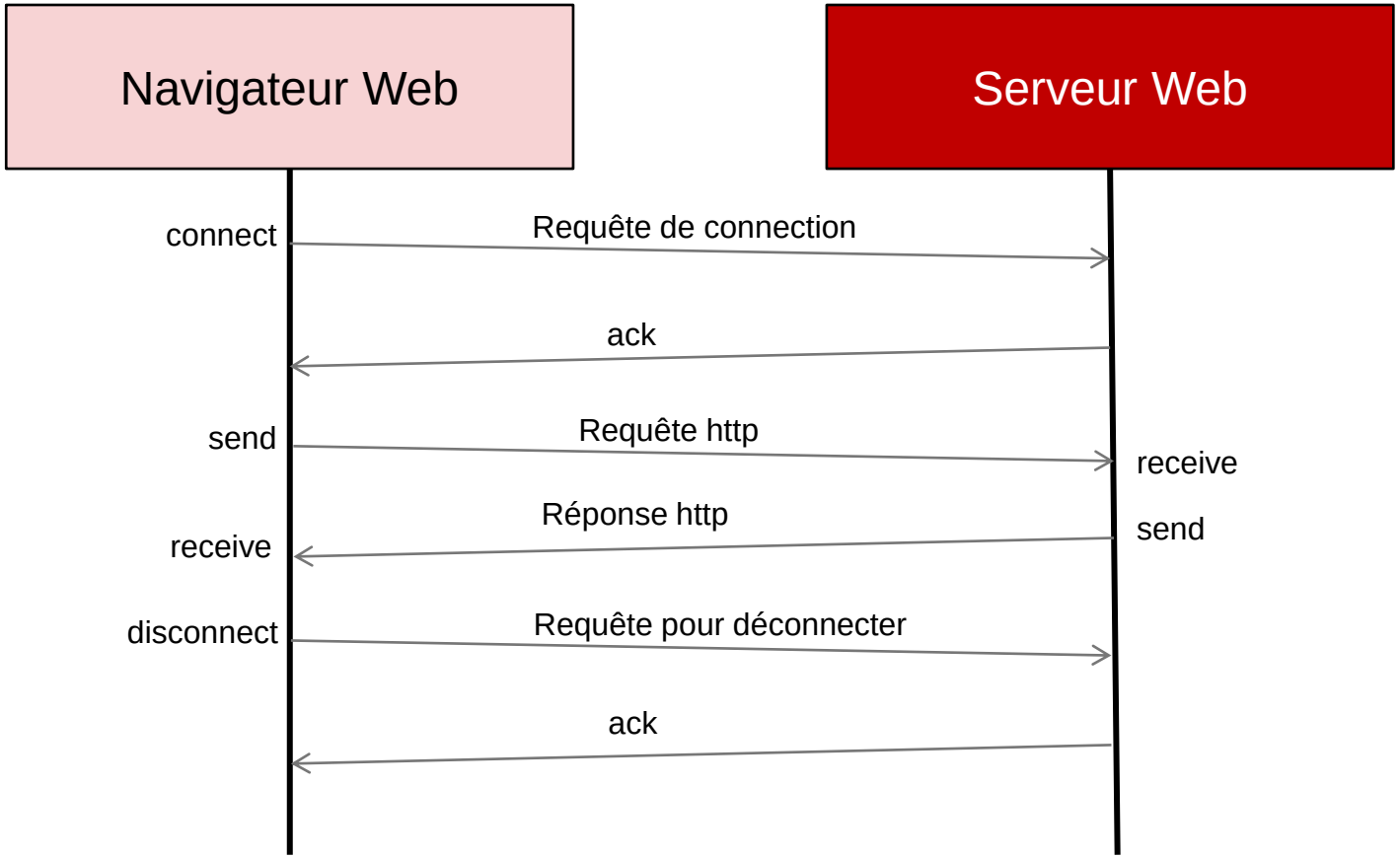
Receive ([sender], message storage object)

Send ([receiver], message)

Connect (sender address, receiver address), *pour la communication connexion-orienté*

Disconnect (connection identifier), *pour la communication connexion-orienté*

EXEMPLE: COMMUNICATION INTERPROCESSUS HTTP



PASSAGE DE MESSAGE

Un processus envoie un message à un autre et continue son exécution interne

- Le message peut prendre du temps afin d'atteindre l'autre processus

Le message peut être stocké dans la file de données du processus de destination

- Si ce dernier n'est pas immédiatement prêt à recevoir le message

PASSAGE DE MESSAGE

Le message est reçu par le processus de destination lorsque:

- Ce dernier atteint un point dans son exécution interne où il est prêt à recevoir des messages

Ce type de communication s'appelle: *le passage de message asynchrone* (parce que l'envoi et la réception ne se font pas en même temps)

PASSAGE DE MESSAGE ASYNCHRONE

uOttawa
L'Université canadienne
Canada's university

Les opérations d'envoi et de réception bloquantes:

- Un récepteur est bloqué s'il atteint un point où il est capable de recevoir des messages mais qu'il n'y a pas de messages en attente

- Un expéditeur est bloqué s'il n'y a pas d'espace dans la file de données entre l'expéditeur et le récepteur
    - Cependant, dans plusieurs cas, on s'attend à des files arbitrairement longues, ce qui veut dire que l'expéditeur ne sera presque jamais bloqué

PASSAGE DE MESSAGE ASYNCHRONE

uOttawa
L'Université canadienne
Canada's university

Les opérations d'envoi et de réception non-bloquantes:

- Les opérations d'envoi et de réception retournes immédiatement
  - Elles retournent une valeur de statut qui peut indiquer qu'aucun message est arrivé au récepteur
- Le récepteur peut tester si un message est en attente et il peut possiblement effectuer d'autres traitements
  - Il peut optionnellement être notifié par le système lorsqu'un message est reçu

PASSAGE DE MESSAGE SYNCHRONE

uOttawa
L'Université canadienne
Canada's university

On suppose que l'envoi et réception se passent en même temps

- Souvent, on n'a pas besoin d'un tampon intermédiaire

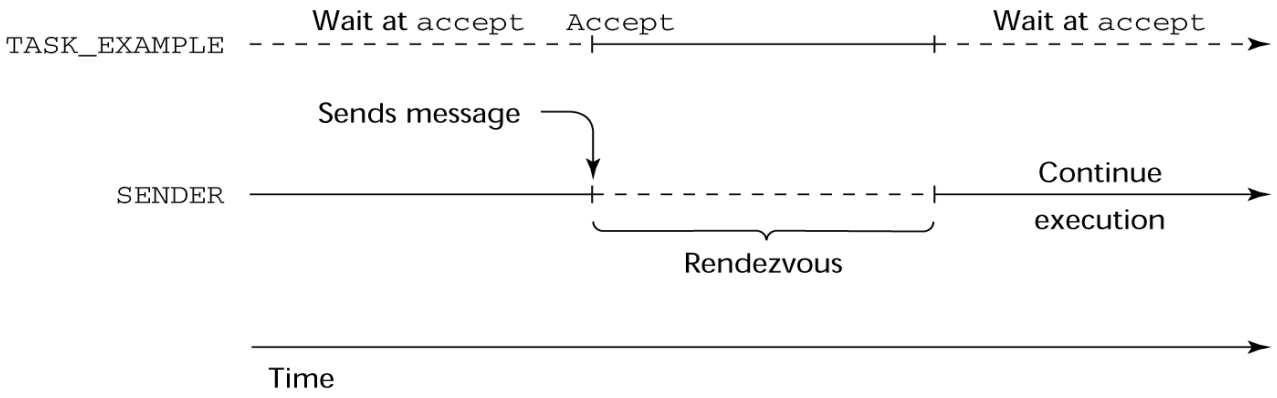
Ceci est aussi appelé un rendez-vous et implique une synchronisation plus proche:

- Les opérations d'envoi et de réception peuvent seulement se passer si les deux partis sont prêts

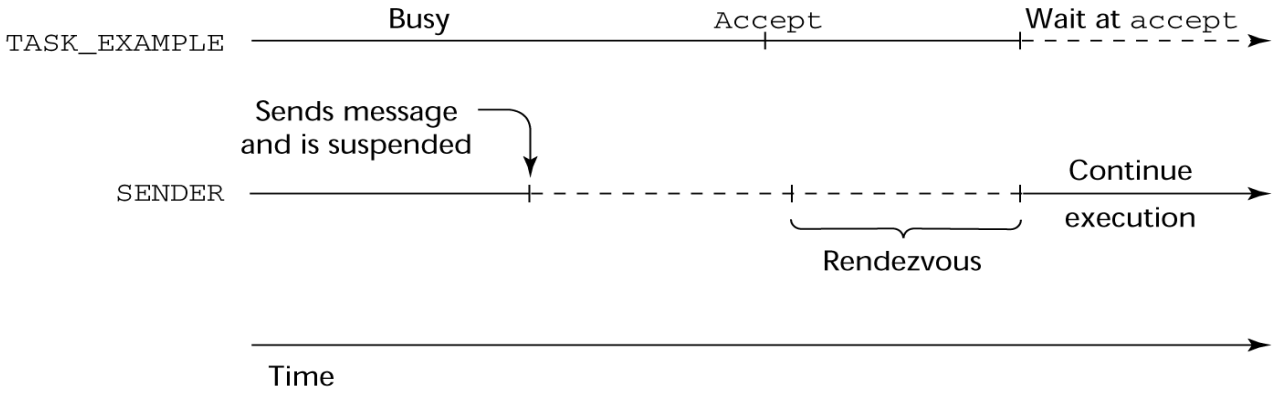
L'expéditeur doit attendre le récepteur ou le récepteur doit attendre l'expéditeur

RENDEZ-VOUS

(PAS AUSSI ROMANTIQUE QU'IL PARAIT!)



(a) TASK_EXAMPLE waits for SENDER



(b) SENDER waits for TASK_EXAMPLE

BLOCAGE INDÉFINI ET TIMEOUT

uOttawa
L'Université canadienne
Canada's university

Les opérations de connexion et de réception peuvent aboutir à un blocage indéfini

- Ex. une requête de connexion bloquante peut causer le processus qui a initier la requête d'être suspendu indéfiniment si la connexion ne peut être réaliser

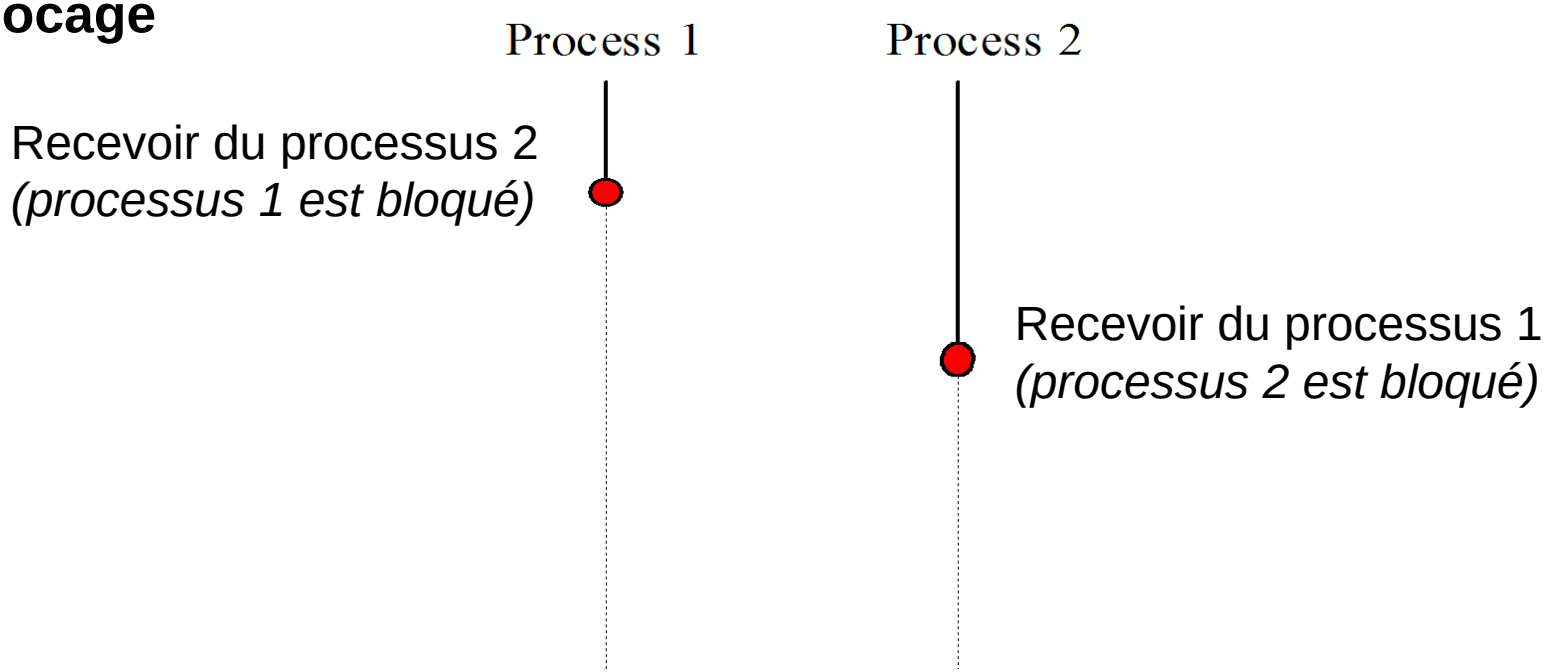
Le blocage indéfini peut être évité en utilisant un timeout

Le blocage indéfini peut aussi être causé par un inter-blocage

INTER-BLOCAGE

Les opérations de blocage qui sont lancées dans la mauvaise séquence peuvent aboutir à un *inter-blocage*

Un *timeout* peut être utilisé pour détecter/ résoudre un inter-blocage

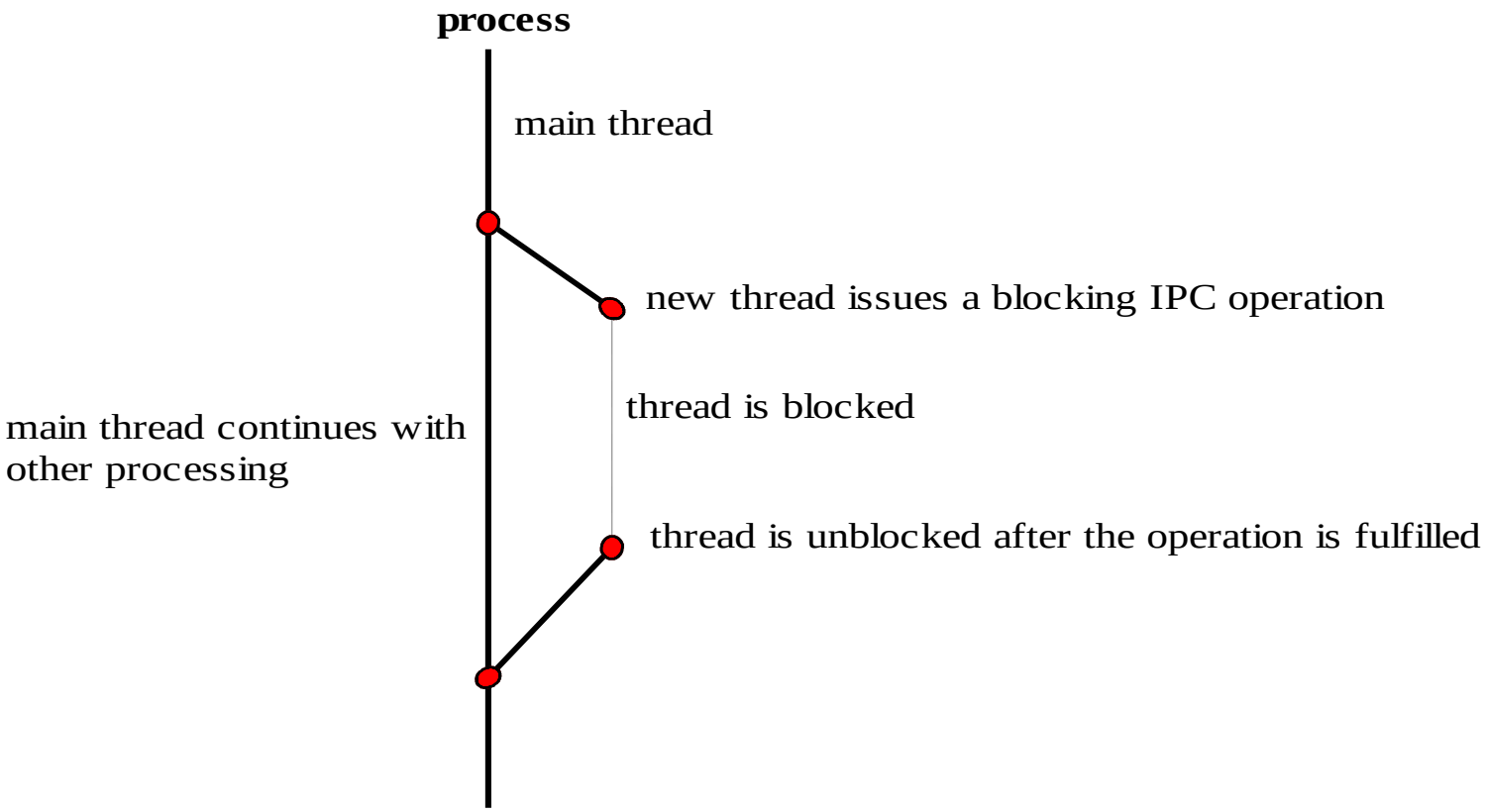


P1 attend les données de P2; P2 attend les données de P1

UTILISER UN THREAD POUR LES OPÉRATIONS BLOQUANTES

uOttawa
L'Université canadienne
Canada's university

Afin d'éviter de bloquer le processus entier sur une opération CIP bloquante, on peut créer un nouveau Thread



REPRÉSENTATION DES DONNÉES

uOttawa
L'Université canadienne
Canada's university

Les données sont transmises sur un réseau dans un flux binaire

Les ordinateurs peuvent avoir des formats de stockage interne différents pour le même type de données

- Une représentation externe des données est nécessaire
- Ceci est connu comme format standard

***La conversion de données (data marshalling) est le processus de* (i) aplatir une structure de données, et (ii) convertir les données en une représentation externe**

Quelques schémas de représentations externes de données bien connus sont:

- External Data Representation (XDR)
- ASN.1 (Abstract Syntax Notation One)
- XML (Extensible Markup Language)
- JSON (JavaScript Object Notation)

ÉCHANTILLON DE FICHIER XML

uOttawa
L'Université canadienne
Canada's university

XML est un langage de balisage basé sur le texte

- HTML n'est PAS un sous-ensemble de XML

Exemple:

```
<message>
        <to>you@yourAddress.com</to>
        <from>me@myAddress.com</from>
        <subject>XML Is Really Cool</subject>
        <text> How many ways is XML cool? Let me
        count the ways... </text>
 </message>
```

JSON VS XML

JSON

```
{
  "movies" : [
    {
      "movie": {
        "id": 1,
        "title": "Big and Badder"
      },
      "movie": {
        "id": 2,
        "title": "The Dot"
      },
      …
      "movie": {
        "id": 4,
        "title": "Invasion of the Dots "
      }
    }
  ]
}
```

XML format

```
<movies>
  <movie>
    <id>1</id>
    <title>Big and Badder</title>
  </movie>
  <movie>
    <id>2</id>
    <title>The Dot</title>
  </movie>
…
 <movie>
    <id>4</id>
    <title>Invasion of the Dots </title>
  </movie>
 </movies>
```

PROTOCOLES BASÉS SUR TEXTE

uOttawa
L'Université canadienne
Canada's university

La conversion de données est le plus simple lorsque les données échangées sont un flux de caractères, ou un texte

- Ce genre de protocole est connu comme étant basé sur texte
- Les données peuvent être facilement analysées dans un programme et affichées afin d'être examinées par humains

Plusieurs protocoles de réseaux populaires sont basés sur texte, ex.:

- FTP (File Transfer Protocol)
- HTTP
- SMTP (Simple Mail Transfer Protocol)

PROTOCOLE

Dans une application distribuée, deux processus effectuent la CIP selon un protocole accepté mutuellement

La spécification d'un protocole doit inclure:

- La séquence données échanger, laquelle peut être décrite en utilisant une charte de séquence de message (MSC)
- Le format de données échanger à chaque étape

Nous allons brièvement explorer un protocole simple basé sur texte: HTTP

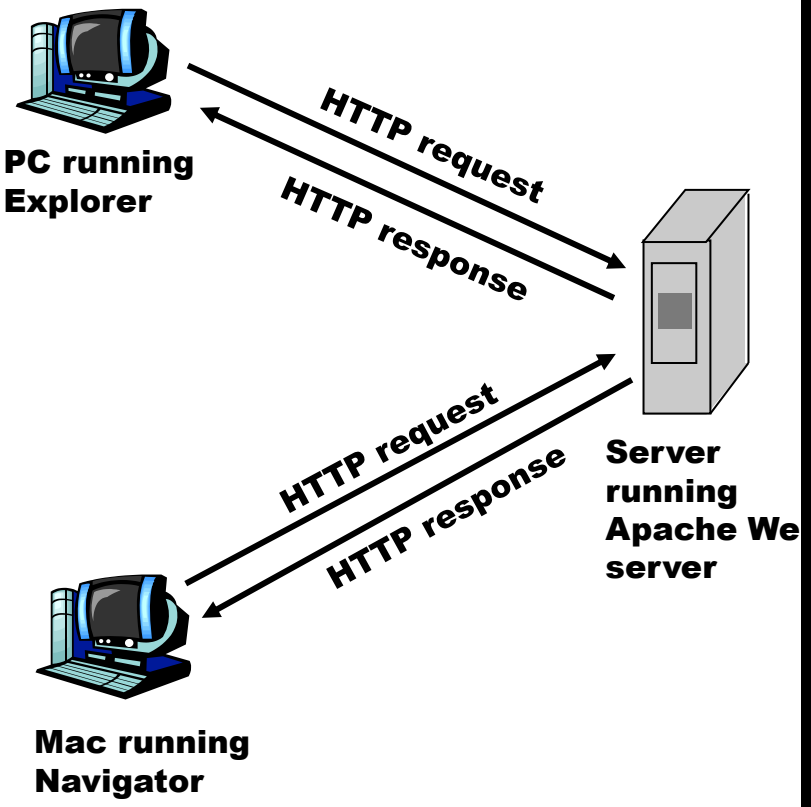
BASES DU HTTP

Modèle client/serveur

- *client:* navigateur qui demande, reçoit, ”affiche” des objets du Web
- *serveur:* serveur web qui envoie des objets en réponses aux demandes

HTTP

- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068
- HTTP/2: RFC 7540 *(publié Mai 2015)*



BASES DU HTTP

HTTP est un protocole de réseau utilisé pour livrer des *ressources* sur l'Internet

- Les ressources peuvent être des fichiers HTML, des fichiers d'image, des résultats de questions…

Une ressource est une pièce d'information qui peut être identifiée par un URL

- Le type de ressource le plus commun est un fichier

Une ressource peut être dynamiquement générée (comme étant un résultat à une question)

BASES DU HTTP

Les applications qui utilisent le protocole HTTP envoies leur données sur un flux bidirectionnel d'octets (bytes)

- Presque toujours TCP

Interaction

- Le client envoie une demande au serveur, suivi par une réponse du serveur au client
- Les demandes/réponses sont encodées en texte

Sans état

- Le serveur ne maintient pas d'information à propos des demandes précédentes des clients

REQUÊTES ET RÉPONSES HTTP

uOttawa
L'Université canadienne
Canada's university

Les format des messages de requête et de réponse sont similaires

Les deux genres de messages consistent de:

- *La première ligne spécifiant la nature du message.*
- *zéro ou plus de lignes d'Entêtes (Header),*
- *Une ligne vide (c.à.d. un CRLF tout seul), et*
- *Un corps de message optionnel (ex. fichier, ou donnée d'une question, ou sortie d'une question).*

Format d'un msg. HTTP:

```
<first line, different for request vs. response>
Header1: value1
Header2: value2
Header3: value3
<optional message body goes here, like file contents or query data;
it can be many lines long, or even binary data $&*%@!^$@>
```

PREMIÈRE LIGNE: REQUÊTE HTTP

uOttawa
L'Université canadienne
Canada's university

Elle contient trois parties séparées par des espaces:

- Nom de la **méthode**
- **Le chemin d'accé local (local path)** de la ressource demandée
- **La version** du protocole HTTP utilisée

Exemple: `GET /path/to/file/index.html HTTP/1.1`

GET est la méthode HTTP la plus commune;

- Elle dit « donne-moi cette ressource"

Le chemin d'accés est la partie de l'URL qui vient après le nom d'hôte (host name)

La version prend toujours le format "HTTP/x", majuscule

PREMIÈRE LIGNE: REQUÊTE HTTP

uOttawa
L'Université canadienne
Canada's university

Elles s'appelle *la ligne du statut*, elle possède aussi trois parties séparées par des espaces:

- **La version** du protocole HTTP utilisée
- **Le code d'état** de la réponse
- **Une phrase** de raison qui décrit le code d'état

Exemples:

- `HTTP/1.1 200 OK`
- `HTTP/1.1 404 Not Found`

Quelques codes du statut populaires:

- **200:** La requête a réussi, et la ressource résultante (ex. fichier ou sortie de script) est retournée dans le corps du message
- **404:** La ressource demandée n'existe pas
- **301:** Déplacé en permanence
- **302:** Déplacé temporairement

LIGNES D'ENTÊTE

uOttawa
L'Université canadienne
Canada's university

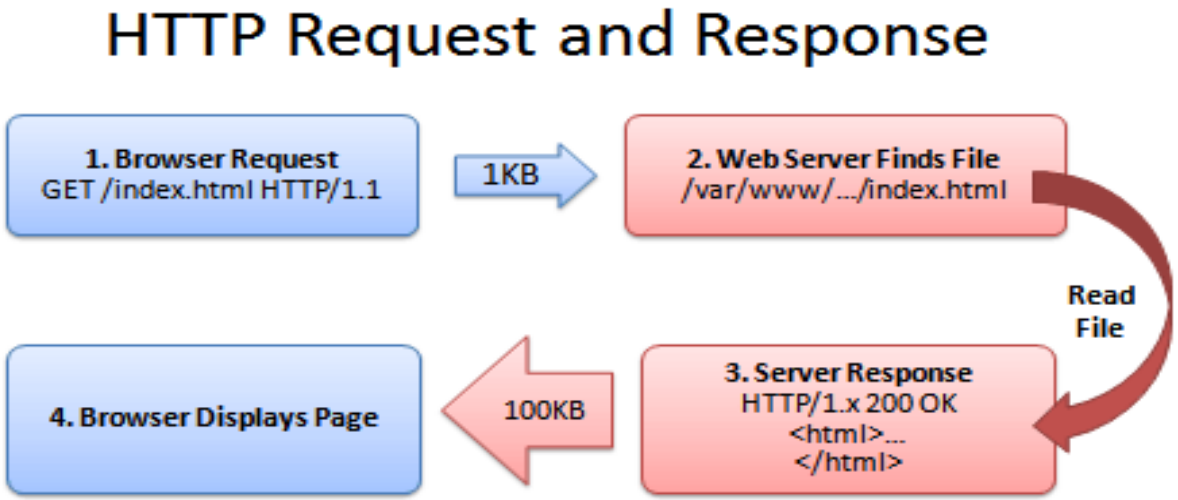
Les lignes d'entête fournissent de l'information à propos de la requête ou de la réponse, ou à propos de l'objet envoyé dans le corps du message

- Ils ne sont pas sensibles à la casse *(case sensitive)*
- HTTP 1.1 définit 46 lignes d'entête (header lines)

Exemples:

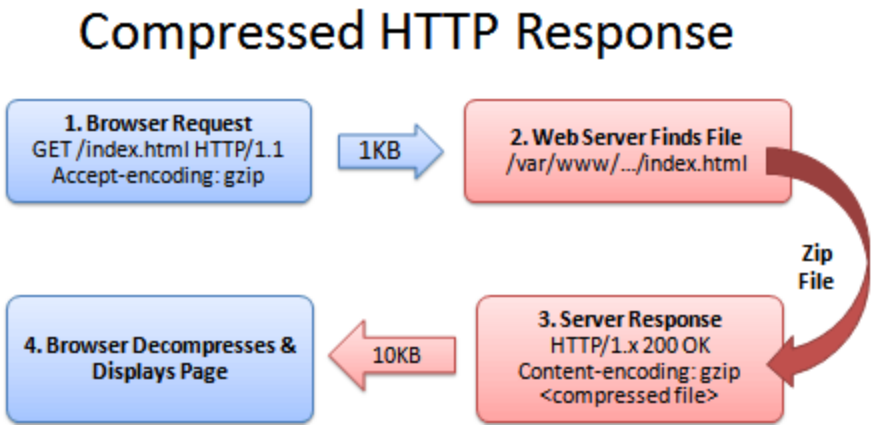
- **Host:** net.tutsplus.com
- **User-agent: Mozilla/3.0Gold**
- **Accept:** text/plain; q=0.5, text/html, text/x-dvi; q=0.8, text/x-c
- **Accept-Language:** en-us,en;q=0.5
- **Accept-Encoding:** gzip,deflate
- **Accept-Charset:** ISO-8859-1,utf-8;q=0.7,*;q=0.7

HTTP SANS COMPRESSION



1. Navigateur: Salut, GET me /index.html
2. Serveur: Ok, laisse-moi voir si index.html traîne aux alentours…
3. Serveur: Je l'ai trouvé! Voici ton code de réponse (200 OK) et je t'envoie un fichier.
4. Navigateur: 100KB? Ouch… j'attend, j'attend… ok, c'est chargé.

HTTP AVEC COMPRESSION



1. Navigateur: Salut, GET index.html? Je peux prendre une version compressée si tu l'as
2. Serveur: Laisse-moi trouver le fichier… oui, c'est ici. Et tu accepte une version compressée? C'est formidable
3. Serveur: Ok, j'ai trouvé index.html (200 OK), je le compresse et je te l'envoie
4. Navigateur: Parfait! C'est seulement 10KB. Je le décompresse et je le montre à l'utilisateur

CORPS DU MESSAGE

Dans une réponse, le corps de message contient:

- La ressource demandée, ou
- Un texte explicatif s'il y a une erreur

Dans une requête, le corps de message contient:

- Les données entrées par l'utilisateur ou les fichiers téléchargés au serveur (uploaded to the server)

AUTRES MÉTHODES HTTP: HEAD

uOttawa
L'Université canadienne
Canada's university

Une requête HEAD est comme une requête GET, à l'exception qu'elle demande au serveur de retourner seulement les lignes d'entête

- Pas de corps de message
- Seules, la première ligne et les lignes d'entête sont retournées

AUTRES MÉTHODES HTTP: POST

uOttawa
L'Université canadienne
Canada's university

Les requêtes POST sont utilisées pour envoyer les données au serveur afin d'être traitées

Une requête POST diffère d'une requête GET par les points suivants:

- Il y a un bloc de données envoyé avec la requête, dans le corps du message
- Il y a généralement des entêtes additionnels pour décrire ce corps du message, comme **Content-Type:** et **Content-Length:**.
- La *requête URL* n'est pas une ressource à récupérer; c'est généralement un programme qui traite les données qu'on envoie
- La réponse HTTP est normalement une sortie de programmes, et non pas un fichier statique.

PASSAGE DE PARAMÈTRES

uOttawa
L'Université canadienne
Canada's university

Dans une requête GET, les paramètres sont passés dans l'URL:

```
GET /path/to/file/stuff.do?param1=value1&param2=value2 HTTP/1.1
```

Dans une requête POST, les paramètres sont passés dans le corps de message:

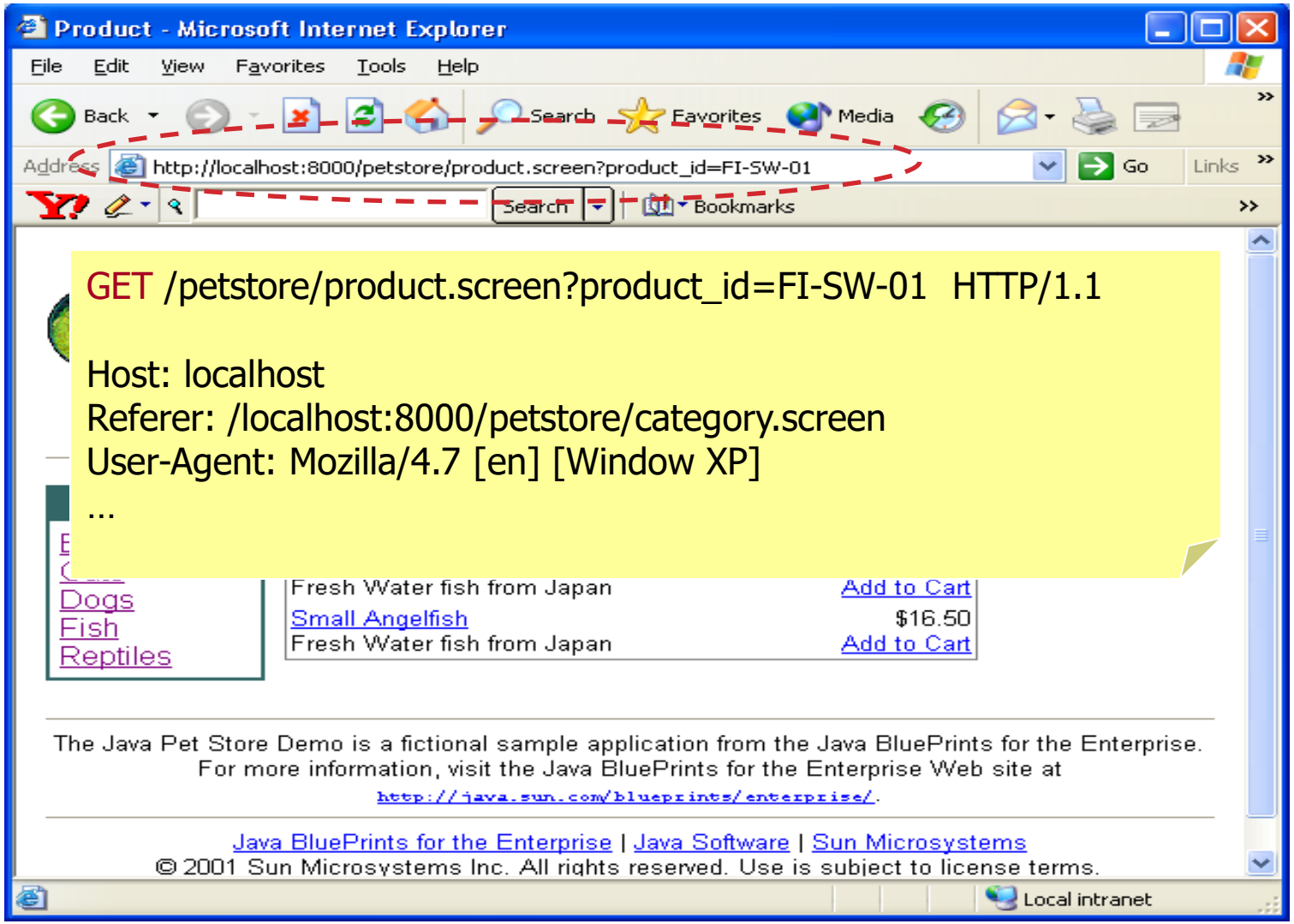
```
POST /petstore/cart.do HTTP/1.1

Host: localhost
Referer: /localhost:8000/petstore/cart.do
User-Agent: Mozilla/4.7 [en] [Window XP]
Content-Type: application/x-www-form-urlencoded
Content-Length: xxxx

action=update&itemQuantity_EST-14=1
```

CLIENT NAVIGATEUR QUI ENVOIE UNE REQUÊTE HTTP GET

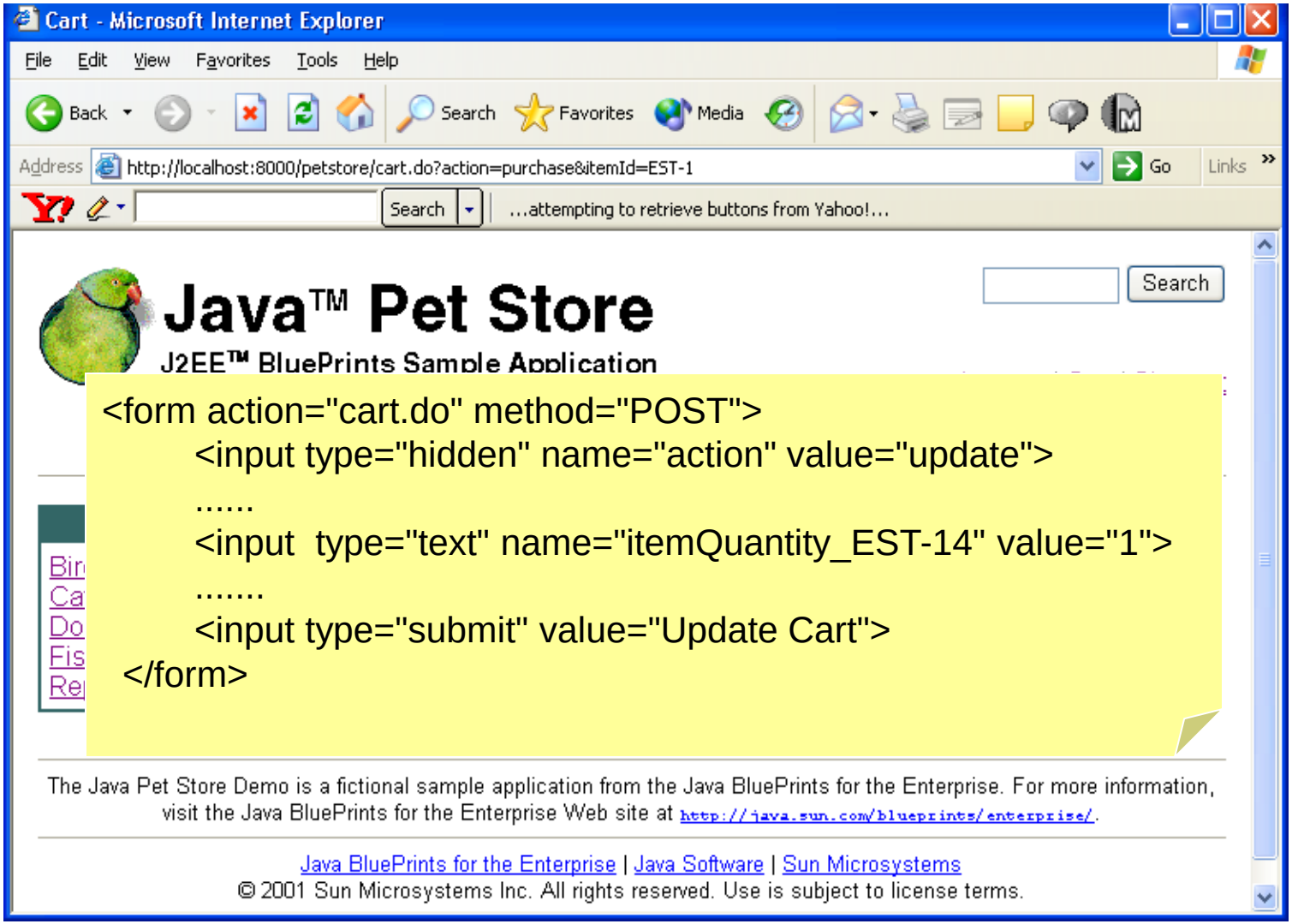
User requests

retrieve data
Map to

HTTP Get



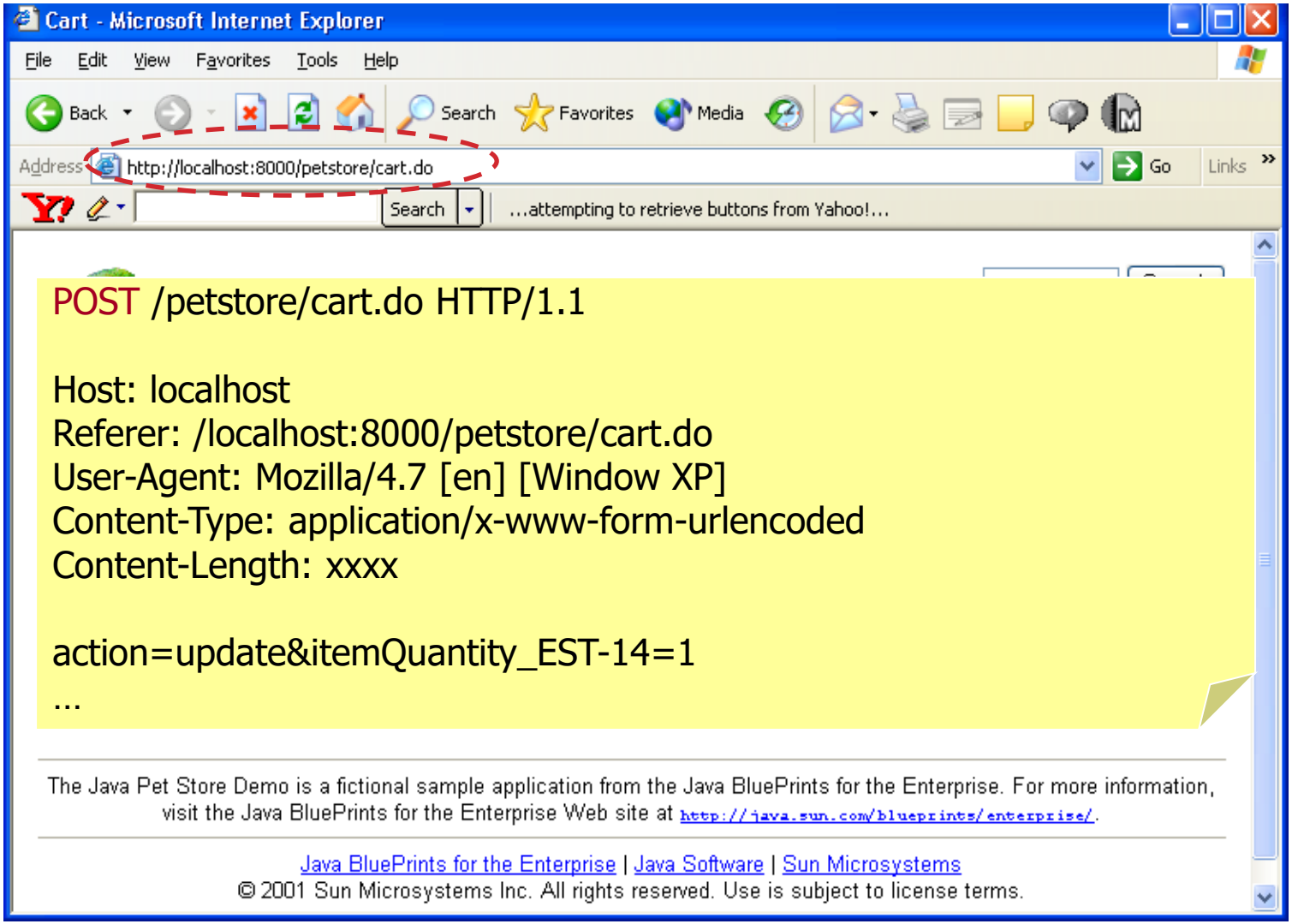
CLIENT NAVIGATEUR QUI ENVOIE UNE REQUÊTE HTTP POST



User requests

update data
Map to

HTTP Post



MERCI!

QUESTIONS?