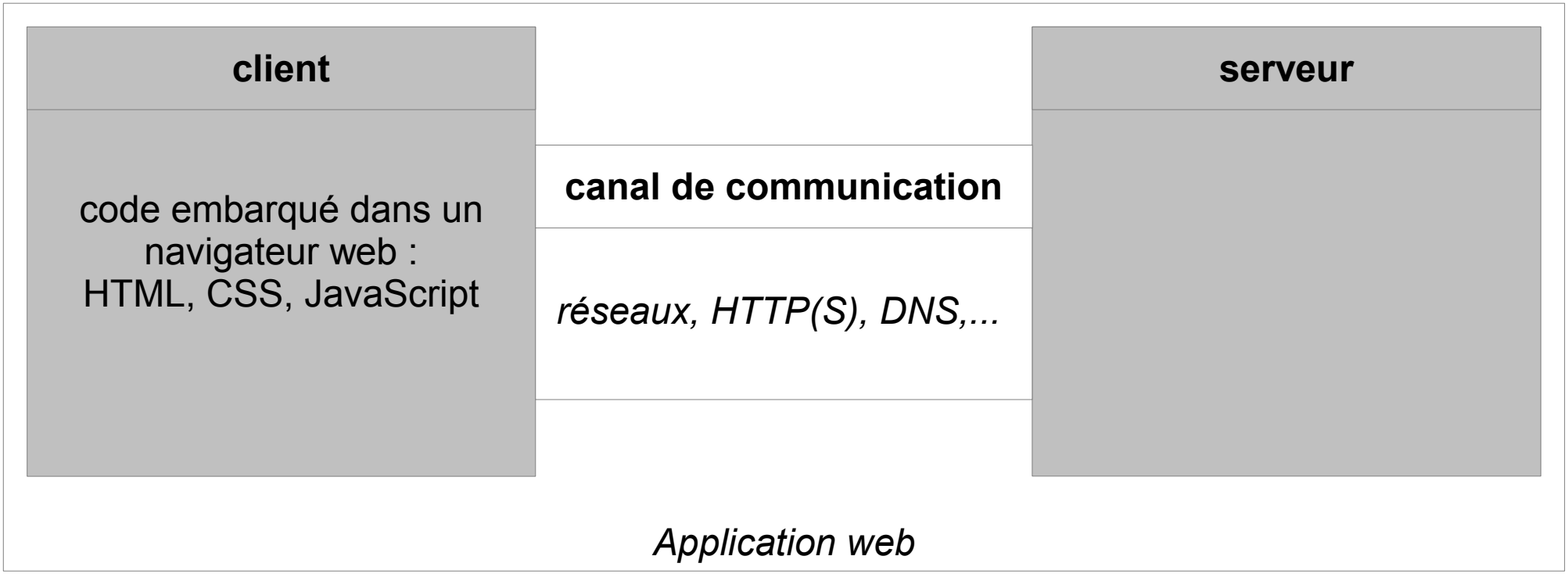
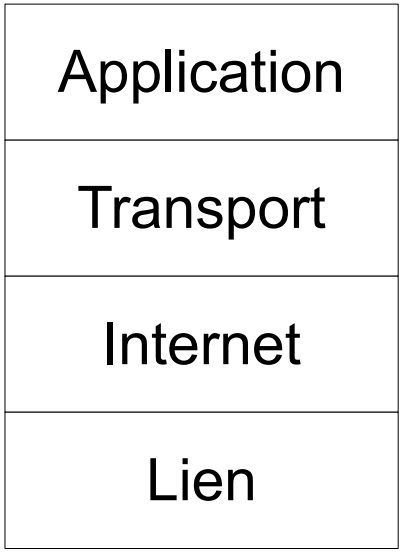
HTTP – Introduction



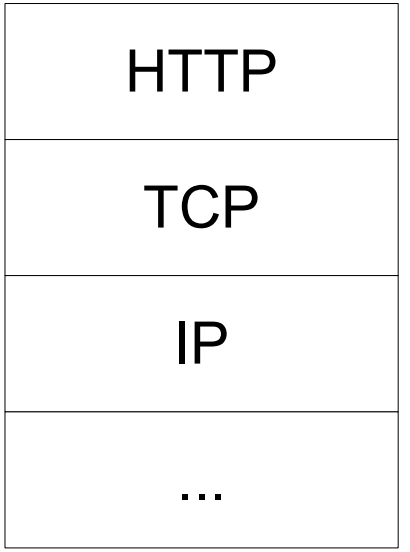
- HyperText Transfer Protocol (HTTP) est le second pilier du web (le premier est le HTML)
- HTTP = protocole réseau asymétrique, basé sur TCP/IP, de niveau Application

HTTP – Rappel TCP/IP

- La suite de protocoles TCP/IP constitue une pile où chaque protocole « supérieur » fournit une abstraction supplémentaire des protocoles « inférieurs »
- Habituellement, on classe les protocoles en 4 niveaux allant de Lien physique (le plus bas) à Application (le plus haut)



- Les *applications web* sont concernées par les protocoles IP (couche Internet), TCP (couche Transport) et HTTP (couche Application)



HTTP – Rappel TCP/IP – Couche Lien

- La couche Lien prend en charge les communications du réseau « local », entre deux équipements « adjacents »
- De nombreux protocoles existent à ce niveau, selon le type de média physique utilisé pour la communication (câble cuivre, fibre optique, ondes radio,...) et selon les caractéristiques des équipements qui communiquent (3G ou 4G ?, Ethernet ou GigaEthernet ?,...)
- Les équipements réseau sont identifiés par leur adresse MAC (00-FF-DC-70-54-CE) et s'échangent des trames de données
- Les protocoles de cette couche généralement intègrent des mécanismes de redondance et de signature, pour détecter et/ou corriger des erreurs de transmission
- Les applications web ignorent ce qui se passe dans Lien

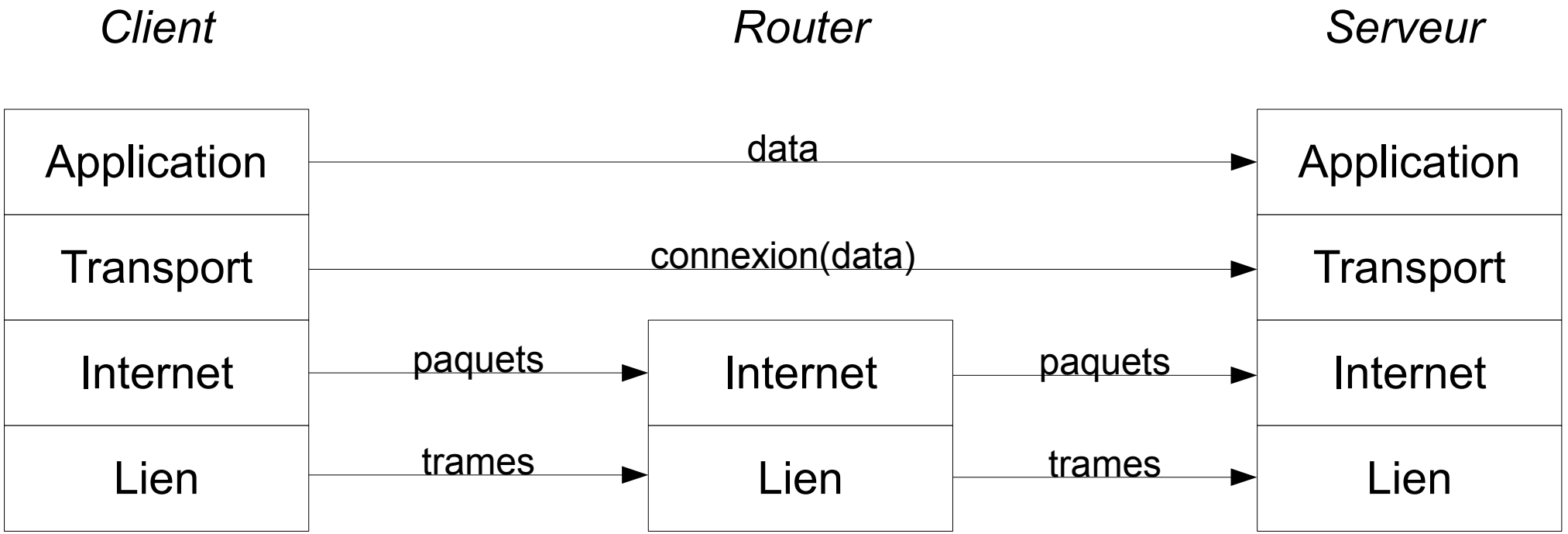
HTTP – Rappel TCP/IP – Protocole IP

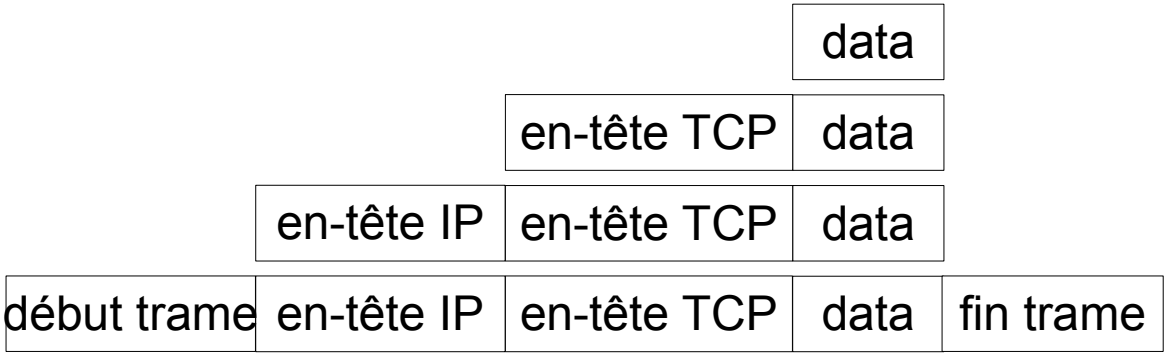
- IP (Internet Protocol) fournit une abstraction de la couche Lien : il propose une gestion uniforme d'échange de données, indépendamment de la réalité physique
- Identification des noeuds dans le réseau par adresse IP : soit IPv4 (195.75.83.11), soit IPv6 (2001:0db8:0000:85a3:0000:0000:ac1f:8001)
- Découpage de données en paquets (de taille variable)
- Service fourni = best-effort : les paquets peuvent se perdre en route et peuvent arriver dans le désordre
- IP définit un mécanisme de routage dynamique et décentralisé des paquets
- Intérêt dans les applications web : éventuellement l'adresse IP pour des contrôles de sécurité

HTTP – Rappel TCP/IP – Protocole TCP

- TCP (Transmission Control Protocol) fournit une abstraction de la couche Internet : il propose la communication connectée de bout à bout
- Identification des noeuds dans le réseau par le numéro de port (p.ex. 80 ou 443) [en plus de l'adresse IP]
- Service fourni = livraison garantie : les paquets perdus sont réémis, et remis dans l'ordre à la destination ; TCP délivre des ensembles de données entre deux points quelconques de l'Internet
- Intérêt dans les applications web : notion de communication connectée entre deux points, initiée et pilotée par l'un des points (client-serveur, situation asymétrique)

HTTP – Rappel TCP/IP – Exemple





HTTP – Abstraction de Transmission

- HTTP fournit une abstraction de la couche Transmission : il propose l'appel d'opérations à distance (avec données en entrée et en sortie) entre une application serveur et ses clients. Le modèle proposé est sans état.
- Identification des opérations par leur chemin (p.ex. /accueil)

  [en plus de l'adresse IP et du port]
- Service fourni = requête/réponse : le client envoie des requêtes, le serveur fournit les réponses (éventuellement dans le désordre)
- Intérêt dans les applications web : c'est le fondement des communications sur le web.

HTTP – Requête

- Une requête HTTP (envoyée par le client au serveur) contient :
  - le **chemin** de l'opération visée (/index.html ou /listes/voitures/recherche?marque=Volvo)
  - la **méthode** de requêtage : GET, POST, HEAD, PUT,...
  - le **corps** de la requête (=les données en entrée), éventuellement vide
  - les **en-têtes** associées : p.ex. Accept, Authorization, Cache-Control, Cookie, Content-Length, Content-Type, If-Modified-Since,...

HTTP – Réponse

- Une réponse HTTP (envoyée le serveur au client en réponse à une requête) contient :
  - le **corps** de la réponse (=les données en sortie), éventuellement vide
  - les **en-têtes** associées : p.ex. Status (200, 404, 500,...), Content-Length, Content-Type, Location, Set-Cookie, WWW-Authenticate,...

HTTP – Approche REST (1)

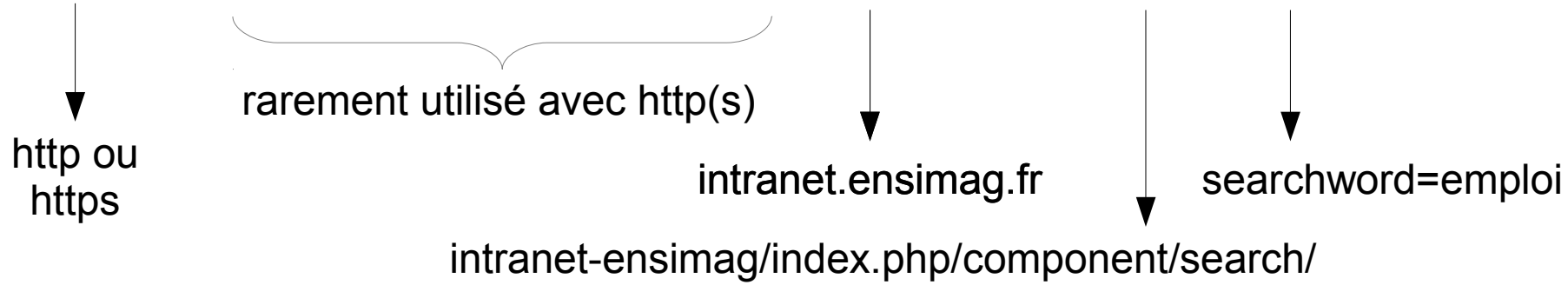
- REST propose une certaine vue abstraite du serveur et des opérations offertes. Les applications web qui suivent cette approche ont par construction certaines bonnes propriétés.
- Le serveur contient des données : ressources. Chaque ressource est désignée par son adresse unique (chemin).
- Selon le cas, le client dispose d'actions sur les ressources :
  - GET : obtenir une représentation de la ressource
  - HEAD : obtenir les méta-données de la ressource
  - PUT : modifier la ressource [existante] en fournissant la représentation du nouvel état
  - DELETE : supprimer la ressource
  - POST : créer une nouvelle ressource en fournissant son contenu, et en obtenant son adresse en retour
- Le serveur utilise toute le plage de codes de retour

HTTP – Approche REST (2)

- Avantages des méthodes (GET, POST,...) en REST :
  - GET est sûre (on peut l'exécuter sans crainte de modifier les données sur le serveur)
  - GET, PUT et DELETE sont idempotentes (exécuter une fois ou plusieurs fois la même requête revient au même)
  - seule POST n'est ni sûre ni idempotente
- Problème des développeurs web : les navigateurs ne fournissent que GET et POST
- Avantages globaux :
  - le mécanisme REST ne s'intéresse pas à l'état du client
  - accès concurrents résolus de manière naturelle (premier arrivé-premier servi, dernier mot l'importe)

URLs et DNS

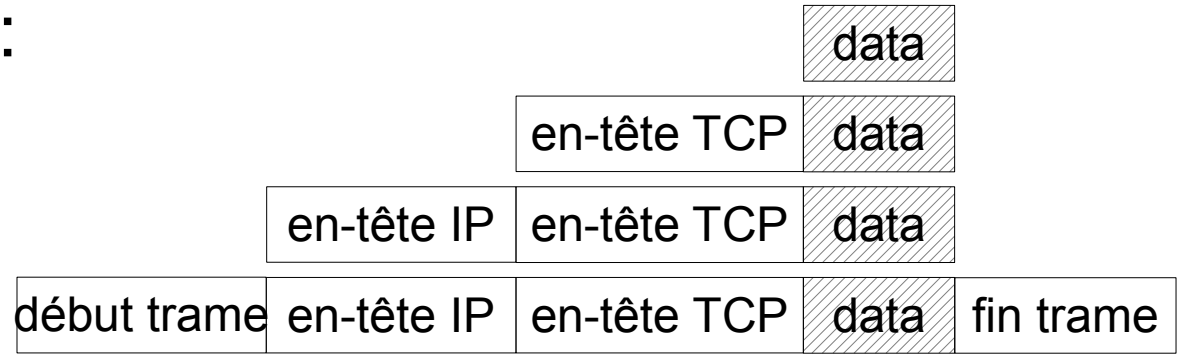
- Les adresses des ressources sur le web sont des URLs (Uniform Resource Locators) :
  scheme://username:password@domain:port/path?query#fragment



- */path?query#fragment* fait partie de la requête HTTP
- *scheme* indique le protocole à utiliser (et le port par défaut, si non spécifié)
- *port* fait partie de la connexion niveau TCP
- *domain* est d'abord traduit en adresse IP grâce au dispositif DNS (Domain Name System)

HTTPS (1)

- HTTPS signifie « HTTP over TLS » (Transport Layer Security). C'est donc le protocole HTTP « sécurisé » (par chiffrement) au niveau de la couche Transport.
- Dans HTTPS, on échange par TCP des données HTTP cryptées :



  l'adresse IP et le n°de port restent en clair
- Le mécanisme de cryptage est considéré comme sûr (garantissant la confidentialité), la longueur des clés peut être adaptée aux exigences de sécurité.
- Mais l'échange de clés est vulnérable à l'attaque « man-in-the-middle » : nécessité de signature par tiers de confiance.

HTTPS (2)

- Si le certificat du serveur (=sa clé publique) est signé par une autorité indépendante (et on peut remonter jusqu'à une autorité-racine), alors l'identité du serveur est acquise.
  → HTTPS garantit alors la confidentialité, l'intégrité et l'origine des données
- Une application web doit être entièrement sur HTTPS, sinon elle reste vulnérable à certaines attaques :
  - servir des ressources statiques (images, scripts) par HTTP ouvre la porte aux injections de code
  - mettre l'authentification en HTTPS et le reste en HTTP permet le vol de session