Initiation aux réseaux pour débutants

Filière Informatique de Développement d'Applications (IDA)

Cours préparé pour les étudiants débutants

Date : 3 juillet 2025

Auteur : Grok, créé par xAI

1 Introduction

Les réseaux informatiques sont au cur des applications modernes. Que vous développiez une application web, mobile ou desktop, comprendre les réseaux est essentiel pour concevoir des systèmes fiables, sécurisés et performants. Ce cours vise à fournir une base solide sur les réseaux, leurs principes, et leur lien avec le développement d'applications.

1.1 Objectifs pédagogiques

- Comprendre ce qu'est un réseau informatique et son rôle dans le développement d'applications.
- Maîtriser les concepts de base : protocoles, adressage, modèles, et équipements.
- Découvrir comment les réseaux influencent la conception et le fonctionnement des applications.
- Appliquer ces connaissances à des scénarios simples de développement.

1.2 Durée estimée

3 heures (théorie + exemples pratiques).

2 Qu'est-ce qu'un réseau informatique ?

Un réseau informatique est un ensemble d'appareils (ordinateurs, smartphones, serveurs, etc.) interconnectés pour échanger des données. Ces appareils communiquent via des protocoles, des câbles, ou des technologies sans fil (Wi-Fi, 4G/5G).

2.1 Pourquoi est-ce important pour les développeurs ?

- Les applications modernes (web, mobile, cloud) dépendent des réseaux pour fonctionner.
- Exemples : une application mobile qui envoie des données à un serveur, un site web qui charge des ressources via HTTP, ou une API qui communique entre services.
- Comprendre les réseaux aide à optimiser la performance, sécuriser les échanges et diagnostiquer les problèmes.

2.2 Exemple concret

Quand vous utilisez une application comme WhatsApp, votre message est envoyé via Internet à un serveur, qui le transmet au destinataire. Ce processus repose sur des concepts de réseau que nous allons explorer.

3 Concepts de base des réseaux

3.1 Les composants dun réseau

- **Nuds (Nodes) :** Les appareils connectés (PC, serveurs, smartphones, etc.).
- **Médias de transmission :** Les supports physiques ou sans fil (câbles Ethernet, fibre optique, Wi-Fi).
- **Équipements réseau :**

  - **Routeur :** Dirige les données entre différents réseaux (ex. : connexion Internet).
  - **Switch :** Connecte plusieurs appareils dans un même réseau local (LAN).
  - **Point d'accès Wi-Fi :** Permet la connexion sans fil.
- **Serveurs :** Fournissent des services ou des ressources (ex. : serveur web, serveur de base de données).

3.2 Types de réseaux

- **LAN (Local Area Network) :** Réseau local, comme dans une maison ou un bureau.
- **WAN (Wide Area Network) :** Réseau étendu, comme Internet.
- **MAN (Metropolitan Area Network) :** Réseau à léchelle dune ville.
- **PAN (Personal Area Network) :** Réseau personnel (ex. : Bluetooth).

3.3 Protocoles : Le langage des réseaux

Un protocole est un ensemble de règles permettant la communication entre appareils. Les principaux protocoles pour les développeurs incluent :

- **HTTP/HTTPS :** Pour les sites web et les API.
- **TCP/IP :** La base de la communication sur Internet.
- **DNS :** Traduit les noms de domaine (ex. : google.com) en adresses IP.
- **FTP :** Pour le transfert de fichiers.
- **WebSocket :** Pour des communications en temps réel (ex. : chat).

3.4 Adressage réseau

Chaque appareil sur un réseau est identifié par une **adresse IP** (Internet Protocol).

- **IPv4 :** Format classique (ex. : 192.168.1.1).
- **IPv6 :** Nouveau format pour plus dadresses (ex. : 2001 :0db8 ::1).
- **Masque de sous-réseau :** Définit la partie du réseau (ex. : 255.255.255.0).
- **Passerelle (Gateway) :** Point de sortie vers un autre réseau (souvent le routeur).

3.5 Ports

Les ports sont des "portes" logicielles sur un appareil, utilisées pour identifier une application ou un service spécifique.

- Exemples : Port 80 (HTTP), 443 (HTTPS), 22 (SSH).
- Une application peut écouter sur un port spécifique (ex. : serveur web sur le port 8080).

4 Le modèle OSI et TCP/IP

4.1 Le modèle OSI

Le modèle OSI (Open Systems Interconnection) divise la communication réseau en 7 couches :

1. **Physique :** Transmission brute des données (câbles, Wi-Fi).
2. **Liaison de données :** Gestion des erreurs et adressage local (ex. : Ethernet).
3. **Réseau :** Routage des données (ex. : IP).
4. **Transport :** Fiabilité de la transmission (ex. : TCP, UDP).

5. **Session :** Gestion des connexions entre applications.
6. **Présentation :** Formatage des données (ex. : chiffrement SSL/TLS).
7. **Application :** Interface avec lutilisateur (ex. : HTTP, FTP).

4.2 Le modèle TCP/IP

Plus simple, il regroupe les couches OSI en 4 :

1. **Accès réseau :** Couches physique et liaison.
2. **Internet :** Couche réseau (IP).
3. **Transport :** Couche transport (TCP/UDP).
4. **Application :** Couches session, présentation, application.

Pourquoi cest important ? En tant que développeur, vous interagissez principalement avec la couche **application** (ex. : création dune API REST avec HTTP) et parfois la couche **transport** (ex. : choix entre TCP et UDP pour une application temps réel).

5 Réseaux et développement dapplications

5.1 Comment les réseaux impactent les applications ?

— **Latence :** Le temps que met une donnée à voyager (important pour les applications temps réel comme les jeux ou le streaming).
— **Bande passante :** La quantité de données transférables par seconde.
— **Sécurité :** Les données doivent être protégées (ex. : HTTPS, VPN).
— **Scalabilité :** Les applications doivent gérer plusieurs utilisateurs connectés via le réseau.

5.2 Exemple : Création dune application client-serveur

Imaginons que vous développez une application de chat :

— **Client :** Linterface utilisateur (ex. : une app mobile).
— **Serveur :** Gère les messages et les stocke.
— **Protocole :** WebSocket pour une communication en temps réel.
— **Port :** Votre serveur écoute sur un port (ex. : 3000).
— **Adresse IP :** Les clients se connectent à lIP du serveur.

5.3 Outils pour les développeurs

— **Postman :** Tester des API REST.
— **Wireshark :** Analyser le trafic réseau.
— **Ping/Traceroute :** Vérifier la connectivité.
— **cURL :** Envoyer des requêtes HTTP en ligne de commande.

6 Sécurité réseau pour les développeurs

6.1 Menaces courantes

— **Attaques par déni de service (DoS) :** Surcharge dun serveur.
— **Injection SQL :** Si une API nest pas sécurisée.
— **Interception (Man-in-the-Middle) :** Vol de données non chiffrées.

6.2 Bonnes pratiques

— Utiliser **HTTPS** pour chiffrer les communications.
— Valider et nettoyer les entrées utilisateur pour éviter les injections.
— Configurer des **pare-feu** pour limiter laccès aux ports inutiles.
— Mettre à jour régulièrement les bibliothèques et frameworks utilisés.

7 Exercices pratiques

7.1 Exercice 1 : Tester une connexion réseau

— Utilisez la commande `ping google.com` dans un terminal pour vérifier la connectivité.
— Analysez le temps de réponse (latence).

7.2 Exercice 2 : Créer une requête HTTP

— Utilisez Postman ou cURL pour envoyer une requête GET à une API publique (ex. : `https://jsonplaceholder.typicode.com/posts`).
— Observez la réponse JSON et identifiez les en-têtes HTTP.

7.3 Exercice 3 : Simuler une communication client-serveur

Écrivez un petit programme (en Python, par exemple) pour créer un serveur TCP simple qui écoute sur le port 12345 et un client qui envoie un message.

Listing 1 – Serveur TCP simple

```python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('127.0.0.1', 12345))
s.listen(1)
conn, addr = s.accept()
data = conn.recv(1024)
print("Message␣reçu␣:", data.decode())
conn.close()
```

Listing 2 – Client TCP simple

```python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('127.0.0.1', 12345))
s.send("Bonjour␣!".encode())
s.close()
```

8 Ressources pour aller plus loin

— **Livres :**
  — *Computer Networking : A Top-Down Approach* (Kurose & Ross).
  — *TCP/IP Illustrated* (W. Richard Stevens).
— **Sites :**

  - Cisco Networking Academy (cours gratuits).
  - MDN Web Docs pour HTTP et WebSockets.
- **Outils :**
  - Wireshark pour analyser le trafic.
  - Postman pour tester les API.

9 Conclusion

Ce cours vous a introduit aux bases des réseaux informatiques, en mettant laccent sur leur importance pour le développement dapplications. Vous avez appris les concepts clés (protocoles, adressage, modèles OSI/TCP-IP), les équipements réseau, et les implications pour la sécurité et la performance des applications. En tant que développeur IDA, ces connaissances vous aideront à concevoir des applications robustes et à interagir efficacement avec les infrastructures réseau.

Prochaines étapes :

- Approfondir les protocoles comme HTTP/REST pour le développement web.
- Explorer les bases de données distribuées et les architectures microservices.
- Expérimenter avec des outils comme Docker pour simuler des environnements réseau.