

**Objectifs** À l'issue de ce TP, vous serez capable de : câbler et configurer un petit réseau physique de Raspberry Pi, attribuer des adresses IP statiques, diagnostiquer la connectivité, administrer une machine à distance en SSH et transférer des fichiers en SCP, découper un réseau en sous-réseaux par VLSM, et mettre en place un routage statique entre deux sous-réseaux.

**Attention** Contexte de cette séance : la salle des Raspberry Pi n'étant pas toujours disponible, ce TP se déroule sur les PC de la salle habituelle, démarrés directement sur une clé USB flashée avec Ubuntu Desktop (x86\_64) — pas de VM, pas de multi-boot Ventoy, une clé = une image, flashée à l'avance pour plus de fiabilité. Partout où ce TP mentionne un « Raspberry Pi » ou un « Pi », lisez « poste démarré sur la clé USB ». Ce choix rejoint directement les projets dans lesquels vous utilisez déjà Ubuntu (version ARM) sur les Raspberry Pi — les commandes vues ici sont donc les mêmes que sur le matériel réel.

**Matériel nécessaire** Par poste : 1 clé USB3 (8 Go minimum) flashée à l'avance avec Ubuntu Desktop x86\_64 (une image par clé — voir Étape 0).

1 switch Ethernet non managé (5 ports minimum) et son câble d'alimentation.

Un câble RJ45 par poste, plus un câble RJ45 pour relier le poste de contrôle de l'enseignant au switch.

1 adaptateur USB-Ethernet par poste jouant le rôle de routeur (exercice 7 — routage entre deux sous-réseaux), si le PC n'a qu'une seule carte réseau filaire.

Accès au BIOS/UEFI de chaque poste (touche de boot menu, F12/F2/Échap selon le modèle) pour démarrer sur la clé USB et, si besoin, désactiver le Secure Boot.

En secours : une clé USB de rechange déjà flashée, pour remplacer rapidement une clé défectueuse sans tout re-flasher en pleine séance.

**Durée indicative** : 3h

- **Mode de travail** : seul, sur un mini-réseau isolé (sans Internet)
- **Pré-requis** : TP1 (commandes de base Linux).

## 1ère partie — Mise en réseau physique

### Étape 0 : Préparation des clés USB bootables

Cette étape est à faire par l'enseignant avant la séance (ou par les étudiants en tout début de séance si le temps le permet).

1. Téléchargez l'image Ubuntu Desktop (x86\_64) depuis le site officiel [ubuntu.com/download/desktop](https://ubuntu.com/download/desktop).
2. Flashez chaque clé USB3 avec balenaEtcher ou Rufus — un flashage direct de l'image, sans passer par un outil multi-boot, pour un démarrage plus fiable et reproductible d'un poste à l'autre.
3. Sur chaque poste, démarrez sur la clé USB (touche de boot menu au démarrage, souvent F12, F2 ou Échap selon le modèle) ; désactivez le Secure Boot dans le BIOS si le démarrage est refusé.
4. Une fois la session graphique lancée (utilisateur live ubuntu, sans mot de passe), définissez un mot de passe pour cet utilisateur (nécessaire avant d'activer SSH à l'étape 4) :

```
$ passwd
```

**Point de contrôle** Chaque poste doit démarrer sur sa clé et afficher le bureau, sans passer par le disque dur interne du PC. Notez sur une feuille quel poste utilise quelle clé, pour retrouver facilement une machine en cas de souci.

## Étape 1 : Configuration physique

5. Éteignez tous les postes.
6. Connectez chaque poste au switch via un câble Ethernet. Branchez le switch sur une alimentation si nécessaire.
7. Allumez les postes sur leur clé USB et attendez qu'ils démarrent complètement.
8. Vérifiez sur chaque poste que l'interface Ethernet est détectée :

```
$ ip link show
```

Vous devriez voir une interface comme eth0, enp2s0 ou enp3s0 (le nom exact dépend de la carte réseau du PC).

## Étape 2 : Configuration des adresses IP statiques

Puisqu'il n'y a pas de serveur DHCP (ni d'accès Internet) sur ce réseau isolé, les adresses IP sont attribuées manuellement, sur un réseau privé, par exemple 192.168.1.0/24.

Lubuntu est basé sur Ubuntu : le réseau y est géré par NetworkManager, comme sur la VM Lubuntu du TP1 — les commandes sont donc identiques à celles déjà vues.

9. Listez les connexions disponibles :

```
$ nmcli connection show
```

10. Configurez l'IP statique sur la connexion filaire (remplacez X par un numéro unique pour chaque poste, ex. 10 pour le premier, 11 pour le second) :

```
$ sudo nmcli connection modify "Wired connection 1" \
    ipv4.addresses 192.168.1.X/24 ipv4.method manual
$ sudo nmcli connection up "Wired connection 1"
```

Alternative en interface graphique : cliquez sur l'icône réseau en haut à droite → Modifier les connexions → onglet IPv4 → Manuel.

11. Vérifiez la configuration :

```
$ ip addr show eth0
```

**Point de contrôle** Vous devez voir votre IP apparaître (ex. 192.168.1.10) avec le suffixe /24. Si elle n'apparaît pas, relancez `sudo nmcli connection up`.

## Exercice 1

Attribuez des IP uniques à chaque Pi (par exemple 192.168.1.10 à 192.168.1.25). Notez-les sur une feuille pour référence.

**Attention** Problème courant : en cas de conflit d'IP, un Pi ne répondra pas sur le réseau. Vérifiez avec `ip addr` et corrigez si besoin.

## Étape 3 : Test de connectivité

12. Depuis un Pi (ex. 192.168.1.10), testez la connexion vers un autre (ex. 192.168.1.11) :

```
$ ping 192.168.1.11
```

- Si ça marche : vous voyez des réponses (ex. « 64 bytes from... »).

- Si ça échoue : vérifiez les câbles et les IP, ou utilisez `ping -c 4 192.168.1.11` pour limiter à 4 paquets et ne pas bloquer le terminal.

13. Testez depuis tous les Pi vers tous les autres.

## Exercice 2

Créez un tableau simple (sur papier ou dans un fichier texte) listant les IP et notez si le ping réussit entre chaque paire. Discutez en groupe des raisons d'un échec potentiel (mauvaise configuration IP, câble défectueux, Pi éteint...).

## Étape 4 : Configuration SSH pour connexions distantes

SSH permet de contrôler un poste depuis un autre, sans écran ni clavier branchés dessus.

14. Installez et activez le service SSH (l'image live ne l'active pas par défaut) :

```
$ sudo apt install openssh-server -y
$ sudo systemctl enable --now ssh
```

15. Depuis un poste, connectez-vous à un autre (remplacez par l'IP et le nom d'utilisateur cible) :

```
$ ssh utilisateur@192.168.1.11
```

**Astuce** Le mot de passe demandé est celui que vous avez défini à l'Étape 0 avec `passwd`, pour l'utilisateur par défaut de l'image live.

16. Une fois connecté, tapez `whoami` pour vérifier votre identité, puis `exit` pour quitter.

## Exercice 3

Connectez-vous depuis chaque Pi vers un autre. Essayez d'exécuter une commande à distance sans ouvrir de session interactive, par exemple :

```
$ ssh utilisateur@192.168.1.11 'hostname'
```

Sécurité basique : changez les mots de passe par défaut pour éviter les accès non autorisés.

## Étape 5 : Transfert de fichiers via SCP

SCP utilise SSH pour copier des fichiers entre machines.

17. Créez un fichier test sur un Pi :

```
$ echo "Bonjour du réseau !" > test.txt
```

18. Copiez-le vers un autre Pi :

```
$ scp test.txt utilisateur@192.168.1.11:/home/utilisateur/
```

19. Sur le Pi cible, vérifiez :

```
$ cat test.txt
```

## Exercice 4

Chaque participant crée un fichier avec son nom et l'envoie à tous les autres. Vérifiez la réception de chaque côté.

## Étape bonus (si le temps le permet) : un serveur web simple

20. Sur un Pi, placez-vous dans un dossier contenant un fichier index.html simple, puis lancez le serveur intégré de Python :

```
$ python3 -m http.server 8000
```

21. Sur un autre Pi, accédez-y via un navigateur (si Desktop) à l'adresse `http://192.168.1.10:8000` (remplacez par l'IP réelle), ou en ligne de commande :

```
$ curl 192.168.1.10:8000
```

## Exercice 5

Créez une page HTML simple et servez-la. Accédez-y depuis les autres Pi.

## 2nde partie — Création de sous-réseaux (VLSM)

### Exercice 6 — Découpage du réseau 192.168.1.0/24 en 2 sous-réseaux

- Sous-réseau A : 10 hôtes → besoin d'un masque /28
  - Sous-réseau B : 5 hôtes → besoin d'un masque /29
22. Calculez les plages d'adresses valides (à la main, ou avec `ipcalc` si installé).

| Sous-réseau  | Adresse réseau | Masque                   | Plage utilisable   | Broadcast    |
|--------------|----------------|--------------------------|--------------------|--------------|
| A (10 hôtes) | 192.168.1.0    | /28<br>(255.255.255.240) | 192.168.1.1 - .14  | 192.168.1.15 |
| B (5 hôtes)  | 192.168.1.16   | /29<br>(255.255.255.248) | 192.168.1.17 - .22 | 192.168.1.23 |

Un /28 offre 16 adresses (14 utilisables) : suffisant pour 10 hôtes. Un /29 offre 8 adresses (6 utilisables) : suffisant pour 5 hôtes. Le sous-réseau B démarre juste après la fin du sous-réseau A pour ne pas chevaucher les plages.

23. Configurez quelques Pi dans le sous-réseau A (192.168.1.2 à .14) et quelques Pi dans le sous-réseau B (192.168.1.18 à .22) — réservez .1 et .17 pour le routeur de l'exercice 7.

**Point de contrôle** Les Pi du même sous-réseau doivent pouvoir se « pinguer » entre eux. En revanche, un Pi de A ne doit pas pouvoir joindre un Pi de B pour l'instant : il n'existe encore aucun routeur entre les deux. C'est ce que l'exercice 7 va résoudre.

### Exercice 7 — Routage statique entre deux sous-réseaux (simulation de routeur)

Cet exercice réutilise les deux sous-réseaux calculés à l'exercice 6 (192.168.1.0/28 et 192.168.1.16/29) : un Pi va jouer le rôle de routeur entre eux, avec une interface dans chaque sous-réseau.

24. Désignez un Pi comme routeur, avec deux interfaces :
- `eth0` → sous-réseau A, adresse 192.168.1.1/28
  - `usb0` (adaptateur USB-Ethernet — le nom exact peut varier, vérifiez avec `ip link show`) → sous-réseau B, adresse 192.168.1.17/29
25. Activez le routage IP sur ce Pi (obligatoire pour qu'il relaie le trafic entre ses deux interfaces) :

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

26. Sur les autres Pi du sous-réseau A, ajoutez une route statique vers le sous-réseau B, via le routeur :

```
$ sudo ip route add 192.168.1.16/29 via 192.168.1.1
```

27. Sur les Pi du sous-réseau B, ajoutez la route symétrique vers le sous-réseau A :

```
$ sudo ip route add 192.168.1.0/28 via 192.168.1.17
```

28. Testez la connectivité inter-sous-réseaux :

```
$ ping 192.168.1.18 # depuis un Pi du sous-réseau A
```

**Point de contrôle** Un Pi de A doit maintenant joindre un Pi de B en passant par le routeur. Utilisez `ip route` pour vérifier que la route ajoutée est bien présente, et `traceroute` (si disponible) pour observer le saut supplémentaire via le routeur.

**Astuce** Une route ajoutée avec `ip route add` disparaît au redémarrage du poste — c'est volontaire ici pour garder l'exercice réversible. En usage réel, on la rendrait permanente via la configuration NetworkManager (`nmcli connection modify ... +ipv4.routes`).

## Exercice 8 — Mini-chat en ligne de commande (Python + sockets)

Écrivez deux petits scripts Python : un serveur et un client, qui communiquent via un socket TCP. Créez-les avec nano plutôt que de les taper en ligne de commande, pour éviter les erreurs d'indentation.

### Sur le Pi serveur

```
$ nano serveur.py
```

```
import socket

s = socket.socket()
s.bind(('0.0.0.0', 12345))
s.listen(1)
conn, addr = s.accept()
print('Connecté :', addr)

while True:
    data = conn.recv(1024)
    if not data:
        break
    print('Reçu :', data.decode())
    conn.send(input('> ').encode())
```

```
$ python3 serveur.py
```

### Sur le Pi client

```
$ nano client.py
```

```
import socket
```

```
s = socket.socket()
s.connect(('192.168.1.10', 12345)) # remplacez par l'IP réelle du
serveur

while True:
    msg = input('> ')
    s.send(msg.encode())
    print('Réponse :', s.recv(1024).decode())

$ python3 client.py
```

**Attention** Remplacez 192.168.1.10 par l'adresse IP réelle du Pi qui fait tourner serveur.py, sans quoi la connexion échouera.

Défi : transformez ce mini-chat en serveur multiclient, capable de dialoguer avec plusieurs clients à la fois (pistes : le module select, ou un thread par client avec threading).