Université de Tamanrasset
Département Math et Informatique
Encadré par : Dr. Gattal Elhachmi **2025/2026**

Introduction Générale

Les travaux pratiques (TP) décrits dans ce document visent à combiner la simulation réseau réaliste avec GNS3 et l'automatisation via Python (utilisant la bibliothèque Netmiko). Ces TP couvrent des technologies avancées Cisco : EtherChannel pour l'agrégation de liens, HSRP pour la redondance des routeurs, PPP pour les connexions point-à-point, et VPN pour les réseaux privés virtuels sécurisés.

Objectifs pédagogiques généraux :

- Maîtriser la configuration de protocoles réseau avancés.
- Développer des scripts d'automatisation pour la surveillance et la maintenance.
- Analyser les performances et résoudre les problèmes courants.
- Préparer aux certifications comme CCNP Enterprise et aux compétences DevNet/Network Automation.

Prérequis :

- Installation de GNS3 avec images Cisco IOSv/IOSvL2.
- Python 3.x avec Netmiko installé (pip install netmiko).
- Connaissances de base en CLI Cisco et programmation Python.

Durée estimée : 2 heures par TP (simulation + programmation + analyse). **Évaluation :** Auto-évaluation via tableaux, quizzes, et rapports optionnels.

Recommandations :

- Créez un dépôt GitHub pour stocker les scripts Python et topologies GNS3.
- Intégrez des vidéos explicatives (ex. : tutoriels Cisco sur YouTube) via liens QR.
- Pour les défis avancés, encouragez les étudiants à expérimenter et documenter les résultats.

TP 1 : EtherChannel

Introduction Théorique

EtherChannel agrège plusieurs liens physiques en un canal logique unique, augmentant la bande passante et assurant la redondance via des protocoles comme LACP (Link Aggregation Control Protocol) ou PAgP. Il supporte le load balancing et la tolérance aux pannes.

Partie 1 : Simulation sur GNS3

Objectif

Configurer un EtherChannel LACP entre deux commutateurs avec gestion VLAN via VTP, et tester la redondance.

Topologie

- Deux commutateurs Cisco IOSvL2 : SW1 (serveur VTP), SW2 (client VTP).
- 4 liens Ethernet (interfaces e0/0 à e0/3).
- VLANs : 10 (SALES), 20 (ENGINEERING).
- Adresses IP exemple : SW1 management 192.168.1.1/24, SW2 192.168.1.2/24.

(Description pour Draw.io : Dessinez deux commutateurs connectés par 4 liens parallèles, avec étiquettes VLAN trunk sur le port-channel. Ajoutez des PCs sur VLANs pour tests.)

Étapes de Configuration

1. **Lancez GNS3, ajoutez SW1 et SW2, connectez-les via 4 liens.**
2. **Sur SW1 (VTP serveur) :**

```
SW1(config)# vtp domain CCNP

SW1(config)# vtp mode server

SW1(config)# vlan 10

SW1(config-vlan)# name SALES

SW1(config)# vlan 20

SW1(config-vlan)# name ENGINEERING

SW1(config)# interface vlan 10

SW1(config-if)# ip address 192.168.10.1 255.255.255.0
```

3. **Sur SW2 (VTP client) :**

```
SW2(config)# vtp domain CCNP

SW2(config)# vtp mode client
```

4. **Configuration EtherChannel :**

- **Sur SW1 :**

```
SW1(config)# interface range e0/0 - 3

SW1(config-if-range)# channel-group 1 mode active
```

SW1(config-if-range)# switchport mode trunk

SW1(config-if-range)# exit

SW1(config)# interface port-channel 1

SW1(config-if)# switchport mode trunk

SW1(config-if)# switchport trunk allowed vlan 10,20

- **Sur SW2 :**

SW2(config)# interface range e0/0 - 3

SW2(config-if-range)# channel-group 1 mode passive

SW2(config-if-range)# switchport mode trunk

SW2(config-if-range)# exit

SW2(config)# interface port-channel 1

SW2(config-if)# switchport mode trunk

SW2(config-if)# switchport trunk allowed vlan 10,20

5. Vérification :

show etherchannel summary (doit afficher "P" pour bundled)

show vtp status

show vlan brief

show interfaces port-channel 1 trunk

6. Test de redondance : Déconnectez un lien dans GNS3, vérifiez la continuité avec ping entre VLANs.

Analyse des Performances

- Mesurez la latence : ping 192.168.20.1 (ajoutez PCs sur VLAN 20).
- Bande passante : Utilisez iPerf entre hôtes pour simuler trafic et observer load balancing.
- Outils : show etherchannel load-balance pour vérifier l'algorithme.

Partie 2 : Programmation avec Python

Objectif

Script pour vérifier l'état EtherChannel via SSH, avec gestion d'erreurs et alertes.

Code Python

from netmiko import ConnectHandler, NetmikoTimeoutException, NetmikoAuthenticationException

```
import logging

logging.basicConfig(level=logging.INFO)

device = {

    "device_type": "cisco_ios",

    "host": "192.168.1.2",  # IP de SW2

    "username": "admin",

    "password": "cisco",

    "secret": "cisco"  # Enable password si nécessaire

}

try:

    with ConnectHandler(**device) as conn:

        output = conn.send_command("show etherchannel summary")

        if "P" in output and "Po1" in output:

            print("✅ EtherChannel est actif et tous les ports sont bundled.")

            logging.info("EtherChannel OK")

        else:

            print("⚠️ EtherChannel non actif ou ports non bundled.")

            logging.warning("Vérifiez les configurations LACP")

except (NetmikoTimeoutException, NetmikoAuthenticationException) as e:

    print(f"❌ Erreur de connexion : {e}")

    logging.error(f"Connexion échouée : {e}")
```

Exécution et Test

- Exécutez le script : python etherchannel_check.py.
- Ajoutez logging pour tracer les erreurs dans un fichier.

Quiz

1. **Question ouverte :** Quelle est la différence entre mode "active" et "passive" en LACP ? *(Réponse : Active initie la négociation, passive répond seulement.)*
2. **Choix multiple :** Si un port tombe en panne dans EtherChannel, que se passe-t-il ? a. Tout le canal tombe. b. Le trafic continue sur les ports restants. c. Nécessite reconfiguration manuelle. **Réponse correcte :** b.

Défi Avancé

Configurez un load balancing manuel :

SW1(config)# port-channel load-balance src-dst-ip

Testez avec trafic varié et analysez avec show etherchannel port-channel.

Auto-évaluation

Critère	Résultat Attendu	Résultat Obtenu	Problèmes	Solutions
EtherChannel bundled	Oui (P flags)			
VTP propagation	VLANs synchronisés			
Script Python	Connexion réussie			
Redondance testée	Trafic continu			

TP 2 : HSRP

Introduction Théorique

HSRP (Hot Standby Router Protocol) crée une passerelle virtuelle redondante. Le routeur avec la priorité la plus haute est actif ; en cas de panne, un basculement automatique se produit, avec support pour preemption et tracking d'interfaces.

Partie 1 : Simulation sur GNS3

Objectif

Configurer HSRP avec priorité, preemption et tracking pour une redondance fiable.

Topologie

- Deux routeurs Cisco IOSv : R1 (priorité haute), R2.
- Interface LAN : g0/0 sur 192.168.10.0/24.
- Passerelle virtuelle : 192.168.10.1.
- Interface à tracker : g0/1 (ex. lien WAN).

(Draw.io : Routeurs connectés à un switch LAN, avec PC client pinging la VIP.)

Étapes de Configuration

1. **Ajoutez R1, R2 et un switch dans GNS3, connectez g0/0.**
2. **Sur R1 :**

R1(config)# interface g0/0

R1(config-if)# ip address 192.168.10.2 255.255.255.0

R1(config-if)# no shutdown

R1(config-if)# standby 1 ip 192.168.10.1

R1(config-if)# standby 1 priority 110

R1(config-if)# standby 1 preempt

R1(config-if)# standby 1 track g0/1 20 (réduit priorité de 20 si g0/1 down)

3. **Sur R2 :**

R2(config)# interface g0/0

R2(config-if)# ip address 192.168.10.3 255.255.255.0

R2(config-if)# no shutdown

R2(config-if)# standby 1 ip 192.168.10.1

R2(config-if)# standby 1 priority 100

R2(config-if)# standby 1 preempt

4. **Vérification :**

show standby brief (Active/Standby/Listen)

show standby 1

5. **Test : Shutdown g0/1 sur R1, observez basculement (debug standby events).**

Analyse des Performances

- Ping continu depuis PC : ping 192.168.10.1 pendant shutdown.
- Temps de convergence : Mesurez avec show standby avant/après.

Partie 2 : Programmation avec Python

Objectif

Surveiller l'état HSRP et envoyer des alertes (ex. via email ou log).

Code Python

```
from netmiko import ConnectHandler, NetmikoTimeoutException, NetmikoAuthenticationException

import smtplib  # Pour alertes email optionnelles

router = {

   "device_type": "cisco_ios",

   "host": "192.168.10.2",  # IP R1

   "username": "admin",

   "password": "cisco"

}

try:

   with ConnectHandler(**router) as conn:

      output = conn.send_command("show standby brief")

      if "Active" in output:

         print("✅ R1 est actif (Active router).")

      elif "Standby" in output:

         print("ℹ️ R1 est en standby.")

      else:

         print("⚠️ État HSRP inconnu - alerte !")

         # Optionnel : Envoyer email (configurez SMTP)

         # smtplib.SMTP('smtp.example.com').sendmail(...)

except (NetmikoTimeoutException, NetmikoAuthenticationException) as e:
```

```
print(f"❌ Erreur : {e}")
```

Quiz

1. **Ouverte :** Expliquez le rôle du tracking en HSRP. *(Réponse : Réduit la priorité si une interface critique tombe, déclenchant basculement.)*
2. **Multiple :** La preemption permet : a. Basculement manuel seulement. b. Récupération automatique de l'actif. **Réponse :** b.

Défi Avancé

Multi-groupes HSRP pour VLANs : Configurez standby 10 pour VLAN 10, etc.

Auto-évaluation

Critère	Résultat Attendu	Résultat Obtenu	Problèmes	Solutions
R1 actif initial	Oui			
Basculement sur panne	<5s			
Script surveillance	État correct			

TP 3 : PPP

Introduction Théorique

PPP établit des connexions point-à-point sur liens série, avec authentification (PAP : clair, CHAP : chiffré) et support multilink pour agrégation.

Partie 1 : Simulation sur GNS3

Objectif

Configurer PPP avec CHAP/PAP, tester authentification et lien up/down.

Topologie

- R1 et R2 connectés via interfaces série s0/0/0.
- IP : 10.10.10.1/24 (R1), 10.10.10.2/24 (R2).

(Draw.io : Deux routeurs avec lien série DCE/DTE.)

Étapes de Configuration

1. **Connectez s0/0/0 dans GNS3 (assignez clock rate sur DCE : clock rate 64000).**
2. **CHAP sur R1 :**

```
R1(config)# username R2 password cisco123
R1(config)# interface s0/0/0
R1(config-if)# encapsulation ppp
R1(config-if)# ppp authentication chap
R1(config-if)# ip address 10.10.10.1 255.255.255.0
R1(config-if)# no shutdown
```

3. **Sur R2 (symétrique) :**

```
R2(config)# username R1 password cisco123
R2(config)# interface s0/0/0
R2(config-if)# encapsulation ppp
R2(config-if)# ppp authentication chap
R2(config-if)# ip address 10.10.10.2 255.255.255.0
```

4. **Pour PAP (test alternatif) :**

```
R1(config-if)# ppp authentication pap
R1(config-if)# ppp pap sent-username R1 password cisco123
```

(Symétrique sur R2)

5.Vérification :

show interfaces s0/0/0 (Protocol up/up)

debug ppp authentication (pour logs)

Analyse des Performances

- Comparez temps auth : CHAP plus sécurisé mais similaire en vitesse.
- Test lien : Shutdown interface, observez down.

Partie 2 : Programmation avec Python

Objectif

Vérifier statut PPP et alerter sur down.

Code Python

```
from netmiko import ConnectHandler, NetmikoTimeoutException,
NetmikoAuthenticationException

router = {

    "device_type": "cisco_ios",

    "host": "10.10.10.1",

    "username": "admin",

    "password": "cisco"

}

try:

    with ConnectHandler(**router) as conn:

        status = conn.send_command("show interfaces s0/0/0 | include line protocol")

        if "up" in status.lower():

            print("✅ Lien PPP up et opérationnel.")

        else:

            print("❌ Lien PPP down - Vérifiez câble/authentification.")

except Exception as e:

    print(f"❌ Erreur : {e}")
```

Quiz

1. **Ouverte :** Pourquoi CHAP est-il préférable à PAP ? *(Réponse : Hash challenge-response, pas de mot de passe clair.)*
2. **Multiple :** PPP supporte : a. Seulement Ethernet. b. Liens série avec authentification. **Réponse :** b.

Défi Avancé

Multilink PPP : ppp multilink group 1 sur plusieurs interfaces série.

Auto-évaluation

Critère	Résultat Attendu	Résultat Obtenu	Problèmes	Solutions
Auth CHAP réussie	Up/up			
Comparaison PAP/CHAP	Logs debug			
Script check	Statut correct			

TP 4 : VPN

Introduction Théorique

VPN IPSec Site-to-Site crée des tunnels sécurisés via ISAKMP (Phase 1 : IKE) et IPSec (Phase 2 : ESP/AH), avec ACL pour trafic protégé.

Partie 1 : Simulation sur GNS3

Objectif

Configurer IPSec VPN avec pre-shared key et ACL.

Topologie

- R1 (site A : LAN 192.168.1.0/24), R2 (site B : 192.168.2.0/24).
- Interfaces WAN : g0/0 (200.1.1.1 et 200.1.1.2).
- Lien "Internet" simulé entre WAN.

(Draw.io : Deux sites avec routeurs, LANs, et tunnel IPSec entre WAN.)

Étapes de Configuration

1. **ACL sur R1 :**

R1(config)# access-list 100 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255

2.ISAKMP et IPSec sur R1 :

R1(config)# crypto isakmp policy 1

R1(config-isakmp)# encryption aes

R1(config-isakmp)# hash sha

R1(config-isakmp)# authentication pre-share

R1(config-isakmp)# group 2

R1(config-isakmp)# exit

R1(config)# crypto isakmp key vpn123 address 200.1.1.2

R1(config)# crypto ipsec transform-set VPN-SET esp-aes esp-sha-hmac

R1(config)# crypto map VPN-MAP 10 ipsec-isakmp

R1(config-crypto-map)# set peer 200.1.1.2

R1(config-crypto-map)# set transform-set VPN-SET

R1(config-crypto-map)# match address 100

R1(config)# interface g0/0

R1(config-if)# crypto map VPN-MAP

R1(config-if)# ip address 200.1.1.1 255.255.255.0

3 Symétrique sur R2 (inverser ACL et peer).

4 Vérification :

show crypto isakmp sa (QM_IDLE pour actif)

show crypto ipsec sa (pkts encaps/decaps)

Analyse des Performances

- Ping entre LANs : ping 192.168.2.1 depuis site A.
- Capture Wireshark sur WAN pour vérifier chiffrement (paquets ESP).

Partie 2 : Programmation avec Python

Objectif

Monitorer tunnels VPN et alerter sur inactivité.

Code Python

```
from netmiko import ConnectHandler, NetmikoTimeoutException,
NetmikoAuthenticationException

router = {

   "device_type": "cisco_ios",

   "host": "200.1.1.1",

   "username": "admin",

   "password": "cisco"

}

try:

   with ConnectHandler(**router) as conn:

      output = conn.send_command("show crypto isakmp sa")

      if "QM_IDLE" in output:

         print("✅ Tunnel VPN actif (QM_IDLE).")

      else:

         print("⚠ Tunnel down ou négociation échouée.")

except Exception as e:

   print(f"❌ Erreur : {e}")
```

Quiz

1. **Ouverte :** Différence ISAKMP vs IPSec ? *(Réponse : ISAKMP pour négociation clés, IPSec pour protection données.)*
2. **Multiple :** ESP fournit : a. Chiffrement + intégrité. b. Seulement authentification. **Réponse :** a.

Défi Avancé

DMVPN : Ajoutez crypto isakmp key vpn123 address 0.0.0.0 pour multipoint.

Critère	Résultat Attendu	Résultat Obtenu	Problèmes	Solutions
SA établie	QM_IDLE			
Trafic chiffré	Pkts >0			
Script monitor	Statut OK			

TP 5 Configuration Automatisée d'un Tunnel GRE avec Python et GNS3

*__Objectif de l'exercice pratique__

Créer un tunnel GRE entre deux routeurs simulés dans GNS3, puis utiliser Python pour automatiser la configuration des routeurs via Telnet ou SSH.
L'objectif est de configurer un tunnel GRE de manière automatique, évitant ainsi la configuration manuelle sur chaque routeur.

* Prérequis

Avant de commencer, assurez-vous de disposer des outils suivants :

Outil	Fonctionnalité
GNS3	Simulation du réseau et des routeurs
Image Cisco IOS	Pour exécuter des routeurs Cisco (ex. : 7200)
Python 3.x	Pour exécuter le script d'automatisation
Netmiko ou Paramiko	Bibliothèques Python pour gérer les équipements réseau via SSH/Telnet

Installation des dépendances Python :
pip install netmiko

*** Étape 1 : Configuration du réseau dans GNS3**

1. Ouvrez GNS3 et créez un nouveau projet.
2. Ajoutez deux routeurs Cisco (par exemple, modèle 7200).
3. Connectez-les avec un lien Serial (ou Ethernet selon l'image IOS utilisée).
4. Ajoutez un Cloud ou un NAT si vous souhaitez une connectivité externe (facultatif pour cet exercice).
5. Configurez les adresses IP comme suit :

Équipement	Interface	Adresse IP
Routeur A	s0/0	100.0.0.1/30
Routeur B	s0/0	100.0.0.2/30
Routeur A	LAN (e0/0)	192.168.1.1/24
Routeur B	LAN (e0/0)	192.168.2.1/24

*** Étape 2 : Script Python pour configurer le tunnel GRE**

Le script suivant utilise Netmiko pour automatiser la configuration d'un tunnel GRE sur les deux routeurs.

(code Python fourni dans le texte original)

*Explication du script

Chaque commande et ligne de code est expliquée dans le texte original, y compris la création du tunnel,
la définition des adresses IP, la source/destination du tunnel et l'ajout d'une route statique.

* **Étape 3 : Exécution**

1. Lancez les routeurs dans GNS3.
2. Assurez-vous que chaque routeur est accessible via SSH ou Telnet depuis votre machine.
3. Exécutez le script Python :
   python gre_tunnel_config.py
4. Vérifiez dans GNS3 que le tunnel est opérationnel avec :
   show interface tunnel0
5. Testez la connectivité entre les LANs : ping 192.168.2.1 et ping 192.168.1.1

* **Résultat attendu**

Le tunnel doit être opérationnel (up/up) et les pings doivent réussir entre les deux LANs.

* **Ajouts optionnels**

1. Ajout de routes dynamiques avec OSPF.
2. Vérification de la table de routage (show ip route).
3. Sécurisation du tunnel GRE avec IPSec (crypto map).

*** Fourniture d'un projet GNS3 et du script**

Annexes

Liste des Commandes Clés

- EtherChannel : channel-group, show etherchannel summary.
- HSRP : standby ip, show standby.
- PPP : encapsulation ppp, show interfaces serial.
- VPN : crypto map, show crypto sa.

Ressources

- Cisco DevNet : https://developer.cisco.com
- Netmiko Docs : https://ktbyers.github.io/netmiko/
- GNS3 : https://www.gns3.com
- GitHub Repo suggéré : Créez un repo avec scripts et .gns3proj.