BTS SIO — Option SISR — Fiche de Réalisation Professionnelle E6

Mise en place d'une politique Fair Use

Limitation de bande passante à 10 Mbps via les Delay Pools Squid

SRV-WEB-01 (10.11.3.2 — Ubuntu) · CL-WIN11-01 (10.11.3.23) · sio.local

Contexte métier — Politique de Fair Use

Dans une PME dont le lien WAN est partagé entre la navigation web, la VoIP, les applications ERP et les sauvegardes, un seul poste effectuant un téléchargement non contrôlé peut **saturer l'intégralité de la bande passante** et rendre la VoIP inutilisable (jitter, coupures) ou les applicatifs métier inaccessibles (timeouts).

La politique de **Fair Use à 10 Mbps** garantit :

- **Qualité de service garantie :** les flux critiques (VoIP, ERP, sauvegardes IPsec) conservent leur bande passante quelle que soit l'activité de navigation
- **Confort utilisateur maintenu :** 10 Mbps représente un débit confortable pour la navigation web, le webmail et les téléchargements métier
- **Maîtrise des coûts :** prévention de la surfacturation liée au dépassement du quota FAI

I. Prérequis

Architecture de référence

Machine	IP	Rôle	OS
SRV-WEB-01	`10.11.3.2`	`Proxy Squid — Port 3127`	`Ubuntu 22.04 LTS`
pfSense-01	`10.11.3.17`	`Passerelle / Pare-feu`	`pfSense 2.7.2 CE`
CL-WIN11-01	`10.11.3.23`	`Poste client (cible du bridage)`	`Windows 11 Pro`
SRV-AD-01	`10.11.3.18`	`DC / DNS / GPO`	`Windows Server 2025`

Conditions préalables

- **Squid fonctionnel** et en écoute sur le port `3127` — vérifiable avec `sudo systemctl status squid`
- **Proxy configuré en mode manuel** sur les postes clients via GPO (`GPO_ProxySquid`) — tout le trafic HTTP/HTTPS transite par `10.11.3.2:3127`
- **Accès SSH administrateur** disponible vers `SRV-WEB-01` (utilisateur `admin-web` avec `sudo`)
- Sauvegarde de `squid.conf` effectuée avant modification : `sudo cp /etc/squid/squid.conf /etc/squid/squid.conf.bak`

Conversion du débit — Rappel

10 Mbps = 10 000 000 bits/s = 1 250 000 octets/s = ~1,19 Mio/s

> Squid exprime les paramètres de Delay Pools en **octets par seconde**. La valeur à configurer est donc `1250000` (1 250 000 octets/s = 10 Mbps).

II. Configuration Squid — Delay Pools

Les **Delay Pools** de Squid sont un mécanisme natif de contrôle de débit basé sur le principe du **Token Bucket** : un seau de jetons se remplit à un débit constant (le débit autorisé) et se vide lorsque des données transitent. Si le seau est vide, Squid ralentit la transmission jusqu'à ce que de nouveaux jetons s'accumulent — simulant ainsi une limitation de bande passante.

Principe des classes de Delay Pools

Classe	Granularité	Cas d'usage
Class 1	Global — 1 seul seau pour tout le pool	Limitation globale pour tous les clients
Class 2	Global + Par réseau	Limitation par sous-réseau
Class 3	Global + Par réseau + Par hôte	Limitation individuelle par IP ← cas retenu

> **Choix de la Class 3**
> La **Class 3** est retenue car elle permet une limitation **individuelle par adresse IP**. Ainsi, `CL-WIN11-01 (10.11.3.23)` est limité à 10 Mbps indépendamment des autres postes du réseau. Les autres clients (10.11.3.24, .25…) ne sont pas affectés.

Édition du fichier squid.conf

> 🗁 **Fichier / Emplacement : `/etc/squid/squid.conf`**

Ouvrir le fichier de configuration avec l'éditeur `nano` :

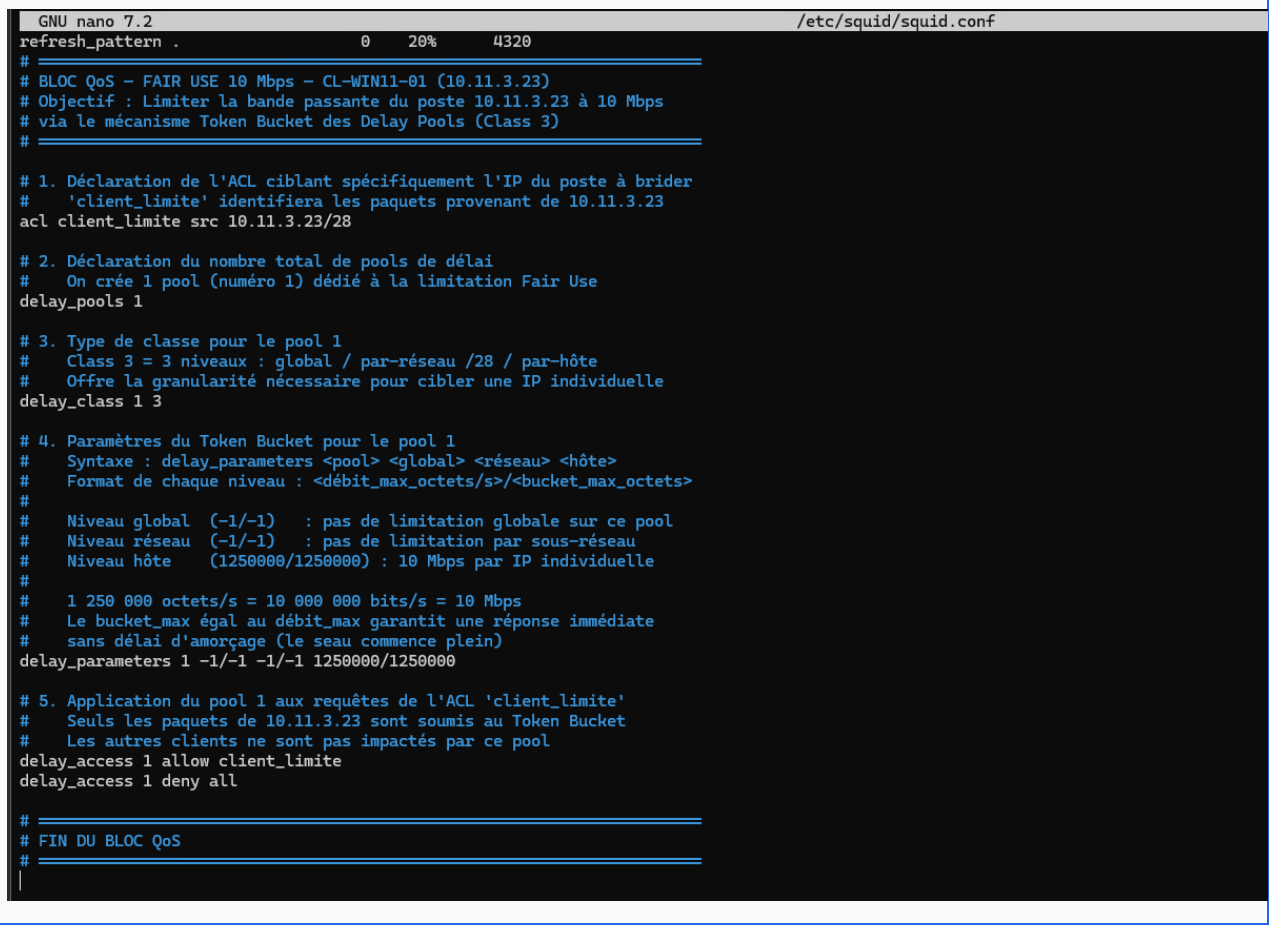
```
sudo nano /etc/squid/squid.conf
```

Ajouter le bloc suivant **à la fin du fichier**, après toutes les ACL et règles existantes :

```
# ═══════════════════════════════════════════════════════════════════
# BLOC QoS — FAIR USE 10 Mbps — CL-WIN11-01 (10.11.3.23)
# Objectif : Limiter la bande passante du poste 10.11.3.23 à 10 Mbps
# via le mécanisme Token Bucket des Delay Pools (Class 3)
# ═══════════════════════════════════════════════════════════════════

# 1. Déclaration de l'ACL ciblant spécifiquement l'IP du poste à brider
#    'client_limite' identifiera les paquets provenant de 10.11.3.23
acl client_limite src 10.11.3.23/28
```

```
# 2. Déclaration du nombre total de pools de délai
#    On crée 1 pool (numéro 1) dédié à la limitation Fair Use
delay_pools 1

# 3. Type de classe pour le pool 1
#    Class 3 = 3 niveaux : global / par-réseau /28 / par-hôte
#    Offre la granularité nécessaire pour cibler une IP individuelle
delay_class 1 3

# 4. Paramètres du Token Bucket pour le pool 1
#    Syntaxe : delay_parameters <pool> <global> <réseau> <hôte>
#    Format de chaque niveau : <débit_max_octets/s>/<bucket_max_octets>
#
#    Niveau global  (-1/-1)    : pas de limitation globale sur ce pool
#    Niveau réseau  (-1/-1)    : pas de limitation par sous-réseau
#    Niveau hôte    (1250000/1250000) : 10 Mbps par IP individuelle
#
#    1 250 000 octets/s = 10 000 000 bits/s = 10 Mbps
#    Le bucket_max égal au débit_max garantit une réponse immédiate
#    sans délai d'amorçage (le seau commence plein)
delay_parameters 1 -1/-1 -1/-1 1250000/1250000

# 5. Application du pool 1 aux requêtes de l'ACL 'client_limite'
#    Seuls les paquets de 10.11.3.23 sont soumis au Token Bucket
#    Les autres clients ne sont pas impactés par ce pool
delay_access 1 allow client_limite
delay_access 1 deny all

# ═══════════════════════════════════════════════════════════════════════
# FIN DU BLOC QoS
# ═══════════════════════════════════════════════════════════════════════
```

CAPTURE D'ÉCRAN N°1 — **Éditeur nano ouvert sur /etc/squid/squid.conf — Le bloc Delay Pools est visible en bas du fichier avec les lignes acl, delay_pools, delay_class, delay_parameters et delay_access.**



Validation de la syntaxe et rechargement

Avant tout rechargement, **valider la syntaxe** du fichier pour détecter toute erreur de configuration :

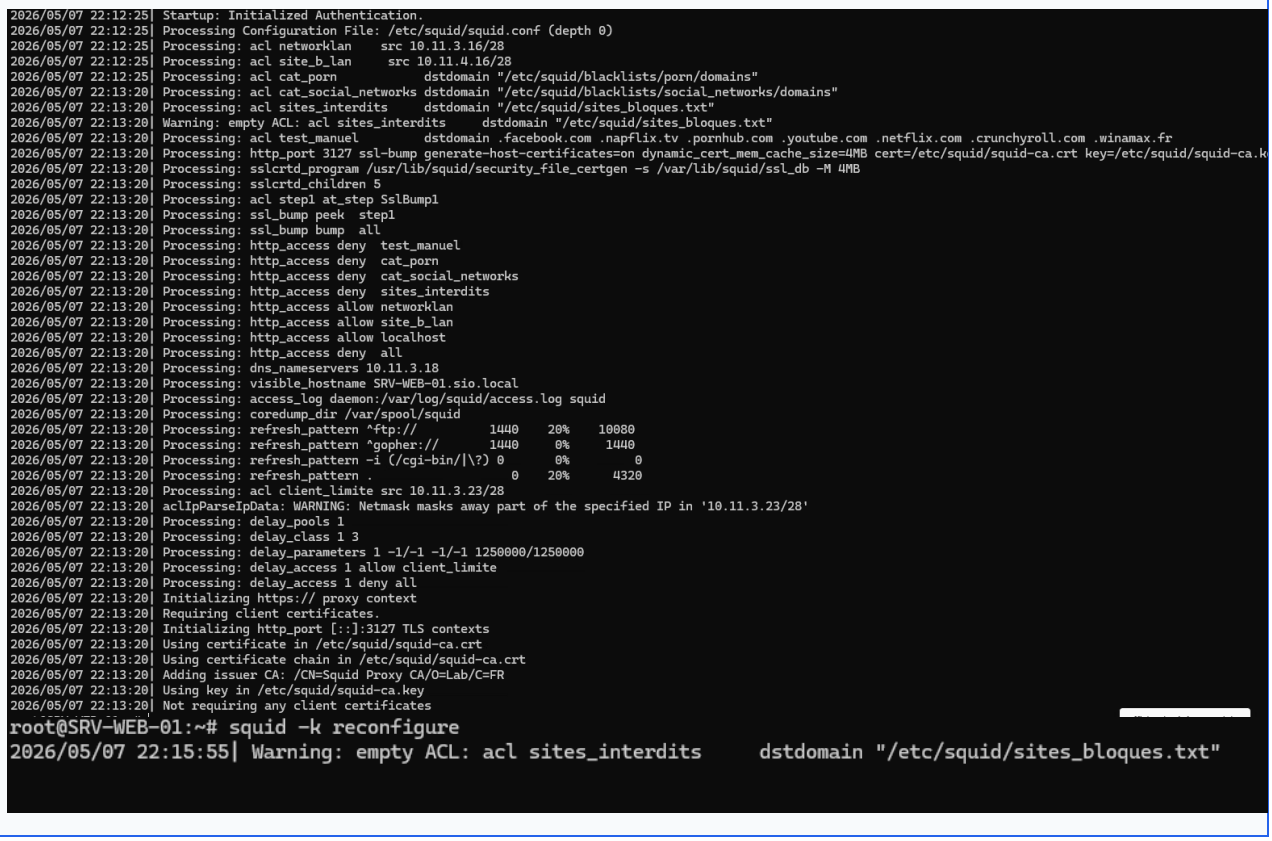
```
# Validation syntaxique — aucun message d'erreur ne doit apparaître
sudo squid -k parse

# Résultat attendu (succès) :
# 2026/... kid1| Processing Configuration File: /etc/squid/squid.conf...
# (aucune ligne 'ERROR' ou 'FATAL')

# Rechargement de la configuration sans coupure de service (graceful reload)
sudo squid -k reconfigure

# Vérification que le service est toujours actif après rechargement
sudo systemctl status squid
```

Différence entre reconfigure et restart

squid -k reconfigure : Recharge la configuration **sans couper** les connexions en cours — méthode recommandée en production.

systemctl restart squid : Arrête puis redémarre complètement Squid — coupe toutes les sessions actives — à utiliser uniquement en environnement de test.

CAPTURE D'ÉCRAN N°2 — **Terminal SRV-WEB-01 — Résultat de 'sudo squid -k parse' sans aucune ligne ERROR/FATAL, suivi de 'sudo squid -k reconfigure' avec confirmation de rechargement.**



III. Tests et Validation

Méthode 1 — Test de téléchargement depuis CL-WIN11-01 (méthode principale)

La preuve la plus directe est d'effectuer un téléchargement d'un fichier volumineux depuis `CL-WIN11-01` avec le proxy actif, et d'observer le débit dans le Gestionnaire des tâches Windows.

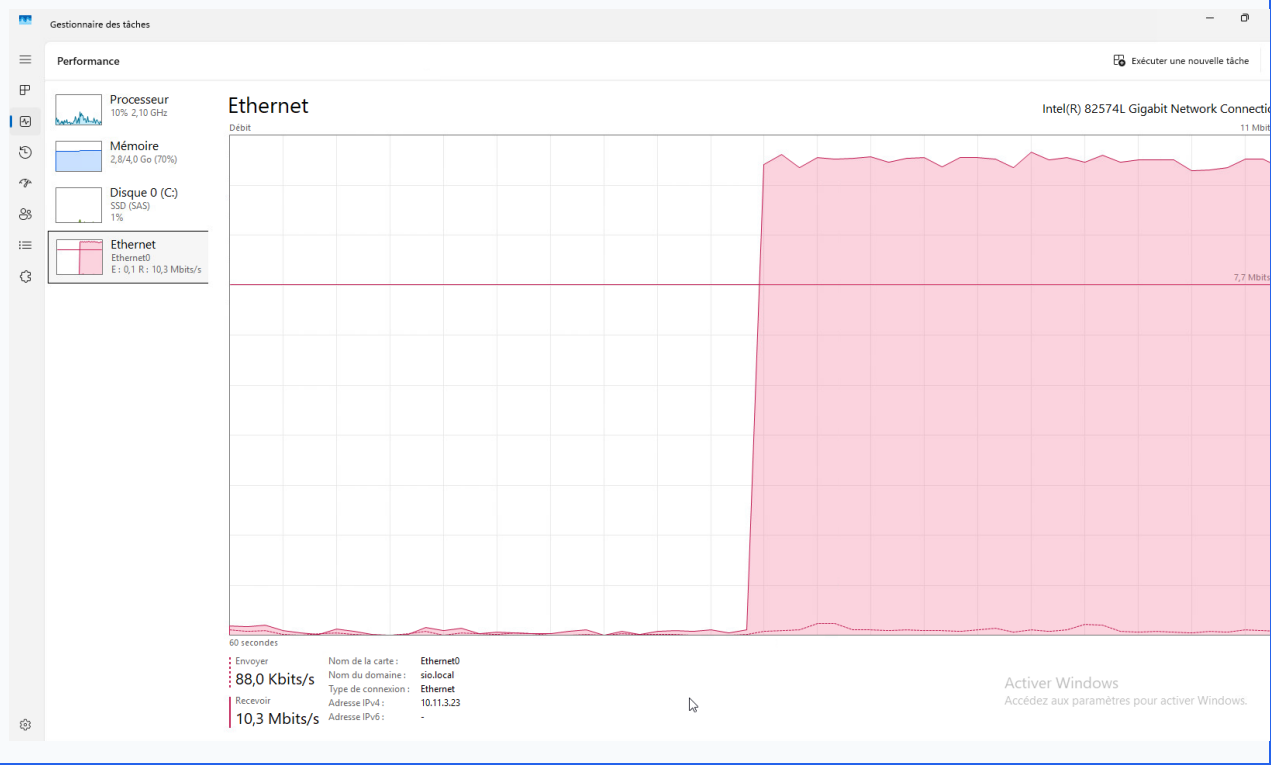
1. **Préparation — Serveur de test** : sur `SRV-WEB-01`, créer un fichier test de 100 Mo accessible via Apache2 (co-localisé) :

```
# Créer un fichier de 100 Mo avec des données pseudo-aléatoires pour le test
sudo dd if=/dev/urandom of=/var/www/html/testfile_100MB.bin bs=1M count=100
# Le fichier sera accessible via : http://10.11.3.2/testfile_100MB.bin
```

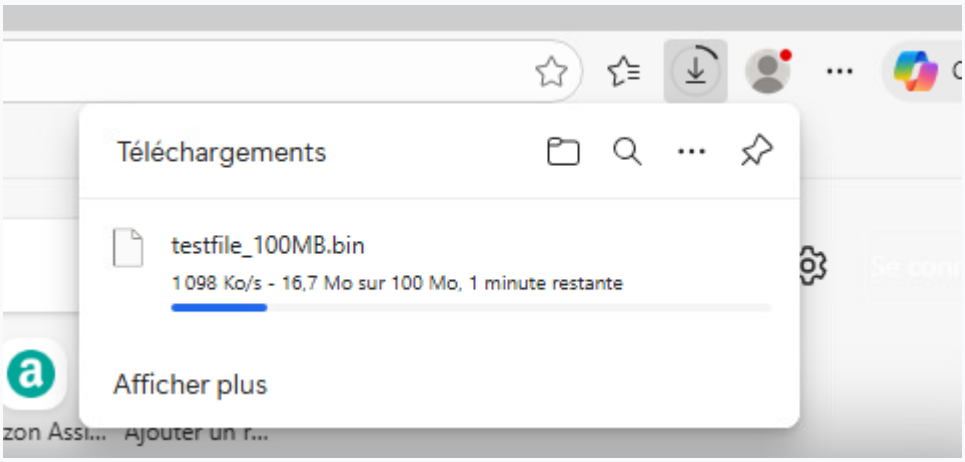
2. **Test depuis CL-WIN11-01** : Ouvrir `Microsoft Edge` (avec le proxy configuré), accéder à l'URL du fichier test :
   - URL : `http://10.11.3.2/testfile_100MB.bin`
   - Le navigateur déclenche le téléchargement — le proxy Squid intercepte et applique le Delay Pool

3. **Observation dans le Gestionnaire des tâches** : pendant le téléchargement, ouvrir le Gestionnaire des tâches (`Ctrl+Shift+Esc`) :
   - Onglet **Performance > Ethernet**
   - Le débit de réception doit être **plafonné à 10 Mbps (1,25 Mo/s)**
   - Sans le Delay Pool, le même fichier se téléchargerait à la vitesse maximale du LAN (100–1000 Mbps)

CAPTURE D'ÉCRAN N°3 — Gestionnaire des tâches Windows 11 — Onglet Performance > Ethernet sur CL-WIN11-01 pendant le téléchargement de testfile_100MB.bin. Le graphique 'Receive' est plafonné à 10 Mbps (~1,25 Mo/s). C'est la preuve visuelle du bridage.



CAPTURE D'ÉCRAN N°4 — Navigateur Edge sur CL-WIN11-01 — Barre de progression du téléchargement affichant une vitesse de ~1,25 Mo/s (10 Mbps). Preuve complémentaire du respect du quota Fair Use.

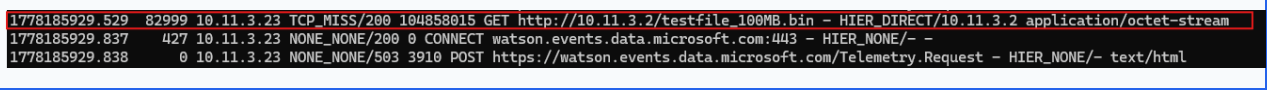


Méthode 2 — Vérification côté serveur (logs Squid)

Les logs Squid permettent de confirmer que les requêtes de `10.11.3.23` passent bien par le pool de délai :

```
# Surveillance en temps réel du log d'accès Squid pendant le téléchargement
sudo tail -f /var/log/squid/access.log | grep 10.11.3.23

# Format des lignes attendues :
# <timestamp> <durée_ms> 10.11.3.23 TCP_MISS/200 <octets> GET http://...
DIRECT/...

# La durée_ms élevée confirme que Squid ralentit intentionnellement la réponse
```

CAPTURE D'ÉCRAN N°5 — Terminal SRV-WEB-01 — Résultat de 'tail -f /var/log/squid/access.log | grep 10.11.3.23' pendant le téléchargement. Les lignes TCP_MISS/200 avec des durées élevées confirment l'application du Delay Pool.



Méthode 3 — Test comparatif (référence avant/après)

Scénario	Débit observé	Conclusion
Sans Delay Pool (baseline)	~100 Mbps (LAN full speed)	Pas de limitation — référence
Avec Delay Pool actif	~10 Mbps (1,25 Mo/s)	Limitation Fair Use appliquée ☑
Autre poste (10.11.3.24)	~100 Mbps (non affecté)	Limitation ciblée — pas d'impact tiers ☑

☑ Critères de validation

Critère 1 : Le débit de téléchargement sur `CL-WIN11-01` est **plafonné à 10 Mbps (±5 %)** en observation Gestionnaire des tâches

Critère 2 : Les autres postes du réseau **ne sont pas affectés** — leur débit reste au maximum du LAN

Critère 3 : La commande `squid -k parse` **ne génère aucune erreur** — configuration syntaxiquement valide

Critère 4 : Le service Squid reste actif après `squid -k reconfigure` — pas de coupure de service

IV. Argumentaire pour l'oral

Justification technique du choix des Delay Pools vs autres solutions

Critère	Delay Pools Squid ☑	QoS pfSense (Traffic Shaper)
Niveau d'action	Applicatif (HTTP/HTTPS déchiffré)	Réseau (flux IP chiffré)
Granularité	Par IP, par URL, par type MIME	Par IP, par port TCP/UDP
Couplage avec les ACL	Natif — même fichier squid.conf	Configuration séparée pfSense
Visibilité dans les logs	Oui — access.log complet	Non — paquet chiffré post-proxy
Cas d'usage HTTPS	Oui — fonctionne via SSL Bump	Oui — niveau réseau seulement

*"Les Delay Pools de Squid sont la solution native la plus cohérente dans notre architecture car le proxy est **déjà le point de passage obligatoire** de tous les flux web (HTTP et HTTPS via SSL Bump). Configurer la limitation directement dans Squid évite d'introduire un équipement supplémentaire et centralise la politique de QoS avec les politiques de filtrage existantes (blacklists UT1).*

*La granularité de la `Class 3` permet une limitation **chirurgicale** : seul le poste `10.11.3.23` est bridé. Les flux critiques (VoIP, ERP, sauvegardes IPsec inter-sites) ne transitent pas par Squid et ne sont donc **pas impactés** — ce qui est exactement l'objectif de la politique Fair Use."*

Évolutions possibles pour un environnement de production

- **Extension à l'ensemble du réseau :** remplacer `acl client_limite src 10.11.3.23/32` par `acl reseau_lan src 10.11.3.16/28` pour appliquer le quota à tous les postes du LAN Site A
- **Différenciation par groupe AD :** combiner avec l'authentification NTLM/Kerberos pour appliquer des quotas différents selon les OU Active Directory (ex: 50 Mbps pour le service IT, 10 Mbps pour les autres)
- **Plages horaires :** utiliser les ACL Squid de type `time` pour désactiver la limitation en dehors des heures ouvrées (22h–8h) et libérer toute la bande passante pour les sauvegardes nocturnes
- **Monitoring du débit réel :** coupler avec un trigger Zabbix sur l'item `net.if.in[eth0]` de `SRV-WEB-01` pour alerter si le débit agrégé dépasse 80 % du lien WAN