Agrégation de liens xDSL sur un réseau radio

Soutenance TX

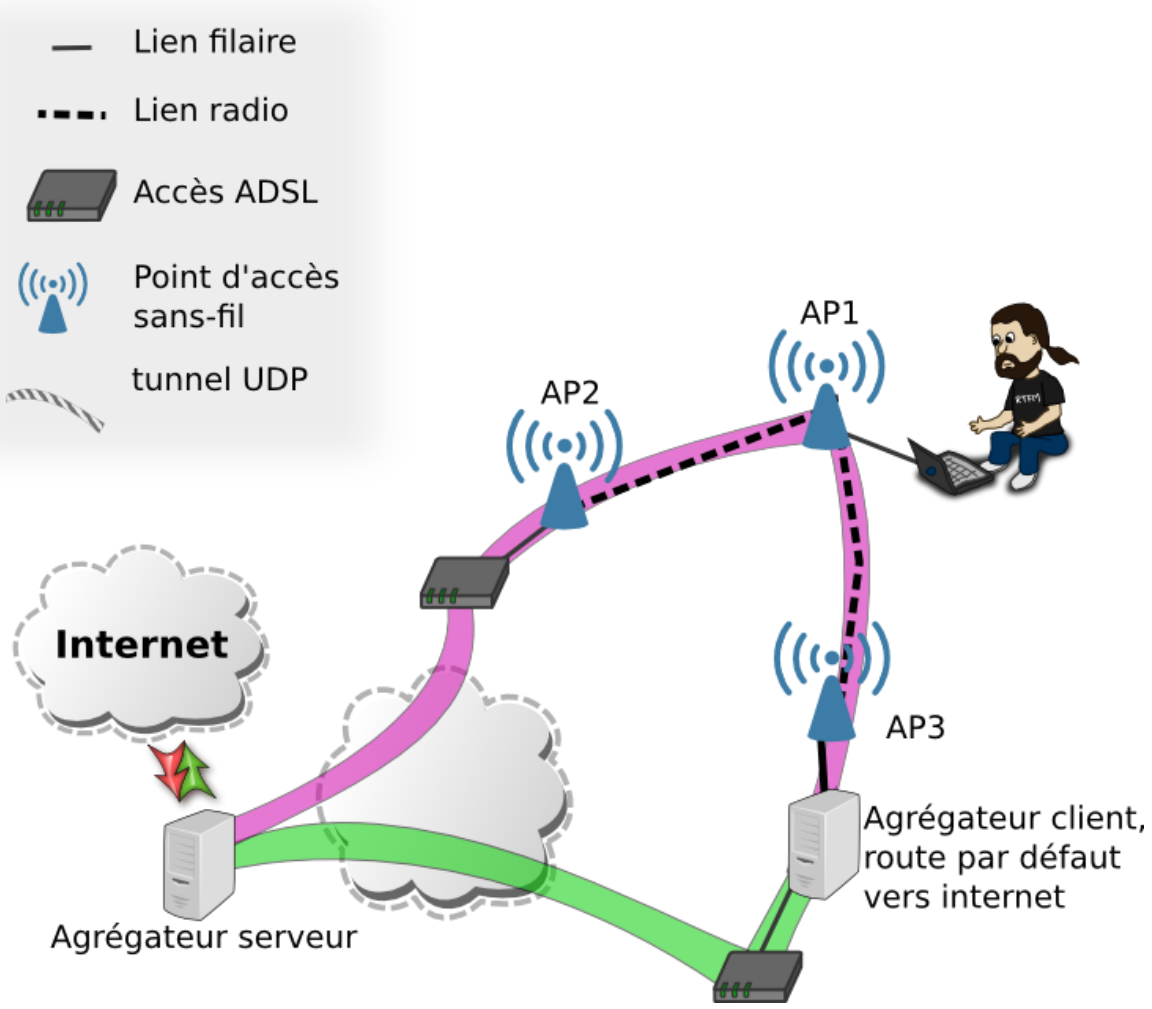
Suiveur: Stéphane Crozat

Commanditaire: tetaneutral.net/Laurent Guerby

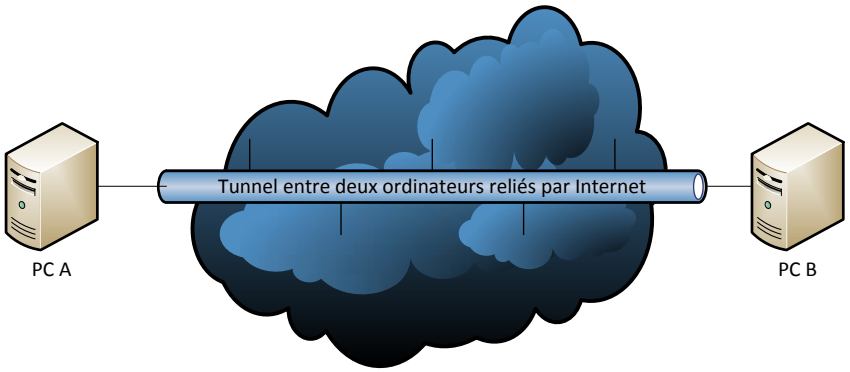
utc Université de Technologie Compiègne

Introduction

Introduction: schéma



Définition d'un tunnel



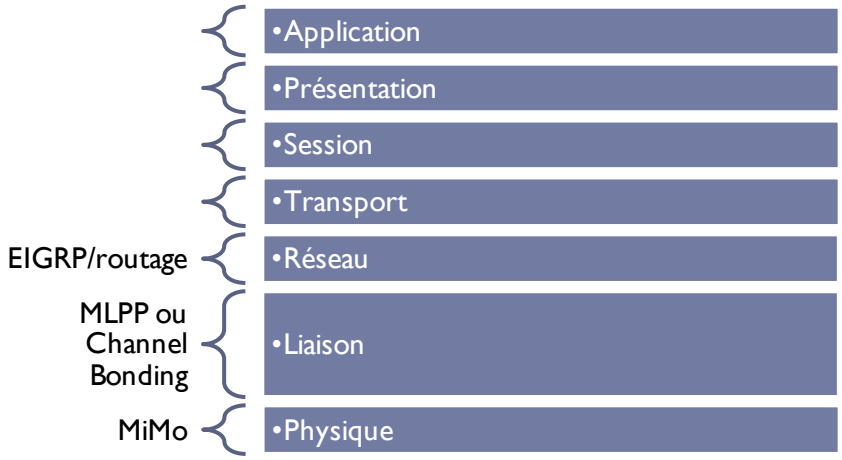
- Encapsulation du trafic au-dessus d'UDP
- Tunnel
  - Niveau 2 et supérieur (tap)
  - Niveau 3 et supérieur (tun)
- Overhead (charge induite)
  - Tap: 4,4%
  - Tun: 3.0%

Introduction: le besoin

- Collecter le trafic
  - De et vers Internet
- Utilisation de plusieurs lignes xDSL
  - Distantes
  - Hétérogènes (débit/opérateur)
  - Niveau 2 (IPv6)

=> *Agrégation*

Les solutions « classiques »



- ~~EIGRP~~
  - Niveau 3…
- ~~MLPPP~~
  - Combinaisons de liens physiques
  - Côté abonné et opérateur
  - Performance fondée sur le lien le plus faible
- ~~Channel Bonding (802.3ad)~~
  - Niveau commutateur
  - Plusieurs modes de fonctionnements
  - Gestion : LACP ou PaGP
    - Ne fonctionne que sur ethernet

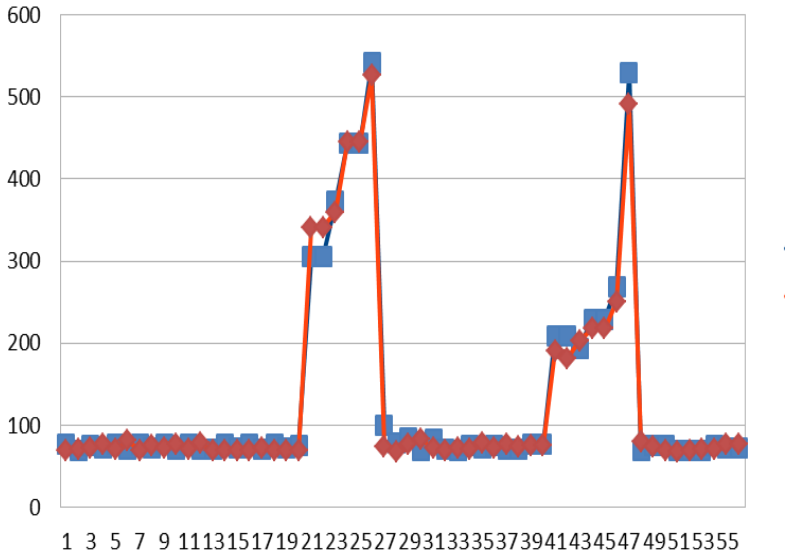
L’étude expérimentale

Etude expérimentale

- Objectif
  - Hiérarchiser les liens vers Internet en fonction de leur capacité
- Mesurer la capacité d'un lien
  - Son débit max un instant t
- Solution ?
  - Surveiller le temps de réponse (délai) d'un ping
- Relation entre saturation du tunnel et le délai d'un ping
  - Déterminé empiriquement

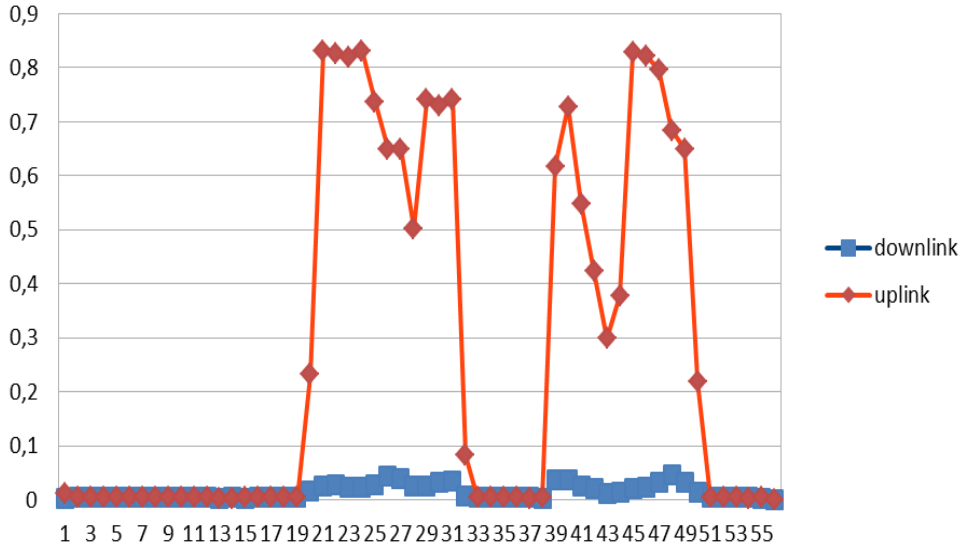
Comportement d'un lien ADSL sous la charge

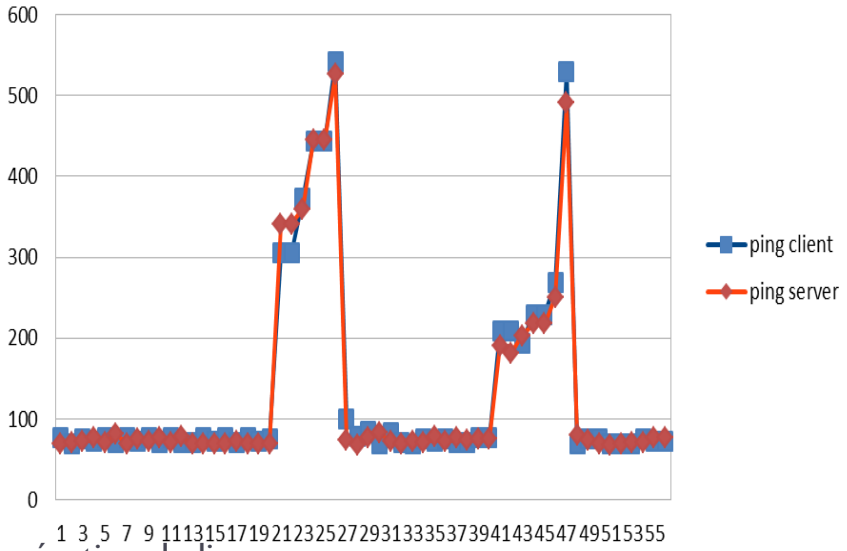
- Free (Freebox)
  - Ping non corrélé sur la saturation d'un lien
- OVH/FDN
  - Ping corrélé sur la saturation d'un lien
- Mesures stockées:
  - Toutes les secondes
  - Stats des paquets (BP)
  - Ping
- Utilisation d'iperf pour générer un trafic



Les résultats d'un test de montée en charge en TCP

- TCP: réduction de sa fenêtre d'émission
- Montée du ping significatif
- => corrélation du ping et de la saturation

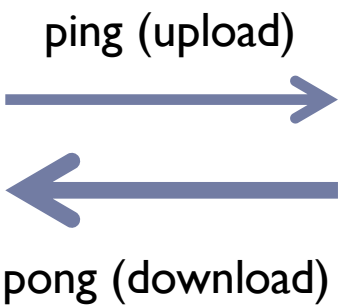




- Même résultat avec UDP

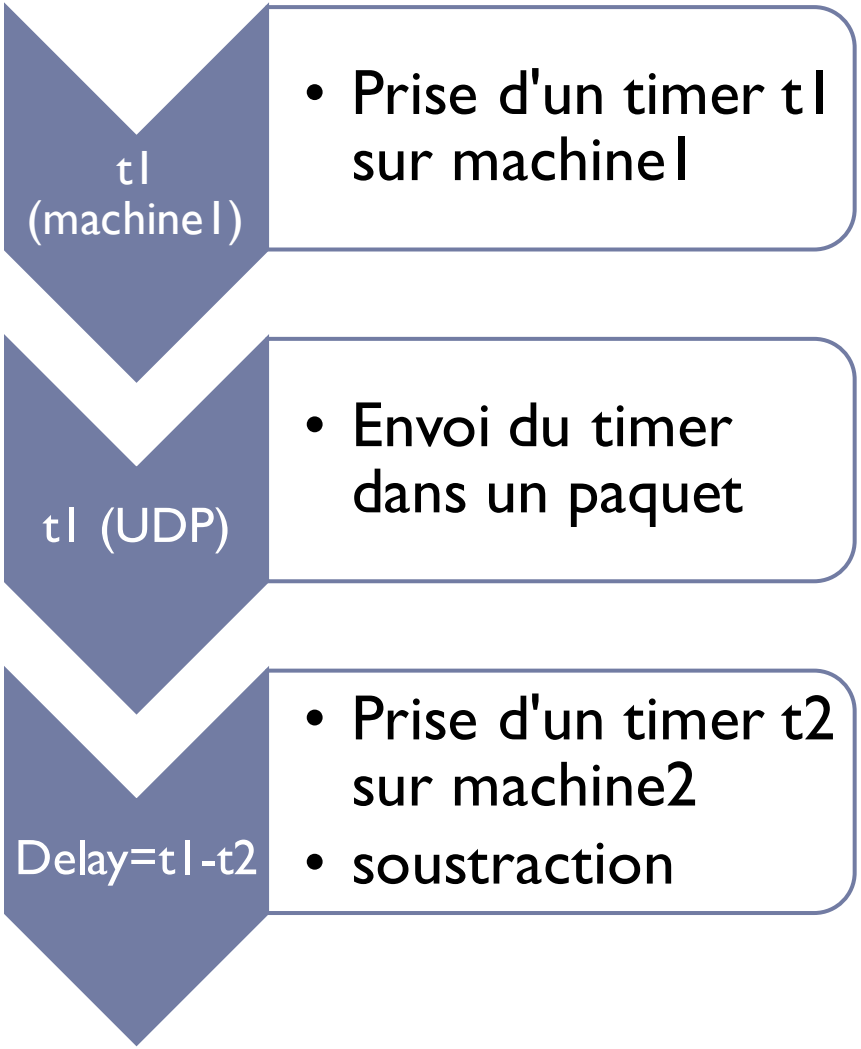
Constat de la détection par ping

- Ping
  - Détection de saturation d'un lien ADSL
  - Non connaissance du sens de la saturation



Où est la saturation ?

Détection de saturation par demi-délai



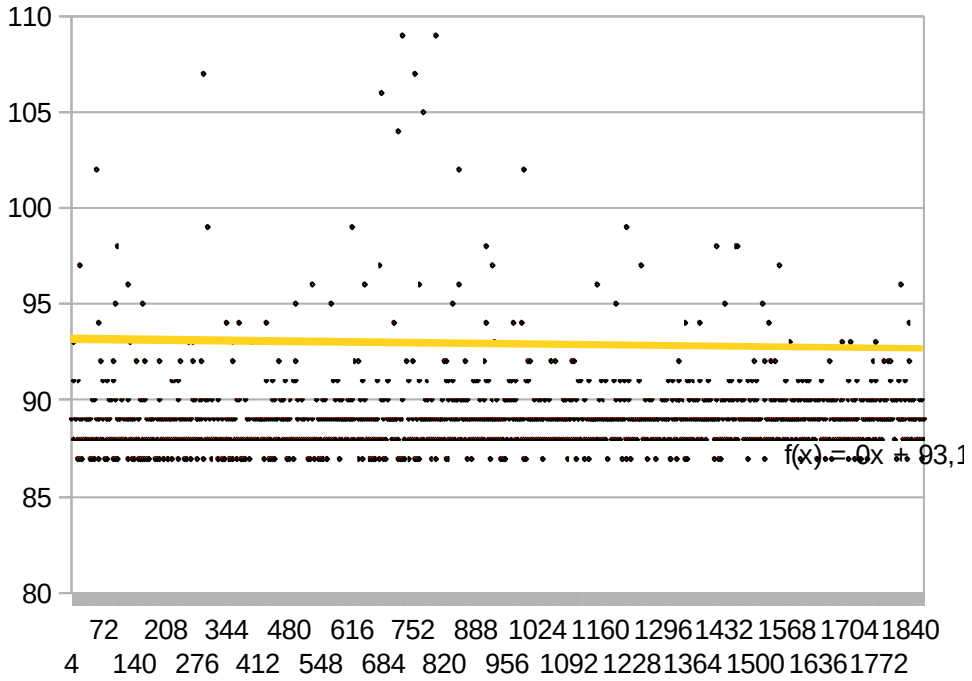
- Utilisation d'un demi ping
- Mesure l'aller
- Problème:
  - Synchronisation entre les 2 machines ?
    - Demi-Ping: 25ms
    - Décalage de moins de 2,5ms (10% d'erreur maximum)
- Idem pour le retour

Comment synchroniser ?

- Synchronisation NTP
  - Erreur NTP de 10ms
  - Erreur relative entre 10% et 50%
  - Non acceptable

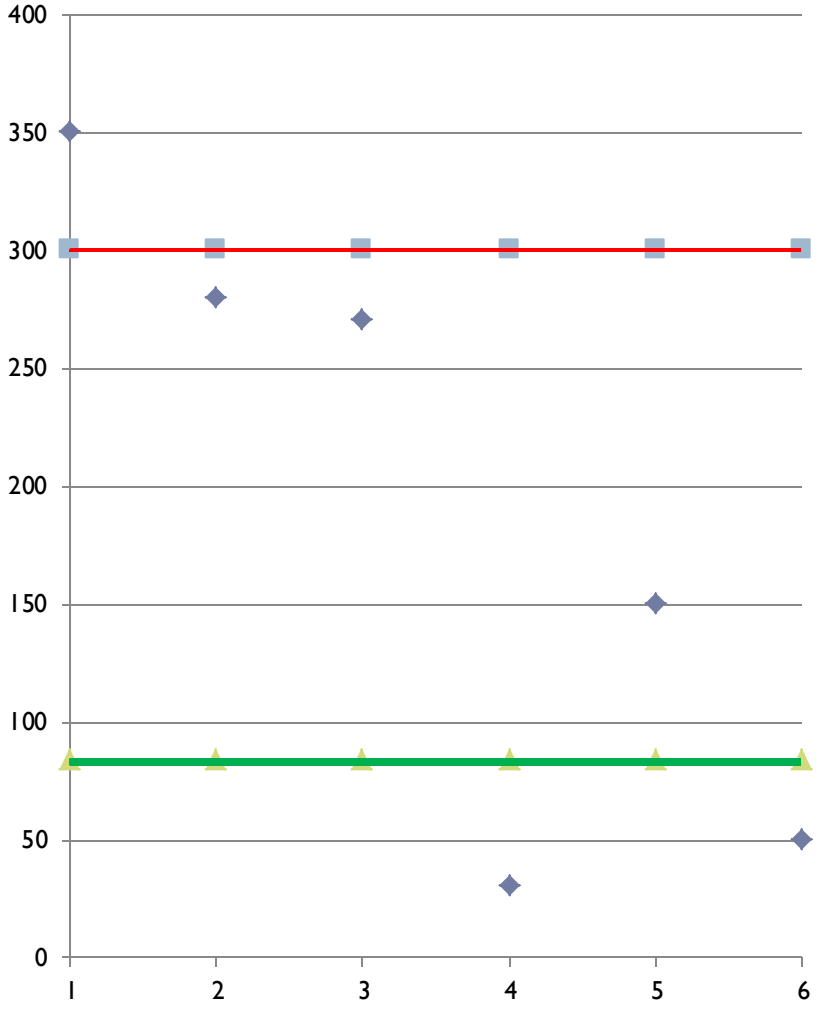
Sans synchronisation !

- Utilisation du délai relatif
- Mesure:
  - *ts_site1 – ts_site2*
  - *ts: timestamp*
- Comparaison avec un minimum local
  - 10 min
- Intérêt:
  - l'évolution relative
- Dérive des horloges
  - 40 min: 0,5ms
  - Erreur relative: 0,4%



Formule de détection de saturation

- On a donc détection si, parmi les 6 derniers échantillons :
  - 1 échantillon est d'une valeur *peak* supérieure à TRIGGER
  - 3 autres échantillons sont supérieurs à 0.25*MOYENNE(*peak*, *peak*, *TRIGGER*)

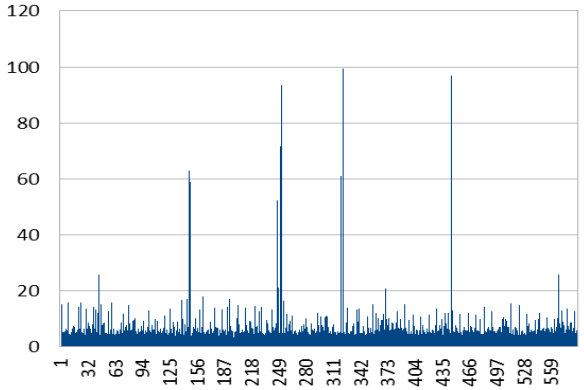


Influence du réseau radio

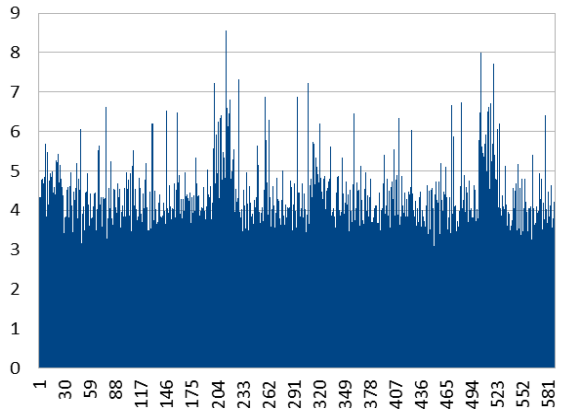
Influence du réseau radio

- Vérifier l'évolution du ping au cours du temps selon le type de réseau radio
- But: vérification d'une hypothèse
  - Ecart type: 8.66

- => Si le réseau radio est de mauvaise qualité
  - Fausse les mesures
  - Ecart type: 0.79



2,4GHz
Nok



5GHz
Ok

Conclusion de l'étude expérimentale

- Ping
  - Détection de saturation d'un lien ADSL
  - Non connaissance du sens de la saturation
- Demi-délai relatif
  - Détection de saturation d'un lien ADSL (dans les deux sens)
  - Formule évitant les faux positifs trouvée
- Réseau radio de 5GHz
  - Hypothèse valide (réseau radio non influent sur la mesure)
- Réseau radio de 2.4GHz
  - Hypothèse non valide (faux positifs possibles)

Les approches abandonnées

Les approches abandonnées de l'agrégation

- Ajout de métrique à B.A.T.M.A.N.
  - B.A.T.M.A.N.: Protocole de routage dynamique (niveau 3)
- MAIS
  - Sujet de recherche en cours
  - Expérimentations pratiques (tetaneutral.net)
    - Instable
    - Non fonctionnelle sur de grand réseau
    - Communication non utile trop importante

- Extension de l'outil linkagreg
  - Outil d'agrégation développé par Fernando Alves
  - Ouverture de plusieurs sockets UDP à travers un tunnel
- MAIS
  - Manque de documentation de l'outil
  - Langage C
    - Développement en temps raisonnable de prototype pas possible
    - Adaptation de l'algorithme de répartition couteux en temps

La solution développée

Petite pause camescope

L'approche choisie

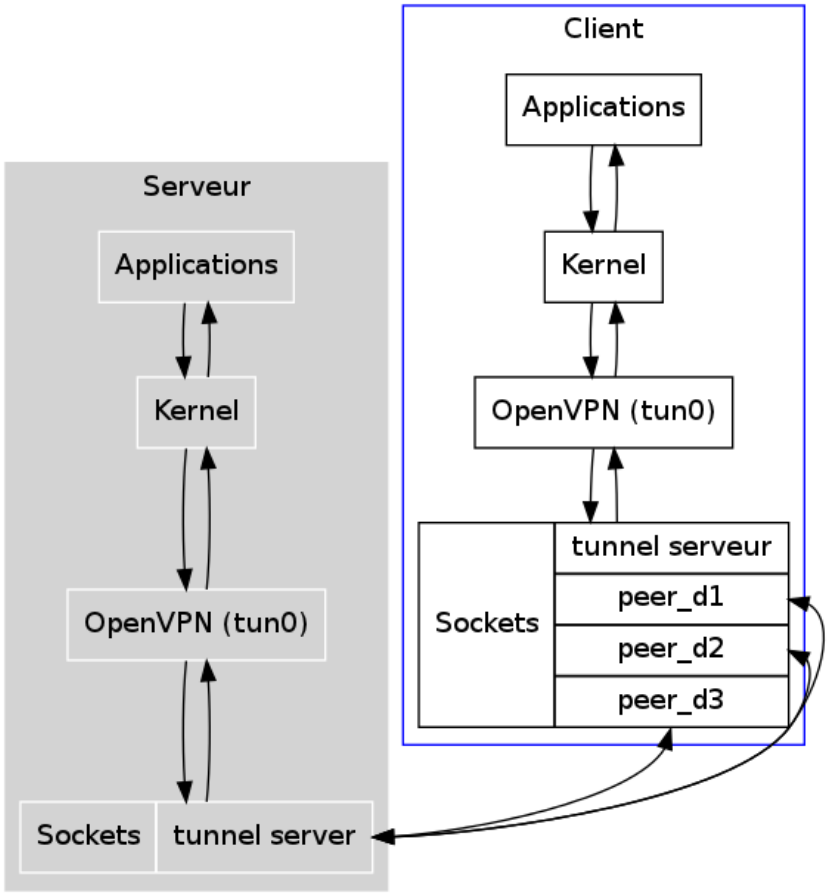
- Approche s'articulant autour de 3 axes:
  - Tunnel
  - Agrégation
  - Routage
- Utilisation de plusieurs outils et de scripts (en python)
  - OpenVPN
    - Tunnel
  - Multi.py
    - Agrégation
    - Algorithme de répartition de charge
      - Répartition de charge pondérée: delta_half_trip_time.py
  - Ip route 2
    - Routage

Les fonctionnalités développées

- Dispatchage
  - sur plusieurs liens
  - sélection pondérée (par les capacités des liens)
- Détection de la capacité et de la variation opportuniste d'un lien
- Script d'initialisation
- Ne pas nécessiter d'avoir plusieurs IP publiques sur le serveur
- Fonctionnement derrière un NAT
  - Exemple: derrière une livebox
- Fonctionne sous Linux (!)

La partie agrégation « multi.py »

- Fonctionnement:
  - Création d'interfaces virtuelles via openVPN
  - Distribution de la charge sur plusieurs tunnels UDP
  - Algorithme de répartition de charge aléatoire pondéré



Détection de saturation du lien et ajustement des pondérations

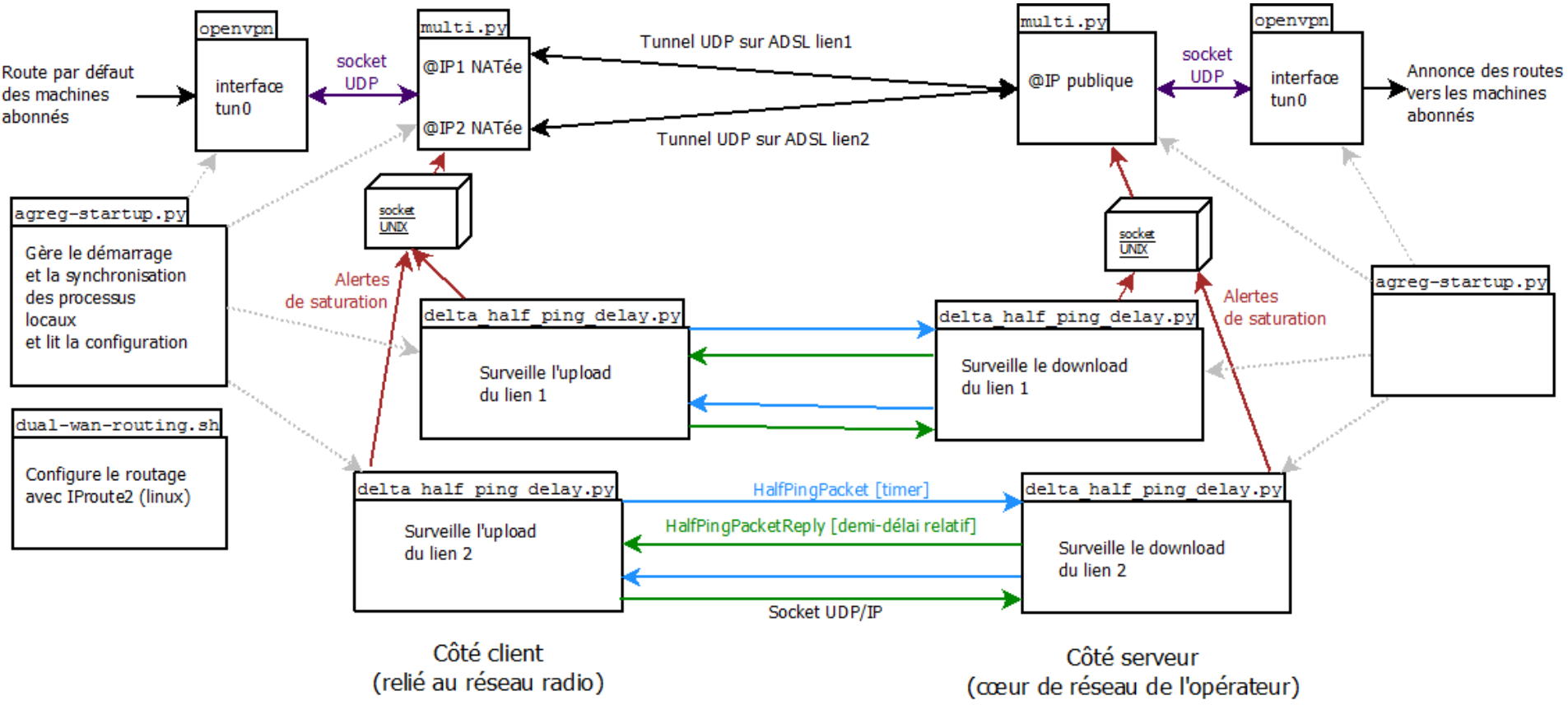
- Formule de détection (variation demi-ping)
  - delta_half_trip_time.py
- Ajustement des pondérations
  - selon le maximum de débit récent (10s)
  - multi.py
- Initialisation des pondérations (détection de la capacité d'un lien)
  - Dissymétrie (facteur de 1000)
  - MaJ d'un lien
  - Puis de l'autre

```
 New weights are [100000000, 100000000]
  received report : SAT TX OVH_ADSL, bw was: 1181kb/s
  New weights are [100000000, 151200]
  received report : SAT TX FDN_ADSL, bw was: 823kb/s
  New weights are [105352, 151200]
...
```

Architecture modulaire

- Plusieurs processus
  - Communication par socket
- Possibilité: développé de la partie critique en C
  - But: gagner en performance
- Utilisation d'openVPN pour le tunnel
  - Pas la peine de réinventer la roue
  - Solution robuste

Architecture (schéma)



Mesure de performances

ping <gateway>
iperf –c –P7 <gateway>

	Ping (ms)	Upload (Mbps)	Download (Mbps)
OVH	60,81	0,855	6,93
FDN	58,67	0,711	7,11
OVH+FDN	N/A	1,53	14,04
TUN (RR)	62,32 (-4%)	1,23 (-19%)	8,90 (-36%)
TUN (BAL)	62,35 (-4%)	1,33 (-12%)	10,3 (-27%)

Commentaires sur les performances

- Ping non impacté
- Upload satisfaisant
- Download décevant
- Occupation CPU (20% en download)
- Validation de la détection par demi-ping

Démo

Conclusions

Bilan

- Production
  - « Nouveaux » résultats intéressants
    - Détection par demi-délai relatif
    - Pondération dynamique
  - Solution encore expérimentale mais fonctionnelle
  - Intégration des mécanismes expérimentés à un autre projet (MLVPN ?)

- Ressenti
  - Mise en œuvre de notions de réseaux avancées
  - Et classiquement, on a appris plein de choses…
  - Difficulté pour les tests

Perspectives

- Fonctionnalités
  - Détection de coupure/rétablissement d'un lien
  - Prise en compte du taux de paquets perdus
  - Plusieurs socket UDP par lien (QoS)
  - Chiffrement et authentification (openVPN)
  - Meilleur gestion des faux positifs
- Performances
  - Consommation CPU trop importante
    - Destination de matériel embarqué
    - => Ré-écriture d'une partie en C
  - Abandon de l'aléatoire pour une méthode déterministe pondérée
  - Augmenter drastiquement la MTU sur l'interface virtuelle ?
    - Réduit le passage des paquets en espace utilisateur

- Merci !

Questions ?