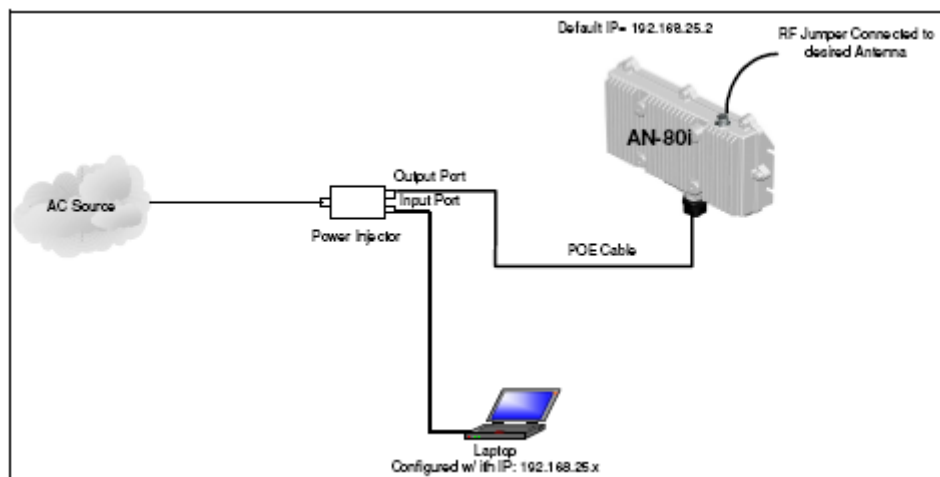
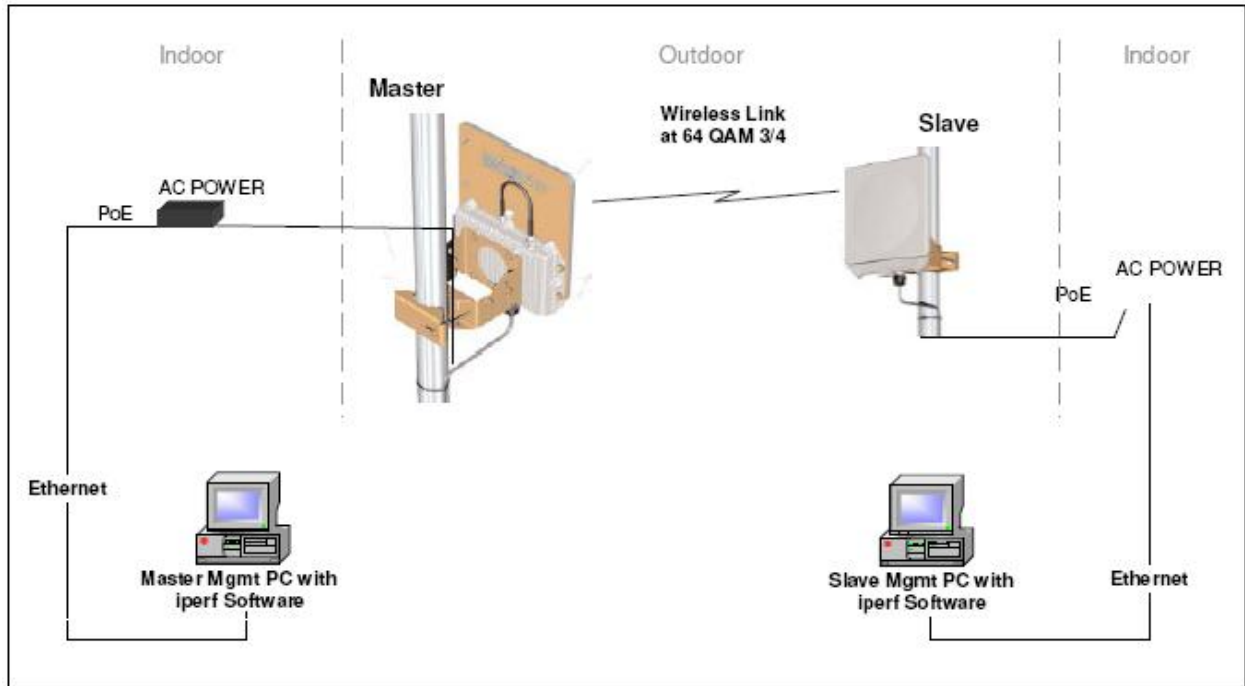


Validation de débit sur les liaisons Wirake TDD Wi100 Wi 200-C5,Wi300

En dehors d'équipements de mesure dédiés, le test peut se réaliser avec Iperf mais il faut utiliser deux PC et que toutes les applications soient fermées sur ces PC.

Les performances seront très liées aux performances des PC utilisés.

Notez que l'on ne va pas tester sur les adresses des radio ou des interfaces mais les adresses des PC actifs disposant du soft Iperf et installés aux extrémités de la liaison.



NOTA : Le synoptique est identique pour les liaisons Full Duplex symétriques et simultanées ; la notion de Master et de Slave des systèmes TDD ne sera pas en prendre en compte pour les liaisons de Type FH Full Duplex.

Télécharger IPERF : <http://iperf.fr/index.php/topic,3804.0.html>

 Iperf pour Windows 2000, XP, 2003, Vista, 7 :

- [Iperf 2.0.5-2](#)

 Iperf pour Linux 32 bits (i386) :

- [Iperf 2.0.5 - paquet DEB](#)

 Iperf pour Linux 64 bits (AMD64) :

- [Iperf 2.0.5 - paquet DEB](#)

 Iperf pour MacOS X :

- [Iperf 2.0.5 \(CPU Intel\)](#)

 Iperf pour Sun Solaris :

- [Iperf 2.0.4 pour Solaris 10 x86](#)

Le test se fait **toujours d'un Master vers un Slave** sur les équipement TDD radio e qui n'est pas le cas sur des liens cuivre ou fibre optique ou bien en FH symétrique Duplex. Par exemple la gamme des FH de type Wi20-Wi200-S et C24 -Wi700

Pour éviter de fausses mesures en cas de liaison limite et ou instable décocher la case "Adaptive modulation "

Ouvrir une Commande prompt cmd sur le PC connecté au Master

La ligne de commande valable pour une liaison a 108 Mbits annoncés est :

```
iperf -c <ip_address> -d -P3 -w 64k -t 30 -f k
```

ip_address est l'adresse du PC distant, attention aux erreurs de frappe.

Le test prend environ 30 secondes

le résultat marqué « SUM » doit être non inférieur à 72 Mbits et dépend du PC pour une liaison qui se déclare a 108 Mbits l'on doit obtenir 90 Mbits en test Iperf.

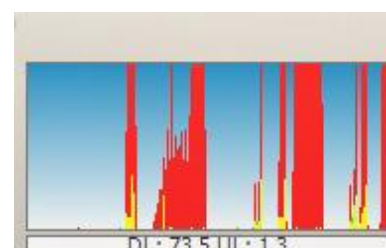
Pour faire un test global du réseau il faut télécharger un fichier distant par exemple a Lyonix ou ailleurs et contrôler le temps mis à le télécharger afin de « voir » le débit Global, l'idéal est de lancer plusieurs sessions avec la même machine sur plusieurs Fichiers afin de bien la saturer; on peut aussi voir le débit Up et Down avec l'outil Bit Meter et additionner les deux données pour connaître le débit réel supporté.

Téléchargez Bit Meter qui vous donnera le débit total avec précision :

<http://logiciel.codes-sources.com/logiciels/BitMeter-II-560.aspx>

Lz débit sur cet exemple est ici exprimé en Ko le multiplier par 8 pour obtenir la valeur en Mbps. dans ce cas le débit dans le réseau pour le téléchargement du fichier BitMeter est de 73,5+1,3= 74,8 Ko x 8 = 598,4 kilobits. Ou bien configurer le Bit Meter en affichage Kilobits ou Megabits

Autres informations sur : <http://lafibre.info/index.php/topic,2077.0.html>



DESCRIPTION UTILISATION D'IPERF

Fonctionnalités générales :

IPerf est un logiciel de mesure de performance réseau, disponible sur de nombreuses plateformes (Linux, BSD, Mac, Windows...). Il se présente sous la forme d'une ligne de commande à exécuter sur deux machines disposées aux extrémités du réseau à tester.

Il permet de mesurer la bande passante, la latence, la gigue et la perte de datagrammes.

Iperf doit être lancé sur deux machines se trouvant de part et d'autre du réseau à tester. La première machine lance Iperf en "mode serveur" (avec l'option -s), la seconde en "mode client" (option -c). Par défaut le test réseau se fait en utilisant le protocole TCP (mais il est également possible d'utiliser le mode UDP avec l'option -u).

Autres fonctionnalités :

Supporte IPV6 (utiliser l'option -V pour spécifier une adresse IPv6) et le multicast. Permet de tester des réseaux WiFi.

Permet de détecter des problèmes sur des câbles réseaux dans un LAN en mesurant des performances asymétriques d'un poste de travail vers plusieurs autres machines du même réseau.

Interopérabilité :

Iperf n'est pas interopérable avec d'autres logiciels. Il n'est pas conseillé de faire des tests avec des versions différentes d'iperf aux deux extrémités.

Contexte d'utilisation :

Test de réseau filaire ou sans fil, d'entreprise ou domestique.

Iperf est utile dans de nombreux cas. Lors de la résolution d'un problème dont le réseau est une cause possible, un test Iperf judicieux peut rapidement mettre en évidence s'il y a ou non réellement un problème réseau.

Exemples :

- si un utilisateur se plaint des performances de sa machine vis-à-vis d'une application hébergée sur un serveur, un test iperf TCP entre sa machine et le serveur permettra de déterminer si le problème est lié au réseau ou à la couche applicative.
- si un utilisateur se plaint d'une lenteur généralisée des accès de sa machine, des tests iperf mettront en évidence des éventuels problèmes réseaux. Si les résultats de tests effectués dans les deux sens montrent une forte asymétrie en termes de performances ou de pertes avec plusieurs machines, c'est souvent le signe d'un câble réseau défectueux.

Limitations, difficultés, fonctionnalités importantes non couvertes :

Il est important d'utiliser Iperf à bon escient, la valeur des résultats est proportionnelle à la pertinence de la mesure par rapport au problème qu'on veut résoudre. Si vous voulez résoudre un problème entre une machine A et une machine B, mesurer les performances entre la machine A et une machine C apporte peu d'informations si C n'est pas directement sur le chemin de routage ou de commutation de A et B.

Lors de tests dans les deux sens, il vaut mieux utiliser l'option "-r" (tradeoff) plutôt que "-d" (duplex). En effet, lors d'un test full duplex à de hauts débits, la CPU des machines aux deux extrémités est très sollicitée et fausse largement la mesure. Avec l'option "-r", le test est d'abord fait dans un sens, puis il est lancé dans l'autre sens. Le résultat est plus pertinent.

Attention à la bande passante en UDP : si vous émettez un flux de 1Gbits/s vers une destination à l'extérieur de votre réseau, votre liaison WAN va devoir écouler ce flux. Si le débit de votre liaison est inférieur (c'est très souvent 10 fois moins), vous allez couper l'accès à Internet sur tout le site.

ENVIRONNEMENT DU LOGICIEL

Distributions dans lesquelles ce logiciel est intégré :

Packages existants pour Fedora, RedHat, CentOS, Debian, Mandriva.

Plates-formes :

Linux

FreeBSD

Irix

MacOS X

Windows (installer kperf)

OpenBSD

Solaris

Logiciels connexes :

Jperf: Interface graphique en java pour IPerf: <http://sourceforge.net/projects/jperf>

Autres logiciels aux fonctionnalités équivalentes :

ttcp : beaucoup plus ancien et limité

ENVIRONNEMENT DE DEVELOPPEMENT

Type de structure associée au développement :

Equipe de développement de "University of Illinois".

Eléments de pérennité :

La dernière version de lperf date de Avril 2008.

ENVIRONNEMENT UTILISATEUR

Liste de diffusion ou de discussion, support et forums :

FAQ: <http://sourceforge.net/mailarchive/forum.php?forum...>

Liste de diffusion en anglais : iperf-users@lists.sourceforge.net

Archive de la liste: <http://archive.ncsa.uiuc.edu/lists/iperf-users/>

Liste de diffusion en français : <http://www.services.cnrs.fr/wws/info/perf-reseau>

cette liste n'est pas centrée sur Iperf mais a pour objectif de traiter toutes les questions relatives aux problèmes des performances réseau.

Divers (astuces, actualités, sécurité) :

Dans les exemples ci-dessous, nous allons considérer que nous allons utiliser deux machines nommées C (avec comme adresse IP: IPC) et S (avec comme adresse IP: IPS).

Exemple pour mesurer la bande passante disponible entre S et C

Attention, cette méthode mesure la bande passante au moment du test. Ce dernier dure par défaut 10 secondes et utilise le protocole TCP sur le port 5001.

```
Sur la machine S: # iperf -s
Sur la machine C: # iperf -c IPS
Résultat (à lire sur la machine S):
_____

Server listening on TCP port 5001TCP window size: 56.0 KByte (default)
_____

[ 6] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 54334
[ 6] 0.0-10.0 sec 112 MBytes 93.7 Mbits/sec
```

Exemple pour générer un débit réseau entre C et S

On génère ici un flux en utilisant le protocole UDP et en fixant la bande passante à 1 Megabits par seconde. Le test dure par défaut 10 secondes.

Il est possible de choisir l'unité de mesure de débit avec l'option -b et en collant les lettres suivantes aux débits :

- * b: bits par seconde
- * k: kilobits par seconde
- * m: megabits par seconde
- * g: gigabits par seconde

Pour un débit en octets par seconde, il faut utiliser ces lettres en majuscule.

```
Sur la machine S: # iperf -s -u
Sur la machine C: # iperf -c IPS -u -b 4m
Résultat (à lire sur la machine S):
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 41.1 KByte (default)
-----
[ 5] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 49617
[ 5] 0.0-10.0 sec 4.77 MBytes 4.00 Mbits/sec 0.066 ms 0/ 3403 (0%)
```

Exemple pour générer un débit réseau entre C et S pendant 10 heures

Il peut être utile de générer un flux réseau plus long pour tester par exemple une liaison Internet pendant les heures d'utilisation. Nous allons donc utiliser l'option -t pour fixer la durée du test précédent à 10 heures (10*3600=36000 secondes).

```
Sur la machine S: # iperf -s -u
Sur la machine C: # iperf -c IPS -u -b 4m -t 36000
Résultat (à lire sur la machine S):
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 41.1 KByte (default)
-----
[ 5] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 49617
[ 5] 0.0-36000.0 sec 4.77 MBytes 4.00 Mbits/sec 0.066 ms 0/ 999403 (0%)
```

Il est également possible d'ajouter l'option `-i 3600` pour avoir un rapport intermédiaire toutes les heures ($1*3600=3600$ secondes).

```
Sur la machine S: # iperf -s -u
Sur la machine C: # iperf -c IPS -u -b 4m -t 3600
Résultat (à lire sur la machine S):
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 41.1 KByte (default)
-----
[ 5] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 49617
[ 5] 0.0-3600.0 sec 4.77 MBytes 4.00 Mbits/sec 0.066 ms 0/ - (0%)
[ 5] 3600.0-7200.0 sec 4.77 MBytes 4.00 Mbits/sec 0.066 ms 0/ - (0%)...
[ 5] 0.0-36000.0 sec 4.77 MBytes 4.00 Mbits/sec 0.066 ms 0/ 999403 (0%)
```

Exemple pour générer 2 flux réseau entre S et C

Il est parfois utile de générer plusieurs flux UDP simultanément pour simuler une application. IPerf permet cela grâce à l'option `-P` et en donnant le nombre de flux à générer. L'exemple suivant génère 4 flux TCP entre S et C (simulation d'un serveur Web par exemple).


```
Sur la machine S: # iperf -s
Sur la machine C: # iperf -c IPS -P 4
Résultat (à lire sur la machine S):
-----
Server listening on TCP port 5001
TCP window size: 56.0 KByte (default)
-----
[ 6] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 64978
[ 7] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 64979
[ 8] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 64980
[ 9] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 64981
[ 7] 0.0-10.0 sec 28.0 MBytes 23.5 Mbits/sec
[ 8] 0.0-10.0 sec 28.0 MBytes 23.4 Mbits/sec
[ 9] 0.0-10.0 sec 28.1 MBytes 23.5 Mbits/sec
[ 6] 0.0-10.0 sec 28.1 MBytes 23.5 Mbits/sec
[SUM] 0.0-10.0 sec 112 MBytes 93.8 Mbits/sec
```

Exemple pour optimiser une connexion TCP entre S et C

Le protocole TCP, bien que capable pour s'adapter aux réseaux large bande, a été conçu lorsque les débits étaient beaucoup moins importants. Les valeurs par défaut des fenêtres TCP (taille des paquets envoyés dans des trames TCP) ne sont pas forcément adaptées aux réseaux actuels. IPerf permet de jouer avec la taille de ces fenêtres avec l'option `-w`. L'exemple suivant génère un flux TCP avec une taille de fenêtre de 130 kilo octets.

```
Sur la machine S: # iperf -s
Sur la machine C: # iperf -c IPS -w 128k
Résultat (à lire sur la machine S):
-----
Client connecting to 192.168.29.1, TCP port 5001
TCP window size: 128 KByte (WARNING: requested 130 KByte)
-----
[ 3] local 192.168.29.157 port 65066 connected with 192.168.29.1 port 5001
[ 3] 0.0-10.0 sec 112 MBytes 93.5 Mbits/sec
```

Exemple pour découvrir la taille du MTU entre S et C

Le MTU est la taille maximale du paquet pouvant être transmis sur la couche réseau sans être segmenté. La découverte de cette valeur peut être utile à l'optimisation de votre réseau et des applications qui tourne dessus. IPerf permet d'obtenir cette valeur grâce à l'option -m (à lancer sur le serveur).

```
Sur la machine S: # iperf -s -m
Sur la machine C: # iperf -c IPS
Résultat (à lire sur la machine S):
-----
Client connecting to 192.168.29.1, TCP port 5001
TCP window size: 56.0 KByte (default)
-----
[ 3] local 192.168.29.157 port 65066 connected with 192.168.29.1 port 5001
[ 3] 0.0-10.0 sec 112 MBytes 93.5 Mbits/sec[ 3] MSS size 1448 bytes (MTU
1500 bytes, ethernet)
```

Exemple pour tester un flux de type VoIP entre C et S

Les paquets de type voix sur IP ont les caractéristiques suivantes : protocole UDP et taille des paquets petite (bien inférieure au MTU). Le meilleur moyen de tester un flux de type VoIP avec IPerf est d'utiliser les options `-l` (taille du datagramme) et `-w` (taille maximale du buffer recevant les datagrammes) en fixant une valeur de datagramme inférieure à celle du buffer.

```
Sur la machine S: # iperf -s -u -l 32k -w 128k -i 1
Sur la machine C: # iperf -c IPS -u -b 1m -l 32k -w 128k
Résultat (à lire sur la machine S):
-----
Server listening on UDP port 5001
Receiving 32768 byte datagrams
UDP buffer size: 128 KByte
-----
[ 3] local 192.168.29.157 port 5001 connected with 192.168.29.125 port
32778
[ 3] 0.0- 1.0 sec 96.0 KBytes 786 Kbits/sec 0.003 ms 0/ 3 (0%)
[ 3] 1.0- 2.0 sec 128 KBytes 1.05 Mbits/sec 0.008 ms 0/ 4 (0%)
[ 3] 2.0- 3.0 sec 128 KBytes 1.05 Mbits/sec 0.017 ms 0/ 4 (0%)
[ 3] 3.0- 4.0 sec 128 KBytes 1.05 Mbits/sec 0.021 ms 0/ 4 (0%)
[ 3] 4.0- 5.0 sec 128 KBytes 1.05 Mbits/sec 0.023 ms 0/ 4 (0%)
[ 3] 5.0- 6.0 sec 96.0 KBytes 786 Kbits/sec 0.022 ms 0/ 3 (0%)
[ 3] 6.0- 7.0 sec 128 KBytes 1.05 Mbits/sec 0.152 ms 0/ 4 (0%)
[ 3] 7.0- 8.0 sec 128 KBytes 1.05 Mbits/sec 0.142 ms 0/ 4 (0%)
[ 3] 8.0- 9.0 sec 128 KBytes 1.05 Mbits/sec 0.115 ms 0/ 4 (0%)
[ 3] 9.0-10.0 sec 128 KBytes 1.05 Mbits/sec 0.098 ms 0/ 4 (0%)
[ 3] 0.0-10.5 sec 1.25 MBytes 1.00 Mbits/sec 0.116 ms 0/ 40 (0%)
```

Remarque : bien que Iperf soit disponible sur de nombreuses plate-forme (Linux, BSD, Mac, Windows), l'option `-l` ne fonctionne pas toujours quand vous utiliser des OS différents entre le client et le serveur.

Exemple pour utiliser IPerf sur un port différent

Par défaut, Iperf utilise le numéro de port 5001 (TCP et/ou UDP). Selon votre configuration (notamment au niveau des ACL des routeurs/firewalls), il peut être utile d'utiliser un autre port, pour cela, il faut passer par l'option -p. L'exemple suivant permet de générer un flux réseau TCP entre S et C sur le port 80 (port Web standard).

```
Sur la machine S: # iperf -s -p 80
Sur la machine C: # iperf -c IPS -p 80
Résultat (à lire sur la machine S): _____
Server listening on TCP port 80
TCP window size: 56.0 KByte (default)
_____
[ 6] local 192.168.29.1 port 5001 connected with 192.168.29.157 port 54334
[ 6] 0.0-10.0 sec 112 MBytes 93.7 Mbits/sec
```

IPerf en multicast

Iperf peut fonctionner en mode multicast (-B). Il faut le lancer de la manière suivante:

```
Sur le serveur: $ iperf -s -u -B 225.0.1.2
Sur le client: $ iperf -c 225.0.1.2 -u -b 3M
```

Cela génère un flux multicast UDP (sur l'adresse 225.0.1.2 de 3 Mb/sec.)

[Iperf](#) est un outil pour mesurer la bande passante et la qualité d'un lien réseau. Ce dernier est délimité par deux machines sur lesquelles est installé Iperf.

La qualité d'un lien est déterminée principalement par les facteurs suivants:

- Latence (temps de réponse ou RTT): peut être mesurée à l'aide d'un ping.
- Gigue ou jitter en anglais (variation de la latence): peut être mesurée par un test Iperf UDP.
- Perte de paquet: peut être mesurée avec un test Iperf UDP.

Quant à la bande passante, elle est mesurée par des tests TCP.

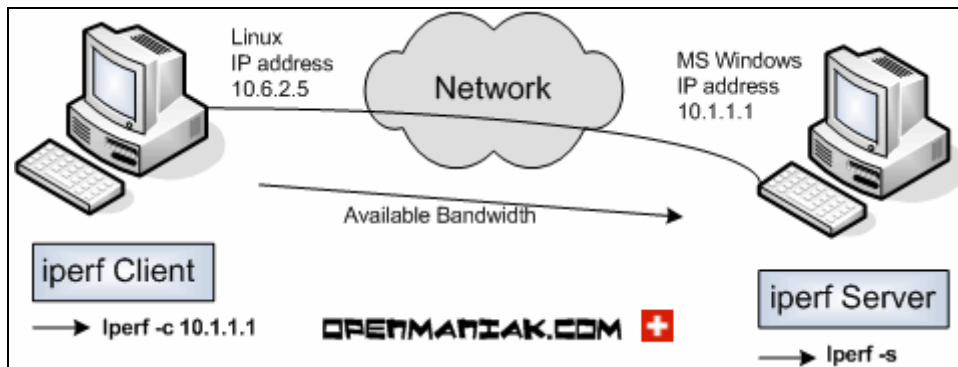
Pour être simple la différence entre TCP (Transmission Control Protocol) et UDP (User Datagram Protocol) est que TCP utilise des processus pour vérifier que les paquets sont correctement envoyés au receveur. ceci n'est pas le cas pour UDP où les paquets sont envoyés sans aucune vérification mais avec l'avantage d'être plus rapide que TCP.

Iperf utilise les différentes propriétés de TCP et d'UDP pour fournir des statistiques sur des liens réseaux.

Iperf peut être installé très facilement sur n'importe quel système UNIX/Linux ou Microsoft Windows. Un hôte doit être configuré en tant que client et l'autre en tant que serveur.

Voici un petit labo où Iperf est installé sur une machine Linux et Microsoft Windows.

Linux est utilisé comme client Iperf et Windows comme serveur Iperf. Bien sûr, il est aussi possible d'utiliser deux machines Linux.



Tests Iperf:

défaut	Paramètres par défaut	-p, -t, -i	Port, temps et intervalle
-f	Formatage des données	-u, -b	Tests UDP, configuration bande passante
-r	Bande passante bidirectionnelle	-m	Affichage de la taille de segment maximale
-d	Bande passante bidirectionnelle simultanée	-M	Configuration de la taille de segment maximale
-w	Taille de la fenêtre TCP	-P	Tests parallèles
-h	Aide		

Jperf:

[Pas d'argument](#)
[-d](#)
[-u, -b](#)

Paramètres par défaut
 Bande passante bidirectionnelle simultanée
 Tests UDP, configuration bande passante

■ Paramétrage Iperf par défaut:
 Voir la section "[Jperf](#)"

Par défaut, le client Iperf se connecte au serveur Iperf sur le port TCP 5001 et la bande passante affichée par Iperf est celle du client au serveur.

Si vous voulez utiliser des tests UDP, utilisez l'argument [-u](#).

Les arguments client Iperf [-d](#) et [-r](#) mesurent les bandes passantes bidirectionnelles. (Voir plus loin dans ce tutorial)

→ Côté client:

```
#iperf -c 10.1.1.1
```

```
-----
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16384 Byte (default)
-----
```

```
[ 3] local 10.6.2.5 port 33453 connected with 10.1.1.1 port 5001
[ 3] 0.0-10.2 sec 1.26 MBytes 1.05 Mbits/sec
```

→ Côté serveur:

```
#iperf -s
```

```
-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
-----
```

```
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 33453
[ ID] Interval Transfer Bandwidth
[852] 0.0-10.6 sec 1.26 MBytes 1.03 Mbits/sec
```

■ Formatage des données: (argument -f)

L'argument -f permet d'afficher les résultats selon le format désiré: bits(b), bytes(B), kilobits(k), kilobytes(K), megabits(m), megabytes(M), gigabits(g) ou gigabytes(G).

Généralement, les mesures de bande passante sont affichées en bits/sec (ou Kilobits/sec, etc ...) et une quantité de données est affichée en octets (or Kilobytes, etc ...).

Pour mémoire, 1 octet (byte en anglais) est égal à 8 bits et dans le domaine informatique 1 kilo est égale à 1024 (2¹⁰).

Par exemple: 100'000'000 octets ne sont pas égaux à 100 Mbytes mais à 100'000'000/1024/1024 = 95.37 Mbytes.

→ Côté client:

```
#iperf -c 10.1.1.1 -f b
```

```
-----
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16384 Byte (default)
-----
```

```
[ 3] local 10.6.2.5 port 54953 connected with 10.1.1.1 port 5001
[ 3] 0.0-10.2 sec 1359872 Bytes 1064272 bits/sec
```

→Côté serveur:

```
#iperf -s
```

```
-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
-----
```

```
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 33453
[ ID] Interval      Transfer    Bandwidth
[852] 0.0-10.6 sec  920 KBytes  711 Kbits/sec
```

[Haut de la page](#)

■ Mesure de la bande passante bidirectionnelle: (argument -r)

Le serveur Iperf se connecte en retour sur le client permettant la mesure de la bande passante bidirectionnelle. Par défaut, seule la bande passante du client au serveur est mesurée. Si vous voulez mesurer la bande passante bidirectionnelle de manière simultanée, utilisez l'argument -d. (Voir le test suivant)

→Côté client:

```
#iperf -c 10.1.1.1 -r
```

```
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
```

```
-----
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16.0 KByte (default)
-----
```

```
[ 5] local 10.6.2.5 port 35726 connected with 10.1.1.1 port 5001
[ 5] 0.0-10.0 sec  1.12 MBytes  936 Kbits/sec
[ 4] local 10.6.2.5 port 5001 connected with 10.1.1.1 port 1640
[ 4] 0.0-10.1 sec  74.2 MBytes  61.7 Mbits/sec
```

→Côté serveur:

```
#iperf -s
```

```
-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
-----
```

```
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 54355
[ ID] Interval      Transfer    Bandwidth
[852] 0.0-10.1 sec  1.15 MBytes  956 Kbits/sec
```

```
-----
Client connecting to 10.6.2.5, TCP port 5001
TCP window size: 8.00 KByte (default)
-----
```

```
[824] local 10.1.1.1 port 1646 connected with 10.6.2.5 port 5001
[ ID] Interval      Transfer    Bandwidth
[824] 0.0-10.0 sec  73.3 MBytes  61.4 Mbits/sec
```

[Haut de la page](#)

- Mesure de la bande passante bidirectionnelle simultanée: (argument -d)
Voir la section "[Iperf](#)".

Pour mesurer les bandes passantes bidirectionnelles simultanées, utilisez l'argument -d. Si vous voulez tester les bandes passantes séquentiellement, utilisez l'argument -r (Voir le test précédent).
Par défaut (c'est-à-dire sans les arguments -r ou -d), seule la bande passante du client au serveur est mesurée.

→ Côté client:

```
#iperf -c 10.1.1.1 -d
```

```
-----  
Server listening on TCP port 5001  
TCP window size: 85.3 KByte (default)  
-----
```

```
-----  
Client connecting to 10.1.1.1, TCP port 5001  
TCP window size: 16.0 KByte (default)  
-----
```

```
[ 5] local 10.6.2.5 port 60270 connected with 10.1.1.1 port 5001  
[ 4] local 10.6.2.5 port 5001 connected with 10.1.1.1 port 2643  
[ 4] 0.0-10.0 sec 76.3 MBytes 63.9 Mbits/sec  
[ 5] 0.0-10.1 sec 1.55 MBytes 1.29 Mbits/sec
```

→ Côté serveur:

```
#iperf -s
```

```
-----  
Server listening on TCP port 5001  
TCP window size: 8.00 KByte (default)  
-----
```

```
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 60270  
-----
```

```
-----  
Client connecting to 10.6.2.5, TCP port 5001  
TCP window size: 8.00 KByte (default)  
-----
```

```
[800] local 10.1.1.1 port 2643 connected with 10.6.2.5 port 5001  
[ ID] Interval      Transfer    Bandwidth  
[800] 0.0-10.0 sec 76.3 MBytes 63.9 Mbits/sec  
[852] 0.0-10.1 sec 1.55 MBytes 1.29 Mbits/sec
```

[Haut de la page](#)

- Taille de la fenêtre TCP: (argument -w)

La taille de la fenêtre TCP correspond aux données qui peuvent être mise en tampon pendant une connexion sans la validation du receveur.

Elle est comprise entre 2 et 65535 bytes.

Sur les systèmes Linux, quand on spécifie une taille de fenêtre TCP avec l'argument -w, le noyau alloue le double de la valeur indiquée.

→ Côté client:

```
#iperf -c 10.1.1.1 -w 2000
```

```
WARNING: TCP window size set to 2000 bytes. A small window size  
will give poor performance (une petite taille de fenêtre donnera de faible performances). See the Iperf  
documentation.
```

```
-----  
Client connecting to 10.1.1.1, TCP port 5001  
TCP window size: 3.91 KByte (WARNING: requested 1.95 KByte)  
-----
```

```
[ 3] local 10.6.2.5 port 51400 connected with 10.1.1.1 port 5001  
[ 3] 0.0-10.1 sec 704 KBytes 572 Kbits/sec
```

→ Côté serveur:


```
#iperf -s -w 4000
```

```
-----
Server listening on TCP port 5001
TCP window size: 3.91 KByte
-----
```

```
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 51400
[ ID] Interval      Transfer    Bandwidth
[852] 0.0-10.1 sec  704 KBytes  570 Kbits/sec
```

[Haut de la page](#)

■ Port de communication (-p), temps (-t) et intervalle (-i):

Le port de communication du serveur Iperf peut être changé avec l'argument -p. Il doit être configuré sur le client et le serveur avec la même valeur, par défaut le port TCP 5001.

L'argument -t spécifie la durée du test en seconde, par défaut 10 secondes.

L'argument -i indique l'intervalle en seconde entre les rapports périodiques de bande passante.

→ Côté client:

```
#iperf -c 10.1.1.1 -p 12000 -t 20 -i 2
```

```
-----
Client connecting to 10.1.1.1, TCP port 12000
TCP window size: 16.0 KByte (default)
-----
```

```
[ 3] local 10.6.2.5 port 58316 connected with 10.1.1.1 port 12000
[ 3] 0.0- 2.0 sec  224 KBytes  918 Kbits/sec
[ 3] 2.0- 4.0 sec  368 KBytes  1.51 Mbbits/sec
[ 3] 4.0- 6.0 sec  704 KBytes  2.88 Mbbits/sec
[ 3] 6.0- 8.0 sec  280 KBytes  1.15 Mbbits/sec
[ 3] 8.0-10.0 sec  208 KBytes  852 Kbits/sec
[ 3] 10.0-12.0 sec 344 KBytes  1.41 Mbbits/sec
[ 3] 12.0-14.0 sec 208 KBytes  852 Kbits/sec
[ 3] 14.0-16.0 sec 232 KBytes  950 Kbits/sec
[ 3] 16.0-18.0 sec 232 KBytes  950 Kbits/sec
[ 3] 18.0-20.0 sec 264 KBytes  1.08 Mbbits/sec
[ 3] 0.0-20.1 sec 3.00 MBytes  1.25 Mbbits/sec
```

→ Côté serveur:

```
#iperf -s -p 12000
```

```
-----
Server listening on TCP port 12000
TCP window size: 8.00 KByte (default)
-----
```

```
[852] local 10.1.1.1 port 12000 connected with 10.6.2.5 port 58316
[ ID] Interval Transfer Bandwidth
[852] 0.0-20.1 sec 3.00 MBytes 1.25 Mbbits/sec
```

[Haut de la page](#)

■ Tests UDP (-u), paramétrage de la bande passante (-b)
Voir la section "[Iperf](#)".

Les tests UDP avec l'argument -u vont donner de précieuses informations sur la gigue (jitter en anglais) ou les pertes de paquets. Si vous ne spécifiez pas l'argument -u, Iperf utilise TCP.

Pour garder une bonne qualité de lien, la perte de paquet ne devrait pas dépasser 1%. Un haut taux de perte de paquet générera un grand nombre de retransmissions de segments TCP ce qui affectera la bande passante.

La Gigue (jitter) est basiquement la variation de la latence et ne dépend pas de cette dernière. Il est tout à fait possible d'avoir des temps de réponse élevés et une gigue très basse. La valeur de la gigue est particulièrement importante pour des liens réseaux supportant la voix sur IP (VoIP, Voice over IP) parce qu'une gigue élevée peut interrompre un appel téléphonique.

L'argument -b permet d'allouer la bande passante désirée.

→ Côté client:

```
#iperf -c 10.1.1.1 -u -b 10m
```

```
-----
Client connecting to 10.1.1.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 108 KByte (default)
-----
```

```
[ 3] local 10.6.2.5 port 32781 connected with 10.1.1.1 port 5001
[ 3] 0.0-10.0 sec 11.8 MBytes 9.89 Mbits/sec
[ 3] Sent 8409 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec 11.8 MBytes 9.86 Mbits/sec 2.617 ms 9/ 8409 (0.11%)
```

→Côté serveur:

```
#iperf -s -u -i 1
```

```
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
-----
```

```
[904] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 32781
[ ID] Interval      Transfer      Bandwidth      Jitter      Lost/Total Datagrams
[904] 0.0- 1.0 sec  1.17 MBytes  9.84 Mbits/sec  1.830 ms    0/ 837 (0%)
[904] 1.0- 2.0 sec  1.18 MBytes  9.94 Mbits/sec  1.846 ms    5/ 850 (0.59%)
[904] 2.0- 3.0 sec  1.19 MBytes  9.98 Mbits/sec  1.802 ms    2/ 851 (0.24%)
[904] 3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec  1.830 ms    0/ 850 (0%)
[904] 4.0- 5.0 sec  1.19 MBytes  9.98 Mbits/sec  1.846 ms    1/ 850 (0.12%)
[904] 5.0- 6.0 sec  1.19 MBytes  10.0 Mbits/sec  1.806 ms    0/ 851 (0%)
[904] 6.0- 7.0 sec  1.06 MBytes  8.87 Mbits/sec  1.803 ms    1/ 755 (0.13%)
[904] 7.0- 8.0 sec  1.19 MBytes  10.0 Mbits/sec  1.831 ms    0/ 850 (0%)
[904] 8.0- 9.0 sec  1.19 MBytes  10.0 Mbits/sec  1.841 ms    0/ 850 (0%)
[904] 9.0-10.0 sec  1.19 MBytes  10.0 Mbits/sec  1.801 ms    0/ 851 (0%)
[904] 0.0-10.0 sec 11.8 MBytes  9.86 Mbits/sec  2.618 ms    9/ 8409 (0.11%)
```

[↩Haut de la page](#)

■Affichage de la taille de segment maximale (argument -m):

La taille de segment maximale (ou en anglais, Maximum Segment Size, MSS) est la plus grande quantité de données, en octets (bytes), qu'un ordinateur peut supporter en un unique et non-fragmenté segment TCP. Elle peut être calculée de la manière suivante:

MSS = MTU - en-têtes (headers) TCP & IP

Les en-têtes TCP & IP occupent 40 octets.

La MTU (Maximum Transmission Unit, unité de transmission maximale) est la plus grande quantité de données qui peut être transférée dans une trame.

Voici quelques tailles de MTU par défaut pour des topologies réseaux différentes:

Ethernet - 1500 octets: utilisé dans un réseau local (LAN).

PPPoE - 1492 octets: utilisé sur des liens ADSL.

Token Ring (16Mb/sec) - 17914 octets: vieille technologie créée par IBM.

Connexion téléphonique - 576 octets

Généralement, une MTU (et une MSS) élevée permet une plus grande bande passante.

→Côté client:

```
#iperf -c 10.1.1.1 -m
```

```
-----
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16.0 KByte (default)
-----
```

```
[ 3] local 10.6.2.5 port 41532 connected with 10.1.1.1 port 5001
[ 3] 0.0-10.2 sec 1.27 MBytes 1.04 Mbits/sec
[ 3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
```

Ici la MSS n'est pas égale à 1500 -40 mais à 1500 - 40 - 12(option Timestamps) = 1448

→Côté serveur:

```
#iperf -s
```

[Haut de la page](#)

■ Configuration de la taille de segment maximale (argument -M):

Utilisez l'argument -M pour changer la taille de segment maximale. (Maximale Segment Size, MSS)
 (Voir le test précédent pour plus d'explication sur la MSS)

```
#iperf -c 10.1.1.1 -M 1300 -m
```

WARNING: attempt to set TCP maximum segment size to 1300, but got 536

 Client connecting to 10.1.1.1, TCP port 5001
 TCP window size: 16.0 KByte (default)

```
[ 3] local 10.6.2.5 port 41533 connected with 10.1.1.1 port 5001
[ 3] 0.0-10.1 sec 4.29 MBytes 3.58 Mbits/sec
[ 3] MSS size 1288 bytes (MTU 1328 bytes, unknown interface)
```

→ Côté serveur:

```
#iperf -s
```

[Haut de la page](#)

■ Tests en parallèle (argument -P):

Utilisez l'argument -P pour lancer des tests en parallèle.

→ Côté client:

```
#iperf -c 10.1.1.1 -P 2
```

 Client connecting to 10.1.1.1, TCP port 5001
 TCP window size: 16.0 KByte (default)

```
[ 3] local 10.6.2.5 port 41534 connected with 10.1.1.1 port 5001
[ 4] local 10.6.2.5 port 41535 connected with 10.1.1.1 port 5001
[ 4] 0.0-10.1 sec 1.35 MBytes 1.12 Mbits/sec
[ 3] 0.0-10.1 sec 1.35 MBytes 1.12 Mbits/sec
[SUM] 0.0-10.1 sec 2.70 MBytes 2.24 Mbits/sec
```

→ Côté serveur:

```
#iperf -s
```

[Haut de la page](#)

■ Aide Iperf:

```
#iperf -h
```

Usage: iperf [-s|-c host] [options]
 iperf [-h|--help] [-v|--version]

Client/Server:

```
-f --format      [kmKM] format to report: Kbits, Mbits, KBytes, MBytes
                  format de rapport: Kbits, Mbits, KBytes, MBytes
-i --interval    #          seconds between periodic bandwidth reports
                  secondes entre les rapports périodiques de bande passante
-l --len         #[KM]     length of buffer to read or write (default 8 KB)
                  longueur du tampon pour lire ou écrire (défaut 8 Kb)
-m --print_mss  #          print TCP maximum segment size (MTU - TCP/IP header)
```

-p --port # affiche la taille de segment TCP maximale (MTU - en-têtes TCP/IP)
 server port to listen on/connect to
 port du serveur
 -u --udp # use UDP rather than TCP
 utilisation d'UDP plutôt que TCP
 -w --window # [KM] TCP window size (socket buffer size)
 taille de la fenêtre TCP (taille du tampon de la socket)
 -B --bind "host" # bind to "host", an interface or multicast address
 attachement à l'"host" (hôte), une interface ou adresse multicast
 -C --compatibility # for use with older versions does not sent extra msgs
 pour l'utilisation avec de vieilles version
 -M --mss # set TCP maximum segment size (MTU - 40 bytes)
 configure le taille de segment TCP maximale (MTU - 40 bytes)
 -N --nodelay # set TCP no delay, disabling Nagle's Algorithm
 configure le paramètre "TCP no delay", désactivation de l'algorithme de Nagle.
 Set the domain to IPv6
 -V --IPv6Version # Configure le domaine en IPv6

Server specific:

-s --server # run in server mode
 lancement en mode serveur
 -U --single_udp # run in single threaded UDP mode
 -
 -D --daemon # run the server as a daemon
 lancement du serveur en démon m

Client specific:

-b --bandwidth # [KM] for UDP, bandwidth to send at in bits/sec (default 1 Mbit/sec, implies -u)
 pour UDP, bande passante à envoyer en bits/sec (défaut 1 Mbit/sec, implique -u)
 -c --client # "host" run in client mode, connecting to "host"
 lancement en mode client, connexion à l'"host" (hôte).
 -d --dualtest # Do a bidirectional test simultaneously
 Fait un test bidirectionnel simultané.
 -n --num # [KM] number of bytes to transmit (instead of -t)
 nombre de bytes à transmettre (au lieu de -t)
 -r --tradeoff # Do a bidirectional test individually
 Fait un test bidirectionnel individuellement
 -t --time # time in seconds to transmit for (default 10 secs)
 temps en seconde pour transmettre (défaut 10 sec)
 -F --fileinput # "name" input the data to be transmitted from a file
 -
 -I --stdin # input the data to be transmitted from stdin
 -
 -L --listenport # port to receive bidirectional tests back on
 port pour recevoir des tests bidirectionnels en retour.
 -P --parallel # number of parallel client threads to run
 nombre de tests client en parallèle à lancer
 -T --ttl # time-to-live, for multicast (default 1)
 time-to-live (temps de vie), pour multicast (défaut 1)

Miscellaneous:

-h --help # print this message and quit
 affiche ce message et quitte
 -v --version # print version information and quit
 affiche l'information de version et quitte

[Haut de la page](#)

JPERF

[Jperf](#) est une interface graphique pour Iperf écrit en Java.

■1. Installation:

[Download](#) Jperf.

→Linux

Décompressez le fichier téléchargé :

```
#tar -xvf jperf2.0.0.zip
```

Lancez Jperf.

```
#cd jperf2.0.0  
#./jperf.sh
```

Si vous avez le message suivant, ceci signifie que vous avez besoin d'installer Iperf avec la commande: "apt-get install iperf"

*Iperf is probably not in your Path!
Please download it here '<http://dast.nlanr.net/Projects/Iperf/>'
and put the executable in your PATH environment variable.*



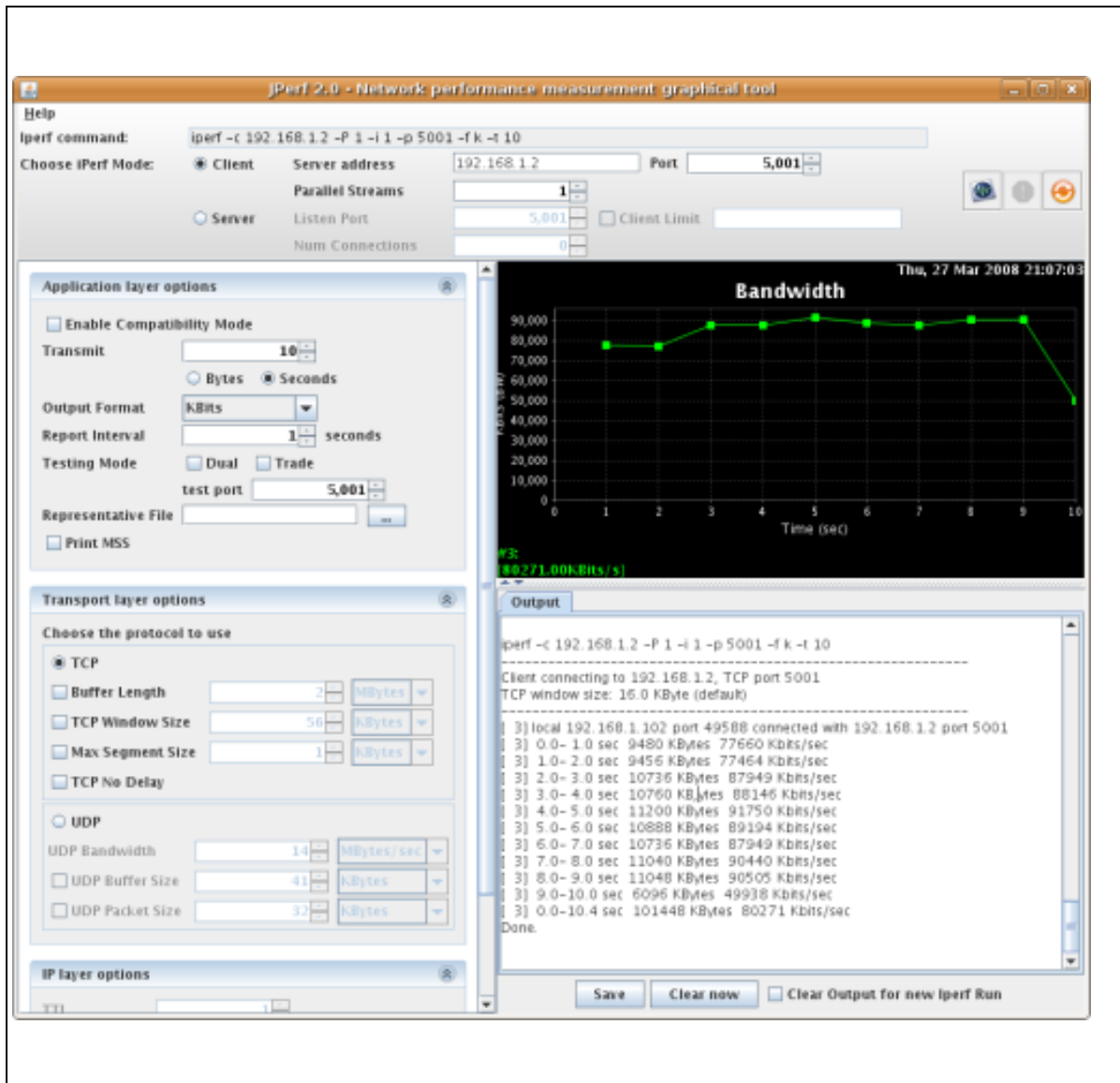
→Microsoft Windows

Décompressez le fichier téléchargé avec votre programme favori.
Accédez au dossier décompressé appelé par défaut "jperf2.0.0" et double-cliquez sur "iperf.bat".
Notez que l'utilitaire Iperf est déjà présent dans le dossier /bin.

2. Exemples:

→ Paramètres par défaut, mesure de la bande passante:
Voir la section "[Iperf](#)" pour plus de détails.

- Client Linux:



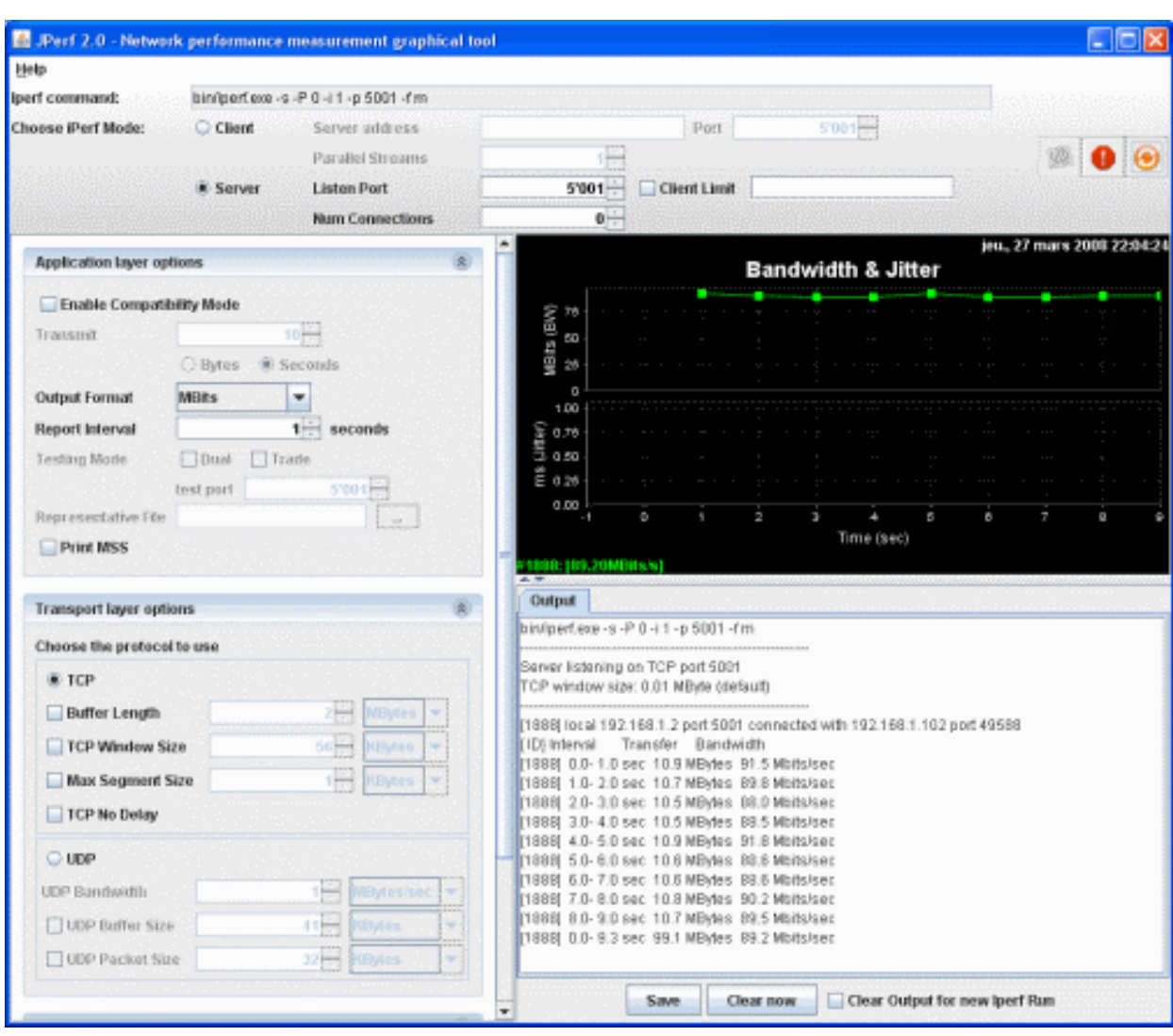
The screenshot shows the jPerf 2.0 graphical tool interface. The main window is titled "jPerf 2.0 - Network performance measurement graphical tool". The interface is divided into several sections:

- Help:** Displays the iperf command: `iperf -c 192.168.1.2 -P 1 -i 1 -p 5001 -f k -t 10`.
- Choose iPerf Mode:** Radio buttons for Client (selected) and Server. Fields for Server address (192.168.1.2), Port (5,001), Parallel Streams (1), Listen Port (5,001), and Num Connections (0).
- Application layer options:**
 - Enable Compatibility Mode:
 - Transmit: 10 (seconds)
 - Output Format: KBits
 - Report Interval: 1 seconds
 - Testing Mode: Dual, Trade
 - test port: 5,001
 - Representative File:
 - Print MSS:
- Transport layer options:**
 - Choose the protocol to use:
 - TCP (selected):
 - Buffer Length: 2 (MBytes)
 - TCP Window Size: 56 (KBytes)
 - Max Segment Size: 1 (KBytes)
 - TCP No Delay:
 - UDP:
 - UDP Bandwidth: 14 (MBytes/sec)
 - UDP Buffer Size: 41 (KBytes)
 - UDP Packet Size: 32 (KBytes)
- IP layer options:**
- Bandwidth Graph:** A line graph titled "Bandwidth" showing Kbits/s over Time (sec). The y-axis ranges from 0 to 90,000. The x-axis ranges from 0 to 10. The graph shows a steady increase in bandwidth from approximately 77,660 Kbits/sec at 0-1 seconds to a peak of 80,271 Kbits/sec at 0-10.4 seconds, followed by a sharp decline.
- Output:** A text area showing the iperf command and the results of the test:

```
iperf -c 192.168.1.2 -P 1 -i 1 -p 5001 -f k -t 10
Client connecting to 192.168.1.2, TCP port 5001
TCP window size: 16.0 KByte (default)

[ 3] local 192.168.1.102 port 49588 connected with 192.168.1.2 port 5001
[ 3] 0.0- 1.0 sec  9480 KBytes  77660 Kbits/sec
[ 3] 1.0- 2.0 sec  9456 KBytes  77464 Kbits/sec
[ 3] 2.0- 3.0 sec 10736 KBytes  87949 Kbits/sec
[ 3] 3.0- 4.0 sec 10760 KBytes  88146 Kbits/sec
[ 3] 4.0- 5.0 sec 11200 KBytes  91750 Kbits/sec
[ 3] 5.0- 6.0 sec 10888 KBytes  89194 Kbits/sec
[ 3] 6.0- 7.0 sec 10736 KBytes  87949 Kbits/sec
[ 3] 7.0- 8.0 sec 11040 KBytes  90440 Kbits/sec
[ 3] 8.0- 9.0 sec 11048 KBytes  90505 Kbits/sec
[ 3] 9.0-10.0 sec  6096 KBytes  49938 Kbits/sec
[ 3] 0.0-10.4 sec 101448 KBytes  80271 Kbits/sec
Done.
```

- Serveur Windows:



Application layer options

- Enable Compatibility Mode
- Transmit: 10
- Output Format: MBits
- Report Interval: 1 seconds
- Testing Mode: Dual Trade
- test port: 5001
- Print MSS

Transport layer options

Choose the protocol to use

- TCP
 - Buffer Length: 2 MBytes
 - TCP Window Size: 50 KBytes
 - Max Segment Size: 1 KBytes
 - TCP No Delay
- UDP
 - UDP Bandwidth: 1 MBytes/sec
 - UDP Buffer Size: 41 KBytes
 - UDP Packet Size: 32 KBytes

Bandwidth & Jitter

Time (sec): -1 to 9

MBits (BW): 0 to 75

ms (Jitter): 0.00 to 1.00

Output

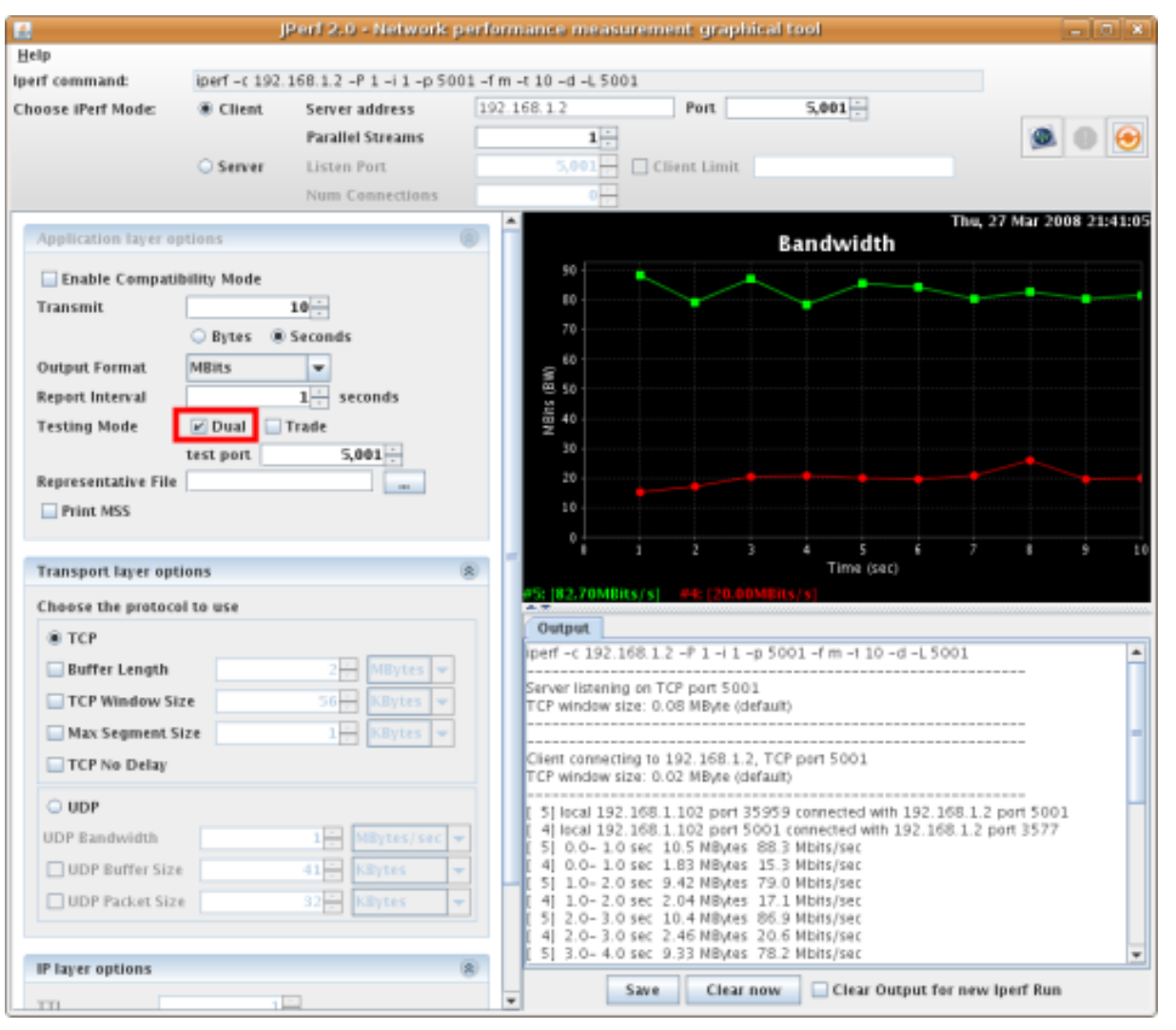
```
bin/perf.exe -s -P 0 -i 1 -p 5001 -f m
Server listening on TCP port 5001
TCP window size: 0.01 MByte (default)

[1888] local 192.168.1.2 port 5001 connected with 192.168.1.102 port 49588
[ID] interval Transfer Bandwidth
[1888] 0.0-1.0 sec 10.9 MBytes 91.5 Mbits/sec
[1888] 1.0-2.0 sec 10.7 MBytes 89.8 Mbits/sec
[1888] 2.0-3.0 sec 10.5 MBytes 88.0 Mbits/sec
[1888] 3.0-4.0 sec 10.5 MBytes 88.5 Mbits/sec
[1888] 4.0-5.0 sec 10.9 MBytes 91.8 Mbits/sec
[1888] 5.0-6.0 sec 10.8 MBytes 89.8 Mbits/sec
[1888] 6.0-7.0 sec 10.6 MBytes 88.8 Mbits/sec
[1888] 7.0-8.0 sec 10.8 MBytes 90.2 Mbits/sec
[1888] 8.0-9.0 sec 10.7 MBytes 89.5 Mbits/sec
[1888] 0.0-9.3 sec 99.1 MBytes 89.2 Mbits/sec
```

[Haut de la page](#) [Jperf](#)

→ Mesure de la bande passante bidirectionnelle simultanée:
 Voir la section "[Iperf](#)" pour plus de détails.

- Client Linux:



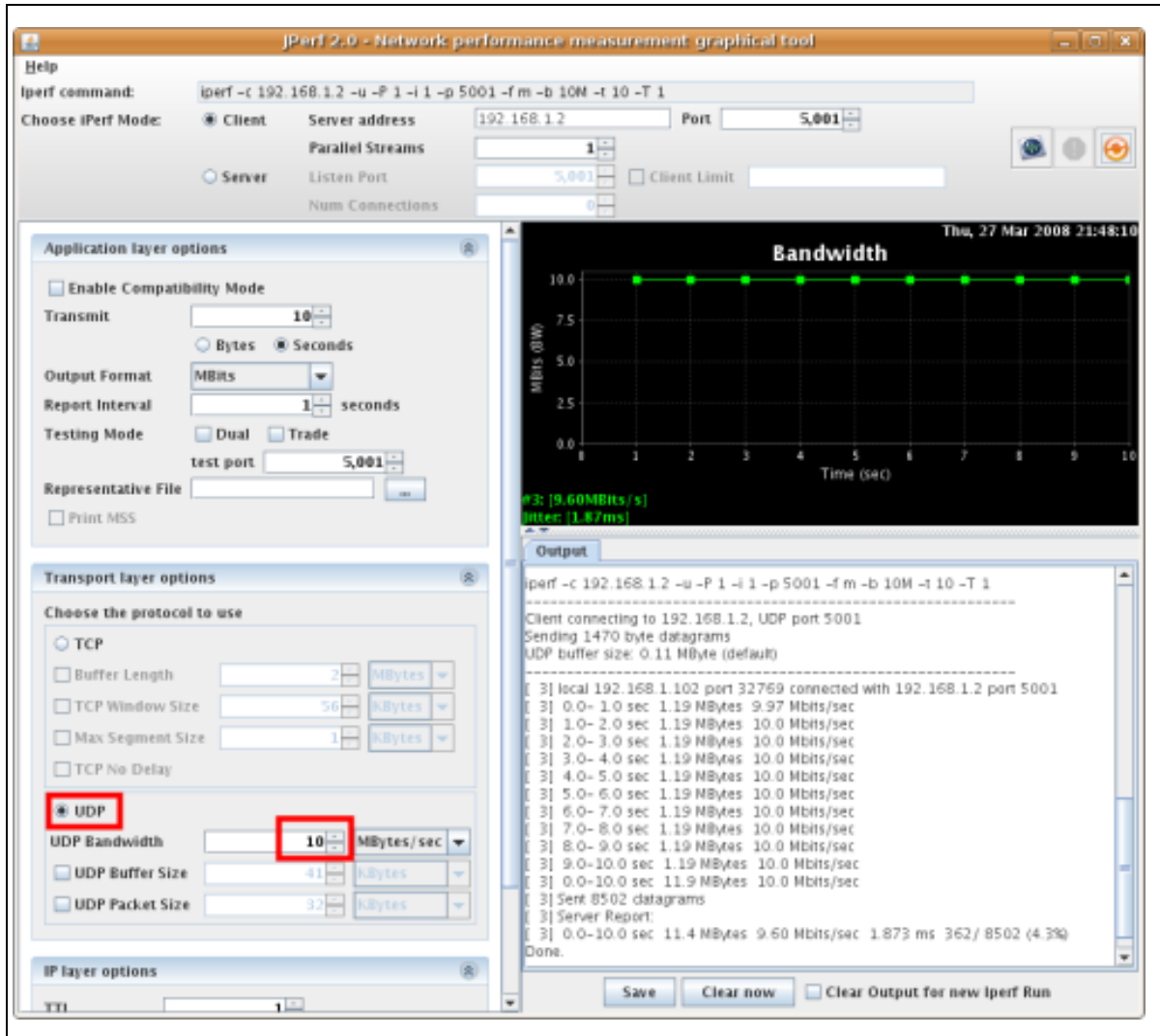
The screenshot shows the JPerf 2.0 graphical tool interface. The main window title is "JPerf 2.0 - Network performance measurement graphical tool". The interface is divided into several sections:

- Help:** Iperf command: `iperf -c 192.168.1.2 -P 1 -i 1 -p 5001 -f m -t 10 -d -L 5001`
- Choose IPerf Mode:** Client (selected), Server address: 192.168.1.2, Port: 5,001. Other options include Parallel Streams, Listen Port, Num Connections, and Client Limit.
- Application layer options:**
 - Enable Compatibility Mode:
 - Transmit: 10 (Units: Bytes/Seconds)
 - Output Format: Mbits
 - Report Interval: 1 seconds
 - Testing Mode: Dual, Trade
 - test port: 5,001
 - Representative File:
 - Print MSS:
- Transport layer options:**
 - Choose the protocol to use: TCP (selected), UDP
 - TCP options: Buffer Length (2 Mbytes), TCP Window Size (56 Kbytes), Max Segment Size (1 Kbytes), TCP No Delay
 - UDP options: UDP Bandwidth (1 Mbytes/sec), UDP Buffer Size (41 Kbytes), UDP Packet Size (32 Kbytes)
- IP layer options:** TTL:
- Bandwidth Graph:** A line graph titled "Bandwidth" showing MBits (BW) on the Y-axis (0 to 90) and Time (sec) on the X-axis (0 to 10). The graph shows two data series: a green line for the client (labeled "C") and a red line for the server (labeled "S"). The client's bandwidth fluctuates between approximately 75 and 90 MBits/sec, while the server's bandwidth fluctuates between approximately 15 and 25 MBits/sec. The current status is displayed as "C: [82.70MBits/s] S: [20.60MBits/s]".
- Output:** A text area showing the command and the results of the test:


```
iperf -c 192.168.1.2 -P 1 -i 1 -p 5001 -f m -t 10 -d -L 5001
Server listening on TCP port 5001
TCP window size: 0.08 MByte (default)
-----
Client connecting to 192.168.1.2, TCP port 5001
TCP window size: 0.02 MByte (default)
-----
[ 5] local 192.168.1.102 port 35959 connected with 192.168.1.2 port 5001
[ 4] local 192.168.1.102 port 5001 connected with 192.168.1.2 port 3577
[ 4] 0.0- 1.0 sec 10.5 Mbytes 88.3 Mbits/sec
[ 4] 1.0- 2.0 sec 9.42 Mbytes 79.0 Mbits/sec
[ 4] 2.0- 3.0 sec 10.4 Mbytes 86.9 Mbits/sec
[ 4] 3.0- 4.0 sec 9.33 Mbytes 78.2 Mbits/sec
```


- ➔ Mesure de la gigue:
Voir la section "[Iperf](#)" pour plus de détails.

- Client Linux:



The screenshot shows the JPerf 2.0 graphical tool interface. The main window title is "JPerf 2.0 - Network performance measurement graphical tool".

Configuration:

- iperf command:** `iperf -c 192.168.1.2 -u -P 1 -i 1 -p 5001 -f m -b 10M -t 10 -T 1`
- Choose IPerf Mode:** Client (selected), Server
- Server address:** 192.168.1.2
- Port:** 5,001
- Parallel Streams:** 1
- Listen Port:** 5,001
- Client Limit:** (empty)
- Num Connections:** 0

Application layer options:

- Enable Compatibility Mode
- Transmit:** 10
- Bytes Seconds
- Output Format:** Mbits
- Report Interval:** 1 seconds
- Testing Mode:** Dual Trade
- test port:** 5,001
- Representative File:** (empty)
- Print MSS

Transport layer options:

- Choose the protocol to use:**
 - TCP
 - Buffer Length: 2 MBytes
 - TCP Window Size: 56 KBytes
 - Max Segment Size: 1 KBytes
 - TCP No Delay
 - UDP**
 - UDP Bandwidth:** 10 Mbytes/sec
 - UDP Buffer Size: 41 KBytes
 - UDP Packet Size: 32 KBytes

IP layer options:

- YTI:** 1

Bandwidth Graph:

Thu, 27 Mar 2008 21:48:10

Mbps (lW)

Time (sec)

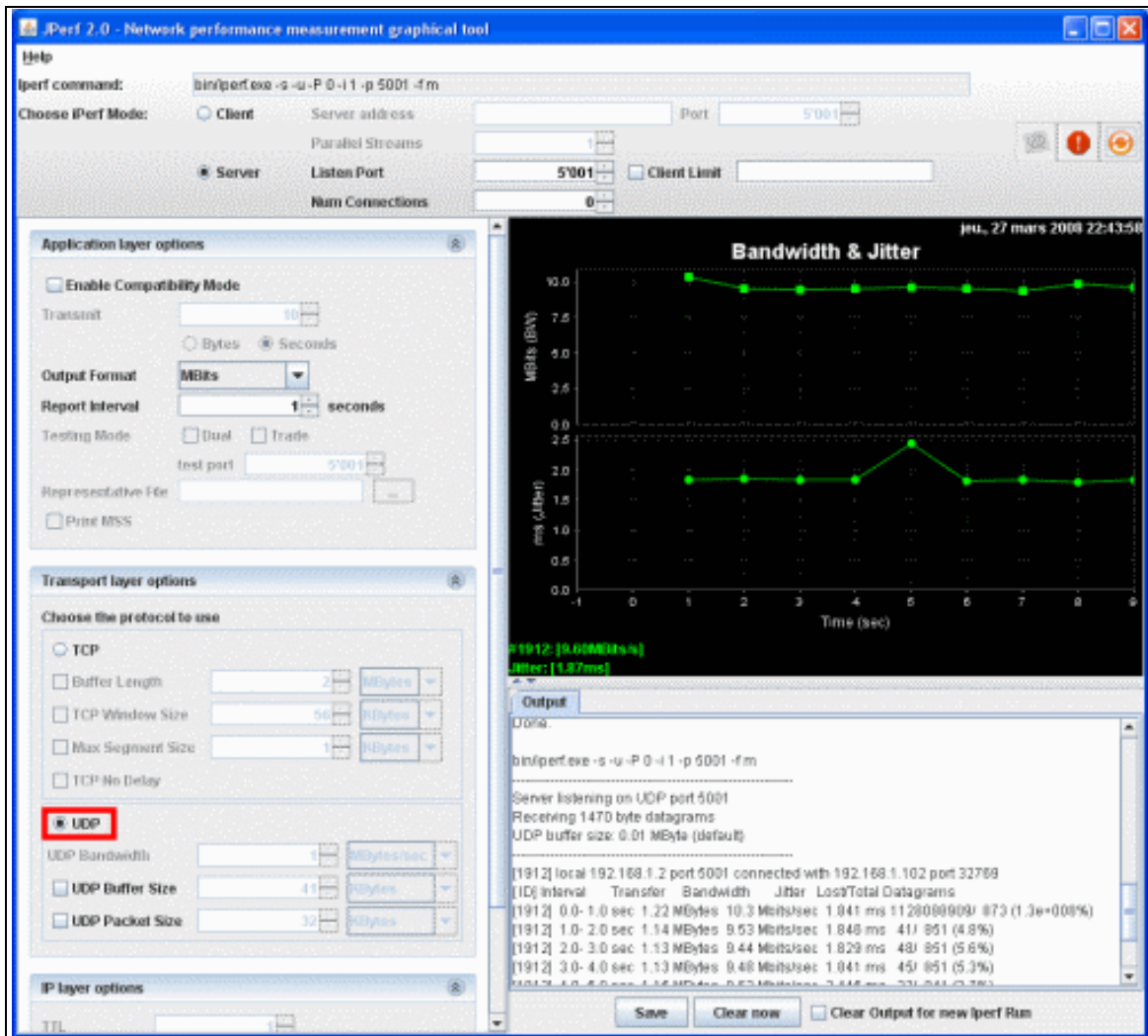
#3: [9.60Mbits/s]
#iter: [1.87ms]

Output:

```
iperf -c 192.168.1.2 -u -P 1 -i 1 -p 5001 -f m -b 10M -t 10 -T 1
-----
Client connecting to 192.168.1.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 0.11 MByte (default)
-----
[ 3] local 192.168.1.102 port 32769 connected with 192.168.1.2 port 5001
[ 3] 0.0- 1.0 sec 1.19 MBytes 9.97 Mbits/sec
[ 3] 1.0- 2.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 2.0- 3.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 3.0- 4.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 4.0- 5.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 5.0- 6.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 6.0- 7.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 7.0- 8.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 8.0- 9.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 9.0-10.0 sec 1.19 MBytes 10.0 Mbits/sec
[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec
[ 3] Sent 8502 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec 11.4 MBytes 9.60 Mbits/sec 1.873 ms 362/ 8502 (4.3%)
Done.
```

Buttons: Save, Clear now, Clear Output for new Iperf Run

- Serveur Windows:



Bandwidth & Jitter

Mbps (BW)

ms (Jitter)

Time (sec)

Output

```
bin/iperf.exe -s -u -P 0 -i 1 -p 5001 -f m
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 0.01 MByte (default)

[1912] local 192.168.1.2 port 5001 connected with 192.168.1.102 port 32769
[0] interval Transfer Bandwidth Jitter Loss/Total Datagrams
[1912] 0.0 - 1.0 sec 1.22 MBytes 10.3 Mbits/sec 1.841 ms 1120000009/ 873 (1.3e+00%)
[1912] 1.0 - 2.0 sec 1.14 MBytes 9.53 Mbits/sec 1.848 ms 41/ 851 (4.8%)
[1912] 2.0 - 3.0 sec 1.13 MBytes 9.44 Mbits/sec 1.829 ms 48/ 851 (5.6%)
[1912] 3.0 - 4.0 sec 1.13 MBytes 9.48 Mbits/sec 1.841 ms 45/ 851 (5.3%)
[1912] 4.0 - 5.0 sec 1.16 MBytes 9.77 Mbits/sec 1.845 ms 37/ 841 (4.3%)
```