# ip-route

## Gestion des tables de routage

ip route est utilisé pour manipules les entrées dans les tables de routage du kernel. Les types de routes sont :

**unicast** L'entrée décrit des véritables chemins vers les destinations couvertes par le préfixe de route.

**unreachable** Ces destinations sont inatteignables. Les paquets sont supprimés silencieusement.

**prohibit** Ces destinations sont inatteignables. les paquets sont supprimés silencieusement et un message ICMP est généré.

**local** Les destinations sont assignées à cet hôte. Les paquets sont délivrés localement.

**broadcast** Les destinations sont des adresses de broadcast. Les paquets sont envoyés sur des liens broadcast.

**throw** Une route de contrôle spéciale utilisé avec les règles de stratégie. Si une telle route est séléctionée, rechercher dans cette table se termine en prétendant qu'aucune route n'a été trouvée et un ICMP unreachable est émis.

**anycast** non implémenté

**multicast** Un type spécial utilisé pour le routage multicast. N'est pas présent dans les tables de routage normales.

Linux 2.x peut grouper les routes dans plusieurs tables de routage identifiées par un numéro dans la plage 1 à 2^31 ou par un nom depuis le fichier **/etc/iproute2/rt_tables**. Par défaut, toutes les routes sont insérées dans la table **main** (254) et le kernel n'utilise que cette table. Les valeurs 0, 253, 254 et 255 sont réservés.

Actuellement, une autre table existe toujours, qui est invisible mais importante, la table **local** (255). Cette table consiste de routes pour les adresses locales et de broadcast. Le kernel maintient cette table automatiquement et l'administrateur n'a généralement pas besoin de la modifier.

Les tables de routages multiples entrent en jeu quand des stratégies de routage sont utilisées.

**add** Ajouter une nouvelle route

**change** Changer une route

**replace** Changer ou ajouter une nouvelle route

**to TYPE PREFIX** Préfix de destination de la route. Si type est omis, assume unicast. PREFIX est une adresse IP ou IPv6 suivi optionnellement par un masque. les PREFIX spécial **default** est équivalent à **0/0** ou **: :/0**

**tos TOS**

**dsfield TOS** La clé Type Of Service. Cette clé n'a pas de masque associé et le match le plus long est interprété comme : 1, compare le TOS de la route et du paquet. S'il ne sont pas égal, le paquet peut matcher une route avec un TOS 0. TOS est soit un nombre hexa 8-bits, soit un identifiant de /etc/iproute2/rt_dsfield.

**metric NUMBER**

**preference NUMBER** La valeur de préférence de la route. NUMBER est un nombre 32-bits arbitraire.

**table TABLEID** La table où ajouter cette route. TABLEID peut être un nombre ou une chaîne du fichier /etc/iproute2/rt_tables. Si omis, assume la table main.

**dev NAME** Nom du périphérique de sortie.

**via ADDRESS** L'adresse du prochain saut.

**src ADDRESS** L'adresse source à préférer en envoyant au destinations couvertes par le préfixe de route.

**realm REALMID** Le domaine auquel la route est assignée. Peut être un nombre ou une chaîne depuis le fichier /etc/iproute2/rt_realms

**mtu MTU**

**mtu lock MTU** Le MTU associé avec le chemin de la destination. Si lock n'est pas utilisé, le MTU peut être mis à jours par le kernel via le Path MTU Discovery.

**window NUMBER** Fenêtre TCP maximum à annoncer à ces destinations, mesurée en octets. Il limite les salves de données maximales que les paires TCP sont autorisés à nous envoyer.

**rtt TIME** Le RTT (Round Trip Time) initial estimé. Si aucun suffixe n'est spécifié les unités sont des valeurs brutes passées directement au code de routage pour maintenir la compatibilité avec les versions précédentes. Sinon un suffixe de s, sec, ou secs est utilisé pour spécifier les secondes et ms, msec ou msecs pour spécifier les millisecondes.

**rttvar TIME** Variante RTT initiale estimée. Les valeur sont spécifiée comme avec rtt.

**rto_min TIME** TimeOut de retransmission TCP minimum à utiliser en communiquant avec cette destination. Les valeurs sont spécifiées comme pour rtt.

**ssthresh NUMBER** Une estimation pour le seuil initial de début lent.

**cwnd NUMBER** le clamp pour la fenêtre de congestion. Il est ignoré si le flag lock n'est pas utilisé.

**initcwnd NUMBER** La taille de fenêtre initialement reçue pour les connexions à cette destination. La taille de fenêtre actuelle est cette valeur multipliée par le MSS de la connexion. La valeur par défaut est 0, signifiant l'utilisation d'une valeur Slow Start

**quickack BOOL** Active/désactive ack rapide pour les connexions à cette destination.

**advmss NUMBER** le MSS (Maximal Segment Size) à annoncer aux destinations en établissant des connexions TCP. Si non spécifié, Linux utilise une valeur par défaut calculée depuis le premier MTU du prochain périphérique.

**reordering NUMBER** Ré-ordonnancement maximum dans le chemin vers cette destination. Si non donné, Linux utilise la valeur sélectionnée avec sysctl (net/ipv4/tcp_reordering)

**nexthop NEXTHOP** Le prochain saut d'une route multi-path. NEXTHOP est une valeur complexe avec sa propre syntaxe similaire à la liste des argument top-level :

**via ADDRESS** Router suivant

**dev NAME** Périphérique de sortie

**weight NUMBER** Poid pour cet élément d'une route multi-path reflétant sa bande passante relative ou qualité.

**scope SCOPE_VAL** Le scope des destinations couvertes par le préfixe de route. SCOPE_VAL peut être un nombre ou une chaîne du fichier /etc/iproute2/rt_scopes. Si omis, ip assume le scope global pour toutes les routes unicast avec gateway, le scope link pour toutes les routes unicast et broadcast directes, et le scope host pour les routes locales.

**protocol RTPROTO** L'identifiant de protocole de routage de cette route. RTPROTO peut être un nombre ou une chaîne depuis le fichier /etc/iproute2/rt_protos. De nombreuses valeurs de protocole one une interprétation fixée :

**redirect** La route a été installée due à une redirection ICMP

**kernel** La route a été installée par le kernel durant l'autoconfiguration

**boot** La route a été installée durant la séquence de boot.

**static** La route a été installée par l'administrateur

**ra** La route a été installée par Router Discovery Protocol.

**onlink** Prétend que le prochain saut est directement attaché à ce lien, même s'il ne matche pas de préfixe d'interface.

**delete** Supprime une route. A les même arguments que ip route add, mais leur sémantique sont un peu différents. Les valeur de clé (to, tos, preference et table) sélectionnent la route à supprimer. Si des attributs optionnels sont présents, ip vérifie qu'ils coïncident avec les attributs de la route à supprimer. Si aucune route n'est trouvée, ip route del échoue.

**show** Liste les routes. La commande affiche le contenu des tables de routage ou des routes sélectionnées par les critères.

**to SELECTOR** Sélectionne seulement les routes depuis la plage de destinations données. SELECTOR consiste d'un modifier optionnel (root, match ou exact) et d'un préfixe. root PREFIX sélectionne les routes avec les préfixes pas plus cours que PREFIX. ex : root 0/0 sélectionne toute la table de routage. match PREFIG sélectionne les routes avec les préfixe pas plus long que PREFIX. ex : match 10.0/16 sélectionne 10.0/16, 10/8 et 0/0, mais pas 10.1/16 et 10.0.0/24. exact PREFIX sélectionne les routes avec le préfixe exact. Non spécifié, assume root 0/0.

**tos TOS**

**dsfield TOS** Selectionne seulement les routes avec le TOS donné

**table TABLEID** Affiche les routes depuis cette table. défaut : table main. TABLEID peut être soit l'ID soit le nom de la table, ou une des valeurs spéciales :

**all** Liste toutes les tables

**cache** dump le cache de routage

**cloned**

**cached** Liste les routes clonnées, par ex. les routes qui ont été forkée dynamiquement depuis d'autres routes parce que certains attributs de route ont été mis à jours (ex : MTU). Actuellement équivalent à table cache.

**from SELECTOR** Même syntaxe que to, mais lie la plage d'adresse source au lieu de destination. ne fonctionne qu'avec les routes clonées.

**protocol RTPROTO** Liste seulement les routes de ce protocole

**scope SCOPE_VAL** Liste seulement les routes dans ce scope

**type TYPE** Liste seulement les routes de ce type

**dev NAME** Liste seulement les routes allant via ce périphérique

**via PREFIX** Liste seulement les routes allant via les routeurs de prochains saut sélectionnés par PREFIX

**src PREFIX** Liste seulement les routes avec les adresses sources préférées sélectionnées par PREFIX

**realm REALMID**

**realms FROMREALM/TOREALM** Liste seulement les routes avec ces domaines

**flush** Vide les routes sélectionnée par certains critères. Les arguments ont la même syntaxe et sémantique de ip route show, mais les tables de routage ne sont pas listées, mais purgées. La seule différence est l'action par défaut : show dump toute la table de routage main, mais flush affiche l'aide. Avec l'option -statistics, la commande devient verbeuse. Si donné 2 fois, dumps également toutes les routes supprimées.

**get** Récupère une simple route et affiche sont contenu exactement comme le kernel la vois.

**to ADDRESS** Adresse de destination

**from ADDRESS** Adresse source

**tos TOS**

**dsfield TOS** Le type de service

**iif NAME** Le périphérique depuis lequel ce paquet est prévu d'arriver

**oif NAME** Force le périphérique de sortie sur lequel ce paquet sera routé.

**connected** Si aucune adresse source (option from) n'est donnée, recherche la route avec le jeu source à l'adresse préférée reçue depuis la première recherche.

Noter que cette opération n'est pas équivalente à ip route show. show affiche les routes de sortie. get les résous et créé de nouveaux clones is nécessaire. Essentiellement, get est équivalent à envoyer un paquet sur ce chemin. Si iif n'est pas donné, le kernel créé une route pour sortir les paquets via de destination demandée. C'est équivalent à pinger la destination avec un ip route ls cache, cependant, aucun paquet n'est actuellement envoyé. Avec iif, le kernel prétent qu'un paquet est arrivé depuis cette interface et recherche un chemin pour forwarder le paquet.

**save** Sauvegarde les information de la table de routage. Fonctionne comme show, excepté que la sortie est une donnée utilisable par restore.

**restore** Restaure les informations de table de routage

# Exemples

Affiche toutes les entrées de route dans le kernel :
**ip ro**
Ajoute une route par défaut via 192.168.1.1 qui peut être atteins via eth0 :
**ip route add default via 192.168.1.1 dev eth0**