Université de Montréal

Une architecture parallèle distribuée et tolérante aux pannes pour le protocole interdomaine BGP au cœur de l'Internet

Par

Wissam Hamzeh

Département d'informatique et
de recherche opérationnelle

Faculté des arts et sciences

Thèse présentée à la Faculté des arts et sciences
en vue de l'obtention du grade de
PhD sciences (PhD. Sc.)
en informatique

Décembre, 2010

Université de Montréal
Faculté des arts et sciences

Cette thèse intitulée :

Une architecture parallèle distribuée et tolérante aux pannes pour le protocole interdomaine BGP au cœur de l'Internet

Présentée par :
Wissam Hamzeh

a été évaluée par un jury composé des personnes suivantes :

ElMostapha Aboulhamid, président-rapporteur
Abdelhakim Hafid, directeur de recherche
Samuel Pierre, membre du jury
Ali Miri, examinateur externe
Denis Cousineau, Représentant du doyen de la FES

Résumé

L'augmentation du nombre d'usagers de l'Internet a entraîné une croissance exponentielle dans les tables de routage. Cette taille prévoit l'atteinte d'un million de préfixes dans les prochaines années. De même, les routeurs au cœur de l'Internet peuvent facilement atteindre plusieurs centaines de connexions BGP simultanées avec des routeurs voisins. Dans une architecture classique des routeurs, le protocole BGP s'exécute comme une entité unique au sein du routeur. Cette architecture comporte deux inconvénients majeurs : l'extensibilité (scalabilité) et la fiabilité. D'un côté, la scalabilité de BGP est mesurable en termes de nombre de connexions et aussi par la taille maximale de la table de routage que l'interface de contrôle puisse supporter. De l'autre côté, la fiabilité est un sujet critique dans les routeurs au cœur de l'Internet. Si l'instance BGP s'arrête, toutes les connexions seront perdues et le nouvel état de la table de routage sera propagé tout au long de l'Internet dans un délai de convergence non trivial. Malgré la haute fiabilité des routeurs au cœur de l'Internet, leur résilience aux pannes est augmentée considérablement et celle-ci est implantée dans la majorité des cas via une redondance passive qui peut limiter la scalabilité du routeur. Dans cette thèse, on traite les deux inconvénients en proposant une nouvelle approche distribuée de BGP pour augmenter sa scalabilité ainsi que sa fiabilité sans changer la sémantique du protocole. L'architecture distribuée de BGP proposée dans la première contribution est faite pour satisfaire les deux contraintes : scalabilité et fiabilité. Ceci est accompli en exploitant adéquatement le parallélisme et la distribution des modules de BGP sur plusieurs cartes de contrôle. Dans cette contribution, les fonctionnalités de BGP sont divisées selon le paradigme « maître-esclave » et le RIB (Routing Information Base) est dupliqué sur plusieurs cartes de contrôle. Dans la deuxième contribution, on traite la tolérance aux pannes dans l'architecture élaborée dans la première contribution en proposant un mécanisme qui augmente la fiabilité. De plus, nous prouvons analytiquement dans cette contribution qu'en adoptant une telle architecture distribuée, la disponibilité de BGP sera augmentée considérablement versus une architecture monolithique. Dans la troisième contribution, on propose une méthode de partitionnement de la table de routage que nous avons appelé DRTP pour diviser la table de BGP sur plusieurs cartes de contrôle. Cette contribution vise à augmenter la scalabilité de la table de routage et la parallélisation

de l'algorithme de recherche (Best Match Prefix) en partitionnant la table de routage sur plusieurs nœuds physiquement distribués.

Mots clefs : Routeurs, BGP, Systèmes distribués, Parallélisme, Fiabilité, Tolérance aux pannes.

Abstract

The increasing number of end users has led to an exponential growth in the Internet routing table. The routing table is expected to reach a size of one million prefixes within the coming few years. Besides, current core routers may easily attain hundreds of connected BGP peers simultaneously. In classical monolithic architecture, the BGP protocol runs as a single entity inside the router. This architecture suffers from two drawbacks: scalability and reliability. BGP scalability can be measured in terms of the number of connected peers that can be handled and the size of the routing table. On the other hand, the reliability is a critical issue in core routers. If the BGP instance inside the router fails, all peers' connections will shutdown and the new reachability state will be propagated across the Internet in a non trivial convergence delay. Although, in current core routers, the resiliency is increased considerably, it's mainly implemented via a primary-backup redundancy scheme which limits the BGP scalability. In this thesis we address the two mentioned BGP drawbacks by proposing a novel distributed approach to increase both scalability and reliability of BGP without changing the semantic of the protocol. The BGP distributed architecture in the first paper is built to satisfy both requirements: scalability and reliability by adequately exploiting parallelism and module separation. In our model, BGP functionalities are split in a master-slave manner and the RIB (Routing Information Base) is replicated to multiple controller cards, to form a cluster of parallel computing entities. In the second paper, we address the fault tolerance of BGP within the distributed architecture presented in the first paper. We prove analytically that, by adopting the distributed architecture of BGP the availability of BGP will be increased considerably versus a monolithic architecture. In the third paper we propose a distributed parallel scheme called DRTP to partition the BGP routing table on multiple controller cards. DRTP aims at increasing the BGP scalability and the parallelization of the Best Match Prefix algorithm.

Keywords : Routers, BGP, Distributed systems, Parallelism, Reliability, Fault-Tolerance.

Table des matières

Liste des figures

Liste des tableaux

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé pour l'accomplissement de mon doctorat. Précisément, mon directeur de recherche Abdelhakim Hafid le chef du laboratoire de Réseaux et Télécommunications pour son soutien, sa persévérance et son encouragement pour réaliser cette thèse.

Tout mon amour pour mon épouse Ola et mes trois merveilleux garçons : Hussein, Hady et Chady qui sont ma source d'inspiration et le moteur qui me pousse pour toujours avancer. Également, une pensée chaleureuse à ma mère qui m'a encouragé malgré les distances lointaines et qui a sans cesse cru en moi.

J'en profite par l'occasion pour remercier énormément mon directeur de recherche à la maîtrise Marc Feeley qui m'a initié au domaine de parallélisme et systèmes distribués.

J'apprécie le travail et le perfectionnisme de l'examinateur externe Ali Miri de l'Université d'Ottawa et les membres de jury : Mostapha Aboulhamid (Université de Montréal), Samuel Pierre (École Polytechnique de Montréal) et Denis Cousineau (Université de Montréal) pour leur évaluation pertinente de ma thèse.

Par l'occasion, je remercie tous les professeurs et le personnel avec qui j'avais un contact ainsi que le groupe du support technique et le chef des labos pour leur effort et leur professionnalisme. Je salue mes collègues au labo Étienne, Abdeltaouab, Djohara, Amadou et Faycal pour leur sincère encouragement.

Finalement, j'aimerais exprimer ma gratitude envers le département d'informatique et de recherche opérationnelle (DIRO) de l'Université de Montréal qui m'a offert l'ambiance scientifique qui m'a épanoui.

CHAPITRE I

1 Introduction

L'Internet est sans équivalent le plus grand réseau mondial qui comporte des milliers de sous-réseaux hétérogènes. Le protocole BGP (Border Gateway Protocol) est le protocole de facto de l'Internet pour définir la connectivité entre ces sous-réseaux. Dans un réseau de commutation tel qu'utilisé dans l'Internet, le routage est la procédure de sélection d'un chemin selon lequel les paquets seront transmis pour atteindre leur destination. Le routeur est la machine dédiée qui effectue une telle procédure. De même, un routeur permet le relayage des paquets entre deux réseaux d'espace d'adressage identiques. Dans le cas où il y a deux réseaux avec deux modes d'adressage différents, la conversion de l'adressage implique l'utilisation d'un mécanisme de conversion particulière. Dans ce cas, la machine qui assure l'interconnexion entre les deux réseaux est une passerelle (en anglais, Gateway) qui sert à réunir deux réseaux parfaitement hétérogènes. Le réseau de l'Internet est constitué de sous réseaux appelés domaines ou bien systèmes autonomes (Autonomous Systems, AS). Chaque AS contient un ensemble de routeurs contrôlés par une administration unique. De même chaque système autonome est représenté par un ou plusieurs nœuds (routeurs) de bordure qui communiquent avec d'autres routeurs de bordures correspondants aux autres systèmes autonomes (ASs). Cette hiérarchie a donné naissance aux deux types de protocoles de routage nommés IGP (Interior Gateway Protocol) et EGP (Exterior Gateway Protocol) utilisés dans le réseau de l'Internet. Les protocoles internes aux domaines tels que (ISIS et OSPF), aussi nommés protocoles intradomaines (IGP), assurent la connectivité entre routeurs au sein du même système autonome. Tandis que les protocoles externes ou bien interdomaines (EGP) gèrent l'échange d'informations entre les domaines. Ces informations sont stockées et maintenues

dans une base de données particulière que l'on appelle Table de Routage. La structure de données la plus utilisée pour implémenter la table de routage est le radix-trie [Sklower91]. Dans le contexte de routage au cœur de l'Internet, la zone libre par défaut (Default Free Zone, DFZ) réfère à l'ensemble des routeurs qui ont une table de routage complète désignée par table de routage globale ou simplement *table de routage*.

1.1 Motivation

BGP est le protocole principal qui échange le plus d'informations pour mettre à jour la table de routage. Pour ce faire, il établit des connexions avec d'autres routeurs de voisins pour échanger leurs meilleurs chemins trouvés en temps réel. Les routeurs orientent les trafics selon les informations (routes) contenues dans leurs tables de routage. Une table de routage dans un routeur au cœur de l'Internet stocke des centaines de milliers de routes et ce nombre croît exponentiellement chaque année. Cette inflation incontrôlable de la table de routage suscite l'inquiétude au sein de la communauté scientifique, voire industrielle, sur la capacité physique des routeurs de supporter une telle croissance. De même, la stabilité de l'Internet dépend largement de l'implémentation du protocole BGP au sein de ces routeurs. Et c'est précisément cela qui fera le sujet de notre thèse.

En effet, dans l'architecture des routeurs typiques adoptée par l'*Internet Engineering Task Force* (IETF) [Yang04], un routeur est principalement composé de deux couches : dans la première couche, il y a l'interface (carte) de contrôle (Control Plane ou Controller Card, CC) où réside le processeur de routage. Dans la deuxième couche, il y a l'interface de l'expédition aussi nommée carte de ligne (en anglais, Forward Plane ou Line Card, LC) dans laquelle réside l'engin d'expédition des paquets. Dans la première couche, des différents protocoles de communication tels que BGP (EGP) et OSPF (IGP) coexistent sur le même processeur pour fournir les meilleurs chemins qui seront utilisés dans la couche la plus basse pour la classification et l'expédition des paquets. Le BGP est le protocole principal qui s'exécute sur l'interface de contrôle pour produire et maintenir la table de routage contenant les meilleures routes ainsi que celles qui sont alternatives. Cette table de routage au niveau de l'interface de contrôle est nommée RIB (Routing Information Base) et

parfois BGP RIB pour signifier le rôle principal de BGP dans sa construction. Une version condensée du RIB contenant seulement les meilleurs chemins est périodiquement mise à la disposition de l'interface d'expédition pour qu'elle soit utilisée lors de la classification et de l'expédition des paquets. Dans ce cas, la table est appelée FIB (Forward Information Base). Dans les routeurs au cœur de l'Internet, les deux processeurs aux couches de contrôle et d'expédition doivent fonctionner en parallèle et sans interruption pour assurer l'exactitude de la connectivité entre les sous-réseaux qui sera traduite par une meilleure qualité de service sur le poste de l'usager. Heureusement, la séparation et la duplication des interfaces d'expédition (LCs) telle que décrite dans [Yang04] ont allégé le risque du goulot d'étranglement au niveau de la couche d'expédition en permettant une capacité d'expédition de l'ordre de térabits/s. De l'autre côté, même si le débit au niveau du LC est plus critique et est situé dans l'ordre de gigabits par seconde, le volume du trafic au niveau de l'interface de contrôle, majoritairement dû aux messages échangés par le protocole BGP, demeure significatif et représente un défi pour les routeurs de la prochaine génération. En effet, c'est cette couche (contrôle) qui assure l'exactitude de l'expédition utilisée dans les LCs. La moindre défaillance au niveau de l'interface de contrôle peut perturber l'expédition des paquets au niveau du FIB où le trafic pourra être dirigé à des destinations incorrectes. Également, la façon monolithique dont BGP s'exécute sur cette interface pourrait le transformer en un goulot d'étranglement ce qui expose la couche de contrôle à une forte dégradation de performance si la tendance de croissance du trafic de l'Internet se maintient. Dans cette thèse, nous nous sommes particulièrement intéressés à résoudre les problèmes hérités de la centralisation de BGP au sein des routeurs en proposant une architecture distribuée de BGP sur la couche de contrôle qui élimine les inconvénients de la centralisation et assure un degré assez élevé de performance, d'extensibilité et de fiabilité des routeurs. Ce qui nous motive à réaliser cette thèse est que le principe de répartition dans les systèmes distribués fournit des avantages considérables prouvés dans les NoW (Network of Workstations). Ces idées sont maintenant envisageables et applicables dans les routeurs grâce à l'évolution technologique des voies de communications internes (Switch Fabric) dans les routeurs où la latence de communication est devenue basse (quelques nanosecondes). De même, la capacité du calcul des processeurs est assez élevée pour favoriser l'exploitation du parallélisme entre les processeurs distribués. Tous ces facteurs

nous permettent de croire qu'il est possible de conceptualiser un routeur distribué pas seulement au niveau matériel, comme c'est le cas présentement, mais aussi au niveau logiciel. Le gain paraîtra concrètement au niveau de la performance, de la scalabilité et de la fiabilité.

1.2 Description précise de la problématique

Les inconvénients sentis dans l'Internet et qui sont dus au protocole BGP sont souvent issus soit de la manière dont BGP est conçu comme une variante de la classe des protocoles vecteur distance (Distance Vector Protocol), soit de la façon monolithique dont ce protocole est implémenté dans la couche de contrôle. Or, les défis qui se manifestent actuellement au niveau de la couche de contrôle des routeurs sont essentiellement : la performance, la scalabilité et la fiabilité.

1) Performance : au niveau de la couche de contrôle, BGP est le protocole qui consomme le plus de ressources. Actuellement, le protocole BGP échange des dizaines de messages par seconde pour mettre à jour des centaines de routes dans la table de routage [RouteViews]. De plus, le protocole peut atteindre des centaines de connexions (TCP) avec des routeurs voisins pour réaliser cet échange. Avec la croissance continue de l'Internet, ces nombres vont sûrement continuer à croître. Cette croissance prévue de trafic du BGP risque de nuire aux autres protocoles qui s'exécutent sur la même interface de contrôle et pourra par la suite produire une dégradation de la performance au niveau du routeur tout entier.

2) Scalabilié : la scalabilité de BGP sur la couche de contrôle est vue par la capacité maximale qu'une carte de contrôle peut supporter pour maintenir un nombre énorme de routes et le nombre de connexions avec d'autres BGP voisins. En fait, la croissance exceptionnelle de l'Internet est directement reliée à la croissance de la table de routage de BGP que ce soit au niveau du FIB ou au niveau du RIB. Les dernières données statistiques sur cette croissance montrent que la taille de la table de routage qui contenait autour de 50,000 préfixes (routes différentes) en 2001 se chiffre à 350,000 préfixes en 2010 et le nombre des systèmes autonomes se situe autour de 40,000 [APNIC]. Or, la croissance de la table de routage pourrait même atteindre un 1 million de préfixes dans quelques années. De même, le nombre des systèmes autonomes augmente linéairement et ceci a un impact direct

sur la scalabilité de BGP au niveau de la carte de contrôle. Pour palier aux conséquences de cette croissance de l'Internet, l'introduction en 1993 d'un mécanisme d'agrégation de plusieurs préfixes en un seul adresse de routage sans classe (en anglais, Classless Inter-Domain Routing) CIDR a pu remédier l'effet de croissance des tables de routage. Malgré que cette approche ait réussi à ralentir la croissance des tables de routage pour quelques années, la croissance a repris sa forme exponentielle dès le début de ce siècle [RouteViews]. Pour contrer le problème de limitation des ressources dans les routeurs d'aujourd'hui, les compagnies de télécom se basent sur le remplacement périodique des routeurs avec d'autres nouveaux de la récente technologie offrant une meilleure capacité physique. Dans ce cas, la durée de vie moyenne d'un routeur peut varier de deux à quatre ans. Malgré que ce processus ait prouvé de bons résultats en permettant à l'Internet de fonctionner toutes ces années, il est clair que ce processus ne sera plus viable à long terme d'autant plus que c'est une procédure coûteuse et qui n'est pas écologique. Il suffit de compter le nombre des routeurs à recycler annuellement.

3) Fiabilité : BGP est une variante de la classe des protocoles distance vecteur (en anglais, Distance Vector Protocols) qui fonctionne selon la propagation des routes avec ses voisins. Cette conception est affectée par l'oscillation rapide des routes (en anglais, Route Flapping). Une route oscillante est produite généralement d'un arrêt suivi d'un démarrage d'une instance BGP et peut être causée soit par une interface de contrôle défectueuse dans un routeur (attaque malicieuse, mauvaise configuration, panne, etc.) ou simplement par un routeur en phase de redémarrage.

Suite à cette oscillation, des annonces des routes inaccessibles seront propagées à tous les routeurs BGP voisins en les obligeant à appliquer un algorithme complexe pour recalculer leurs meilleures routes, à mettre à jour leur FIB et propager cette information d'une façon récursive jusqu'à ce que cette nouvelle information soit atteinte dans tous les routeurs qu'ils l'utilisent. Or, une répétition fréquente de cette procédure (Route Flapping) et à grande échelle pourrait nuire à la stabilité de l'Internet.

1.3 Systèmes distribués et parallélisme

Les systèmes distribués ont connu une popularité marquante dans la dernière décennie grâce à leur capacité de fournir un avantage considérable vis-à-vis les architectures centralisées ou monolithiques. En effet, l'amélioration de la vitesse des processeurs, l'augmentation de la bande passante et la baisse considérable de la latence dans les réseaux ont contribué à une amélioration éprouvée dans bien des applications distribuées. Ces dernières ont tiré profit de la séparation des tâches sur plusieurs processeurs que les systèmes distribués offrent pour exploiter le parallélisme et améliorer la performance. De même, l'utilité des systèmes distribués peut clairement paraître dans les applications à mission critique pour augmenter leur niveau de redondance et par conséquent leur tolérance aux pannes. Pourtant, il est vrai que les avantages offerts par les systèmes distribués sont flagrants, il est aussi vrai que l'implémentation d'une architecture distribuée versus une application centralisée engendre des coûts supplémentaires. Ces coûts peuvent parfois dépasser les bénéfices, dus aux lacunes héritées dans les réseaux tels que les messages perdus, la latence, le manque d'une horloge globale, la gestion de la synchronisation, etc. D'autant plus que la nature du bien des algorithmes n'accepte pas la distribution ni la parallélisation.

1.4 Objectif et Contributions

L'objectif de cette thèse est de conceptualiser et de réaliser une architecture de BGP complètement distribuée et tolérante aux pannes sans modifier la sémantique du protocole de BGP tel que décrit dans [Rekhter06]. Ceci sera fait en trois dimensions. Premièrement, en partitionnant les fonctionnalités de BGP sur plusieurs interfaces de contrôle, deuxièmement en ajoutant un mécanisme de tolérance aux pannes et troisièmement en divisant la table de routage sur plusieurs cartes de contrôles physiquement distribuées. L'architecture proposée sera facilement déployée d'une façon graduelle dans les réseaux de fournisseurs Internet (ISP, Internet Service Provider) sans provoquer une disruption au niveau de la fonctionnalité quotidienne de l'Internet. Notre but ultime de cette thèse est de fournir une architecture de BGP qui maximise la performance, l'extensibilité et la fiabilité

du protocole BGP qui élimine le point individuel de faute (en anglais, Single-Point of Failure). Pour atteindre notre objectif et résoudre les problèmes de limitation liés à la centralisation de BGP dans les routeurs au cœur de l'Internet, nous avons divisé le travail dans cette thèse en trois parties.

Dans la première partie, nous proposons une architecture extensible de BGP basée sur une distribution de grappe de calcul (en anglais, Cluster). Dans cette architecture, au lieu de faire exécuter le protocole BGP sur une interface de contrôle d'une façon centralisée, nous partitionnons les fonctionnalités de BGP sur plusieurs interfaces d'une façon client-serveur pour créer un ensemble de grappes de calcul au niveau des cartes de contrôle. Cette distribution permet de fournir une meilleure performance et une meilleure extensibilité tout en éliminant la monolithicité d'exécution du protocole de BGP sur la couche de contrôle. Cette contribution assure un déploiement transparent aux autres routeurs voisins en permettant à BGP de leur fournir le même comportement externe que celui fourni dans une version monolithique.

Dans la deuxième contribution, nous avons élaboré un mécanisme de tolérance aux pannes et avons démontré mathématiquement que l'architecture résultante produira un gain de fiabilité considérable par rapport aux architectures classiques client-serveur centralisées. Dans la troisième contribution, nous avons conçu et développé une méthode de partition de la table de routage de BGP que nous appelions DRTP (Distributed Routing Table Paritioning). En effet, avec la première architecture décrite, nous avons trouvé que même si l'extensibilité au niveau de BGP a beaucoup été améliorée par rapport à la version monolithique, la table de routage RIB est entièrement dupliquée sur plusieurs interfaces. Pour aller plus loin dans la distribution, DRTP sert à partitionner la structure des données radix-tree de la table de routage sur plusieurs cartes de contrôle. La contribution majeure dans cette approche est le partitionnement de la table de routage sur plusieurs cartes de contrôle physiquement distribuées et la parallélisation de l'algorithme BMP (Best Match Prefix) nécessaire pour effectuer la recherche dans la table de routage.

La division de la thèse est faite par articles de la manière suivante : le deuxième chapitre traite la revue de littérature et les concepts de base de BGP. Le troisième chapitre décrit la première contribution qui est l'architecture distribuée de BGP. Le quatrième chapitre décrit

la conception d'un mécanisme de fiabilité applicable sur l'architecture distribuée réalisée dans le chapitre 3 et une évaluation analytique de fiabilité pour l'architecture résultante. Le cinquième chapitre présente DRTP qui est une méthode de partitionnement basée sur le traitement parallèle et la distribution pour partitionner la table de routage sur plusieurs machines serveur parallèles. Le chapitre 6 est la conclusion de cette thèse.

1.5 Articles publiés/soumis durant cette thèse

1. Hamzeh, W., A., Hafid. "DRTP: A distributed parallel approach for BGP routing table partitioning." Submitted to Journal of Computer Networks, 2011.
2. Hamzeh, W., A., Hafid. "A Highly Reliable Cluster-Based Distributed BGP Architecture with Graceful Restart Capability." Submitted to Journal of Computer Communications, 2011.
3. Hamzeh W., A., Hafid. "A scalable cluster distributed BGP architecture for next generation routers." *IEEE LCN.* Zurich,Switzerland, 2009. 161-168.
4. Hamzeh W., A., Hafid. "A distributed parallel approach for BGP routing table partitioning in next generation routers." *IEEE LCN.* Denver, Colorado, USA 2010. 480-487.

CHAPITRE II

2 État de l'Art

2.1 Concepts de bases des routeurs

Avec les derniers avancements technologiques des réseaux, la capacité de traitement des routeurs pourrait devenir le goulot d'étranglement qui pourrait nuire à l'évolution de l'Internet. En effet, la vitesse de communication est montée de 100 Mb/s dans les années 90 pour dépasser les 100 Gbps au moment présent. Grâce à cette capacité, le routeur doit traiter plusieurs millions de paquets par seconde dans sa couche d'expédition pour qu'il supporte la croissance du trafic dans l'Internet. Selon le RFC [Yang04] d'IETF, un routeur pourrait être conçu comme un ensemble de protocoles qui résident sur des cartes de contrôle et des entités d'expédition qui seront dupliquées sur plusieurs cartes de ligne (en anglais, Line Cards, LCs). L'architecture de base d'un routeur typique dans sa forme simplifiée (Fig. 2.1) est constituée de trois composantes : la première est la carte de contrôle au niveau de la couche de contrôle qui héberge un processeur et une mémoire assez puissante pour stocker une grande base de données qui est la table de routage RIB (en anglais, Routing Information Base). La deuxième composante inclut les cartes de ligne dans la couche d'expédition. Chaque carte de ligne héberge une version contenant seulement les routes actives utilisables dans l'expédition. Cette table de routage s'appelle FIB (Forward Information Base). Et la troisième et dernière composante, c'est le support de communication connu sous le nom « Switch Fabric ». Au niveau de la couche de contrôle, le processeur effectue des fonctions comme le calcul des chemins, l'entretien de la table de routage et la mise à jour de l'information vers la couche d'expédition. Pour ce faire, le processeur des routes exécute (au niveau de la carte de contrôle) des protocoles de communication tels que BGP et OSPF pour construire et maintenir la table de routage. Au niveau de la couche d'expédition, les cartes de ligne (LCs) se composent d'engins qui

exécutent l'expédition des paquets aux ports d'arrivées (ingress) et de sorties (egress) en consultant la table de routage stockée dans le FIB. Le « Switch Fabric » est le support de communication pour transférer des paquets de contrôle et des données entre les cartes au sein d'un routeur. Les fonctionnalités de base dans un routeur IP peuvent être classées en trois catégories : (1) le traitement des routes apprises via les protocoles de communication ; (2) l'expédition des paquets au niveau des cartes de ligne (LCs); (3) les services spéciaux tels que la synchronisation RIB-FIB, la vérification de la consistance et le module CLI (c.-à-d. Command Line Interface).

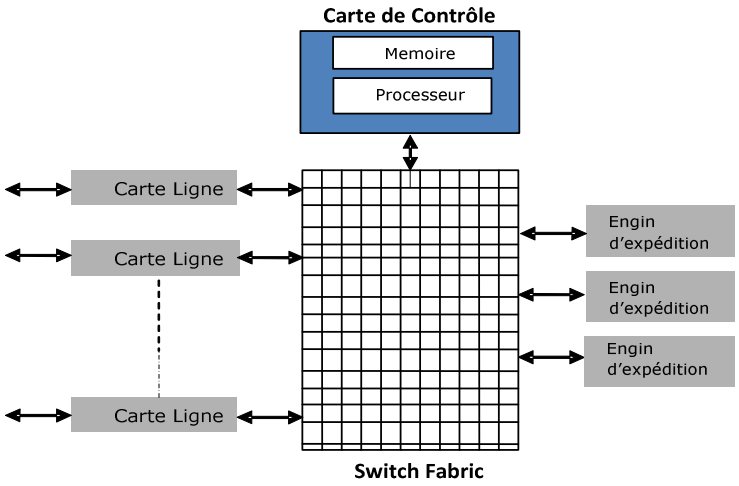


Figure 2.1 Architecture de base d'un routeur

2.1.1 Prochaine génération des routeurs distribués

Malgré la forte croissance du trafic qui pèse sur son infrastructure, l'Internet a suivi un long chemin d'évolution lui permettant de survivre. Durant la dernière décennie, le concept de créer un routeur dont l'architecture est basée sur le principe de grappe de calcul (cluster), initialement introduit dans [Halabi], est devenu réalisable. Ce concept a attiré l'intérêt commercial des compagnies de télécommunications pour concevoir et commercialiser un routeur multi-chassis basé sur ce même concept ([Cisco] [Juniper]).

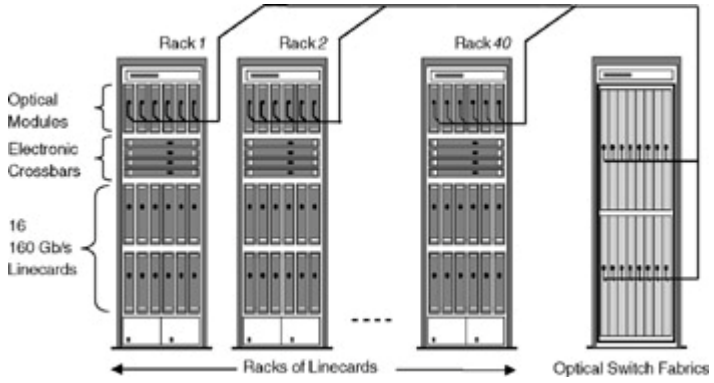


Figure 2.2 Routeur de la prochaine génération (100Tb/s) [Kesl03]

L'idée principale pour concevoir un routeur de la prochaine génération (Fig. 2.2) satisfaisant les contraintes imposées par [Baker95] est de permettre l'extensibilité des routeurs par le moyen de plusieurs étagères (chassis) physiquement distribuées dont chacune regroupe plusieurs cartes de ligne. Ces cartes fonctionnent en parallèle et sont toutes connectées à un « Switch Fabric» optique pour effectuer la classification et l'expédition des paquets. À titre d'exemple, si chaque carte de ligne (LC) peut fournir un débit de 160GB/s, la capacité globale d'un routeur qui est constituée de plusieurs étagères avec plusieurs cartes de ligne pourra atteindre une capacité d'expédition totale située dans l'ordre de 100 Tb/s. L'avantage de cette séparation physique est apparent dans la capacité du calcul accompli au niveau des cartes de ligne ainsi qu'à la fiabilité au niveau de l'expédition. Ceci dit, une faute dans une carte de ligne n'empêchera pas l'expédition dans les autres cartes ce qui permettra au routeur d'effectuer le routage sans arrêt (en anglais, Non Stop Routing) au niveau de la couche d'expédition. Un prototype de ce genre de routeurs est réalisé par les compagnies télécoms sous le nom de (Carrier Grade Router) et est devenu le standard des routeurs au cœur de l'Internet. Par contre, cette architecture ne tient pas compte de la distribution dans les cartes de contrôle puisque l'exécution des protocoles de communication est toujours centralisée. En fait, la seule distribution logicielle implémentée est la redondance pour assurer une fiabilité via une réplication passive. Étant donnée la limitation de distribution qu'engendrent les routeurs quotidiens, un des défis que pose la conception d'une nouvelle architecture de routeurs pour la prochaine génération est la distribution logicielle au niveau des cartes de contrôle.

2.2 BGP (Border Gateway Protocol)

Décrit dans [Rekhter06], BGP est le protocole de facto de l'Internet pour le routage inter-domaine. Il est une variant de classe des protocoles vecteur distance. Il s'exécute dans la quatrième couche de l'architecture (OSI) en utilisant TCP comme un moyen de communication sous-jacent. BGP est un protocole de routage dont la communication se fait via TCP pour assurer une communication fiable avec ses voisins. Ce choix reflète l'importance de la fiabilité de communication entre les routeurs BGP. En fait, le but de BGP est de créer des routes interdomaines sans boucles qui seront stockées dans le RIB dans la couche de contrôle. Par la suite, les meilleures routes seront mises en place dans le FIB (Forward Information Base) pour effectuer l'expédition des paquets dans les cartes de ligne. Pour garder une version récente et valide des routes existantes, BGP échange avec les autres routeurs BGP voisins cinq types de messages que voici :

1- OPEN
2- NOTIFICATION
3- KEEPALIVE
4- UPDATE
5- Route Refresh

BGP utilise le message OPEN pour déclarer à son voisin son intérêt d'ouvrir une session BGP. Le port réservé à BGP pour établir la connexion TCP comme définie dans le RFC 4271 [Rekhter06] est le port 179. D'une façon logique, l'établissement de la connexion se fait selon le paradigme client-serveur. Une fois la connexion établie, les données peuvent circuler entre les deux routeurs BGP dans les deux sens. Le message NOTIFICATION est utilisé pour signaler une erreur qui résulte d'une déconnexion entre les deux routeurs BGP. L'établissement et la terminaison d'une connexion en BGP sont gérés par le module de Machine à État Fini (en anglais, Finite State Machine, FSM) qui est une partie intégrante du protocole.

Malgré la fiabilité du protocole TCP, il se peut tout de même qu'en pratique une connexion TCP soit perdue sans que les deux participants aux deux bouts de la connexion ne soient avertis. Pour cela, BGP ne compte pas sur les mécanismes internes de TCP pour détecter la perte d'une connexion. Or, BGP transmet régulièrement des messages KEEPALIVE dans un intervalle défini (par défaut trois secondes) pour signaler son existence à son

voisin. À défaut de recevoir trois messages KEEPALIVE consécutifs, ceci représente une preuve que la connexion a été perdue entre les deux routeurs BGP. Une fois la connexion entre deux routeurs BGP voisins rétablie, les deux routeurs de BGP commencent à échanger du trafic via des messages de type UPDATE contenant les routes qui sont accessibles à partir de leur routeur. Un message UPDATE est le message le plus important et le plus utilisé dans BGP puisqu'il sert à envoyer les routes qui sont accessibles et celles qui ne le sont pas. Rappelons-nous qu'une route est définie par un préfixe (adresse CIDR) de destination et des attributs qui la caractérisent. Les préfixes qui sont transmis via les messages UPDATE sont encapsulés dans une structure appelée NLRI (Network Layer Reachability Information) qui fait partie du message UPDATE. Le taux du trafic des messages UPDATE pourrait atteindre des dizaines de messages par seconde pour mettre à jour des centaines de préfixes dans un RIB [Labovitz99]. Lorsqu'il s'agit d'une nouvelle route accessible par un routeur, le message UPDATE fait référence à une annonce (ADVERTISMENT). Si la route contenue dans le message UPDATE est inaccessible, le message UPDATE désigne une suppression (WITHDRAW) d'une route. Un exemple de message UPDATE d'un format lisible prendra la forme suivante :

Temps : 25/12/2010 1:30:36
Type : BGP/Message/Update
De : 202.12.28.99 AS4727
À : 193.0.0.1 AS12645
AS_PATH : 4727 2800 7960 4380 7570
NEXT_HOP : 202.12.28.99
ORIGINE : EGP
Annonce: 192.207.156.0/24,192.207.0.0/16,64.168.9.0/16

Figure 2.3 Message UPDATE d'un format lisible

Comme démontré dans la Fig. 2.3, le contenu du message indique que BGP a bien reçu de son voisin situé à l'adresse 202.12.28.99 et dont le numéro du système autonome est AS4777 un message UPDATE. Ce message contient un ensemble de préfixes de

destination encapsulés dans le champ « Annonce » et des attributs communs pour ces préfixes dans les autres champs tels que : (AS_PATH, NEXT_HOP, ORIGIN). Le but du message est d'annoncer la mise à jour dans la table de routage des adresses (préfixes) qui figurent dans le champ « Annonce ». Le rôle de BGP est de vérifier l'exactitude de l'information reçue et ensuite de prendre la décision si la route reçue représente un meilleur chemin pour expédier du trafic vers la destination. Les informations contenues dans la ligne AS_PATH sont primordiales à BGP. Avec cette information, BGP est capable de détecter et éliminer les boucles qui peuvent apparaître entre les systèmes autonomes. Cette liste d'informations indique en ordre inverse tous les systèmes autonomes parcourus par le message UPDATE avant d'atteindre le routeur courant. Le champ NEXT_HOP définit l'adresse du routeur de prochain saut qui peut livrer le trafic vers l'adresse de destination incluse dans le message UPDATE.

2.2.1 Fonctionnement de BGP et détection des boucles

À chaque réception d'un message UPDATE, BGP applique un mécanisme de vérification et d'authentification (champ source et destination) avant d'accepter le message et son contenu. Si après vérification le message est jugé valide, alors BGP procède à l'application du mécanisme de détection des boucles. Ceci est fait en analysant la liste des systèmes autonomes inclus dans le champ AS_PATH. Si cette liste contient le numéro d'un système autonome du routeur récepteur ou bien une répétition de deux numéros, alors BGP vient de détecter une boucle et le message est automatiquement refusé. Comme il est mentionné dans Fig. 2.4, lorsque le message a passé avec succès le mécanisme d'authentification et la détection des boucles, BGP applique des règles de filtrage sur les adresses de destinations annoncées dans le message UPDATE. Il est important de préciser que ces règles décrivent la politique administrative dans chaque AS. Les règles de filtration sont essentiellement mises par l'administrateur système. À titre d'exemple, même si le message BGP en Fig. 2.3 représente un message valide (c.-à-d. sans boucles) qui encapsule une liste des adresses de destinations valides, les règles de filtrage peuvent refuser des routes et en accepter d'autres selon le choix administratif. Dans ce cas, seulement les routes acceptées seront traitées selon la procédure de sélection des meilleurs chemins (Decision

Process). Le but de cette procédure est de comparer chaque route reçue avec toutes les routes similaires existantes dans la table de routage qui pointent sur la même adresse de destination pour ensuite choisir la meilleure route. Si la route reçue est considérée gagnante, elle sera ensuite utilisée dans le FIB pour l'expédition.

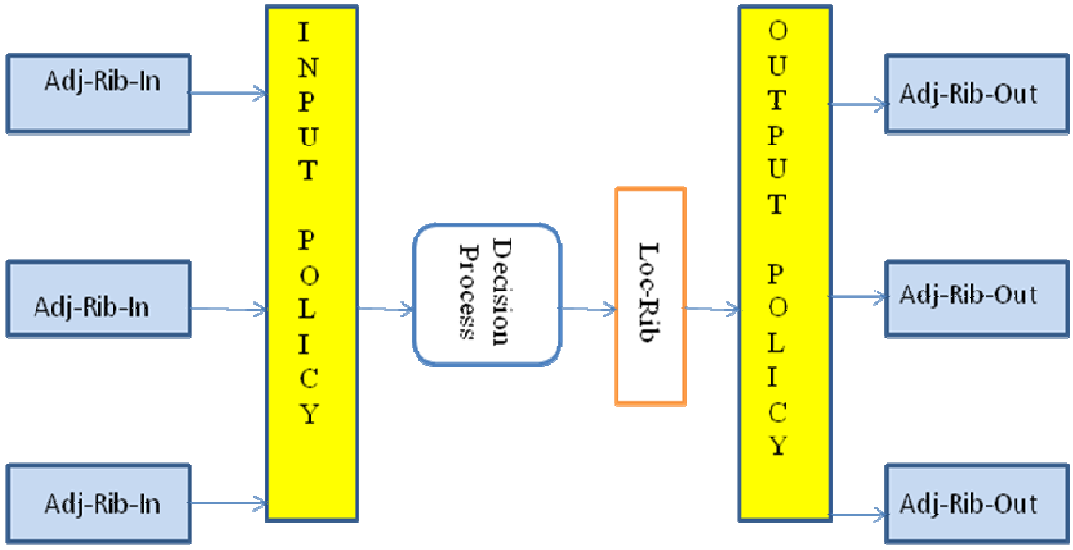


Figure 2.4 Description abstraite de l'implémentation logicielle de BGP

Lorsque les routes sont sélectionnées comme meilleurs chemins aux préfixes annoncés, elles seront mises dans une base de données appelée Loc-Rib qui sera utilisée par le FIB dans la couche d'expédition. Sinon, les routes seront considérées comme des routes alternatives et seront stockées dans les Adj-Rib-In pour une utilisation future. Ensuite, pour chaque route considérée comme meilleur chemin, BGP applique un ensemble de règles de filtrage associées à chaque système autonome du BGP voisin pour savoir s'il est permissible de lui envoyer ces routes ou non. Si c'est le cas, alors la nouvelle route sera stockée dans une base de données Adj-Rib-Out et un message UPDATE contenant cette route sera envoyé au BGP voisin. Suite à cette procédure, trois bases de données feront référence au RIB de BGP. Ces bases sont : Adj_Rib_In, Adj_Rib_Out et Loc-Rib. Même

si le modèle conceptuel du protocole tel que décrit dans [Rekhter06] exige trois bases de données, l'implémentation typique de BGP ([Zebra], [Quagga]) n'est effectuée qu'avec une seule radix-tree [Sklower91] ayant des pointeurs sur les trois bases mentionnées.

2.2.2 Sélection et propagation des meilleurs chemins

BGP est une variante de la classe des protocoles vecteurs distance. La nature de l'Internet exige que chaque routeur BGP soit connecté à plusieurs routeurs voisins. Ceci implique qu'un routeur BGP peut fréquemment recevoir de multiples chemins pour une même destination. Le rôle de BGP est donc de continuellement choisir le meilleur chemin parmi ceux-ci pour mettre à jour son RIB et par ailleurs le FIB. Pour cette fin, BGP applique un algorithme de sélection des chemins en utilisant des attributs spécifiques dont chacun est une caractéristique qui sert à guider BGP pour bien choisir le meilleur chemin d'une façon déterministe [Rekhter06].

2.3 Travaux existants sur le BGP distribué

Les travaux existants peuvent être classés en trois catégories reliées au travail traité dans cette thèse. En premier lieu, les travaux qui se concentrent sur la distribution de BGP. En deuxième lieu, les travaux sur le partitionnement de la table de routage et finalement, les travaux effectués touchant la fiabilité de BGP.

2.3.1 BGP distribué

Au sein des routeurs, l'implémentation monolithique de BGP au niveau de la carte de contrôle engendre des limites de scalabilité héritées telles que le nombre de connexions supportées et la taille maximale de la table de routage. Dans la majorité des routeurs actuels, le nombre de connexions maximales qui peuvent être supportées est limité à 4000 [Cisco]. Cette limitation est essentiellement due au nombre de connexions TCP maximal que peut supporter le système d'exploitation dans la carte de contrôle. De même, la taille de la table de routage dans la carte de contrôle (RIB) pourra dépasser dans les prochaines années la taille de la mémoire physique limitée à 4GB. Pour surmonter les limitations de

BGP au niveau de l'implémentation, les travaux existants focalisent sur la distribution et la parallélisation. Les travaux [Kun06] et [Zhang05] se sont concentrés sur une architecture distribuée des routeurs sans modifier la sémantique de BGP. Les architectures distribuées les plus communes des routeurs sont basées sur la notion du détachement de la couche de contrôle de la couche d'expédition et de la séparation matériel. Cette méthode basée sur la séparation conduit à la création d'une architecture multi-chassis physiquement distribuée [Kesl03]. Selon cette classe d'architectures, le protocole BGP s'exécute sur une même carte de contrôle d'une façon centralisée. Hosgand et al. [Hagsand05] ont conçu et implémenté une architecture distribuée du routeur au niveau logiciel en se basant sur la décomposition des fonctionnalités des protocoles entre autres BGP. Par exemple, quelques fonctions de BGP, comme le module FSM qui gère les connexions avec les routeurs voisins, se sont détachées de la carte de contrôle pour ensuite être exécutées sur la carte de ligne. Selon son approche, ce module est exécuté sur la carte ligne et une couche logicielle définie par un protocole interne pourra ainsi lier les deux composantes du BGP. La communication interne entre les différentes fonctions distribuées sur la carte de contrôle et les cartes de ligne se fait via un protocole de communication spécial qui s'appelle FORZ. Kun et al. [Kun06] ont proposé une approche théorique pour distribuer BGP d'une manière arborescente. Zhang et al. [Zhang05] ont introduit une architecture distribuée de BGP basée sur la communication par agent. Dans cette approche, plusieurs entités de BGP s'exécutent d'une façon distribuée et échangent les meilleures routes. De même, les études [Cavalli06], [VYATTA] et [See08] existantes dans la littérature visent à améliorer la scalabilité en concevant BGP d'une fonction modulaire. XORP [XORP] est une implémentation d'un routeur où l'accent est mis sur l'extensibilité. Le travail présenté dans [Cavalli06] démontrait que les auteurs voulaient réduire la complexité de BGP en détachant les fonctionnalités de la maintenance des sessions du protocole de BGP. Dans [See08], le routeur est composé d'un ensemble de cartes de contrôle où réside sur chaque carte une instance du protocole de BGP. Les routes apprises par chaque instance sont diffusées à toutes les autres instances qui s'exécutent sur les différentes cartes. Ce travail présenté dans un brevet met l'accent sur la duplication sans tenir compte des problèmes de consistance qui peuvent se produire durant l'exécution de BGP.

2.3.2 Partitionnement de la table de routage

La table de routage est un point crucial relié à la scalabilité de BGP. La façon la plus intuitive pour augmenter la scalabilité des tables de routages consiste à partitionner la table sur plusieurs cartes. Les travaux existants dans la littérature visent à diviser la table de routage FIB dans la couche d'expédition pour augmenter la vitesse de recherche de préfixes. On peut classer les travaux de division des tables de routage selon deux catégories : logique et physique. Une méthode logique de partitionnement de la table de routage sert à créer plusieurs partitions sur le même nœud qui est la carte de ligne (LC) pour augmenter la capacité de recherche de préfixes (En anglais, Best Matching Prefix) au moment de l'expédition tandis qu'une méthode de partition physique divise la table sur plusieurs cartes physiquement distribuées. Scudder et al. [Scudder09] proposent une méthode de partition physique du RIB sur la carte de contrôle selon le rang du préfixe. L'idée initiale consiste à diviser la table de routage sur plusieurs serveurs. Considérons le cas le plus simple où la division se fait sur deux serveurs seulement. Le premier serveur répondra aux requêtes visant les préfixes dont le bit le plus fort est de 0 (c.-à-d., 0.0.0.0/1) et un autre serveur qui répond aux requêtes visant les préfixes dont le bit le plus fort, est de 1(c.-à-d. 128.0.0.0/1). Le processus client envoie la requête selon le bit le plus fort du préfixe au serveur correspondant. L'agrégation virtuelle [Ballani09] est une solution à long terme pour partitionner la table de routage FIB sur plusieurs routeurs dans un même système autonome. Akhbarizadeh et al. [Akhbari05] a présenté une technique de partitionnement sur plusieurs nœuds afin d'effectuer la recherche en parallèle. Même si cette approche augmente la performance en termes de recherche, toutes les partitions sont dupliquées sur tous les nœuds. Or, la taille de la table de routage n'est pas prise en considération. TZENG [Tzeng06] a proposé une technique appelée SPAL pour faire le partitionnement de la table de routage sur plusieurs cartes ligne. Cette approche tient compte de la scalabilité en termes de taille de la table en divisant celle-ci selon des bits pivots dans les préfixes. Ces bits sont calculés au moment du démarrage du routeur pour faire une division optimale de la table de routage. Le résultat du partitionnement est vu par l'augmentation de la vitesse de recherche et la scalabilité de la table de routage en fonction de la taille. Waldvogel [Waldvogel01] a proposé une nouvelle approche pour partitionner la

table de routage au niveau de la carte de ligne en utilisant les tableaux de hachages comme structure de données pour représenter la table de routage. La technique de Waldvogel vise à augmenter la rapidité de la recherche en transformant la recherche BMP en une recherche binaire entre les différentes partitions sur un même nœud physique. Pour cette fin, la technique introduit des faux préfixes « marqueurs » qui servent à guider la recherche BMP (Best Match Prefix) entre les partitions. La vitesse de recherche a été améliorée et devenue O (logW); tandis que la rapidité de la recherche est dans l'ordre de O(W) où W est la longueur du préfixe dans le cas d'une table de routage radix-tree [sklower91]. Cette amélioration de la rapidité de recherche est réalisée au détriment d'une augmentation de la table de routage causé par l'ajout des marqueurs. En DRTP [Hamzeh10], le partitionnement de la table de routage se fait au niveau la carte de contrôle. DRTP augmente la scalabilité de la table de routage tout en exploitant le parallélisme pour augmenter la performance. L'avantage de DRTP est la scalabilité en termes de mémoire de la table de routage et de la parallélisation de l'algorithme de BMP qui sera effectivement traduit par une amélioration de la recherche BMP. La table 2.1 récapitule une comparaison sur les différentes techniques de partitionnement au niveau de la performance selon la vitesse de la recherche, la scalabilité de la table de routage et le parallélisme.

TABLE 2.1: COMPARAISON DES TRAVAUX SUR LE PARTITIONNEMENT DE LA TABLE DE ROUTAGE.

	Scalabilité de mémoire de la table de routage	Performance	Parallélisme
Akhbarizadeh		X	X
Tzeng	X	X	
Waldvogel		X	
DRTP [Hamzeh10]	X	X	X

2.3.3 Fiabilité

Les travaux effectués sur la fiabilité sont classés en deux catégories. Les travaux de la première catégorie de [Cisco] et [Juniper] sont basés sur l'utilisation de la redondance avec l'adoption de techniques telles que le Multihoming [Abley05] et le « BGP Graceful restart » [Sangli07]. Ces travaux ne nécessitent aucune modification du protocole BGP pour

effectuer l'implémentation. La deuxième catégorie regroupe les travaux à long terme qui nécessitent un changement dans la sémantique du protocole BGP [Kushman07]. L'idée est de créer des tunnels entre les routeurs dans un système autonome pour rediriger le trafic en cas de pannes.

2.4 Récapitulation

Dans la majorité des travaux existants sur la distribution de BGP au niveau de la couche de contrôle, il est apparent qu'il y ait un consensus à réaliser une architecture distribuée au sein des routeurs de la future génération. Cette architecture sera conceptualisée d'une façon modulaire en éliminant toute centralisation telle qu'elle existe dans les architectures monolithiques. Malgré que la tendance générale vise la distribution et la modularité, il est clair que les travaux existants n'ont pas réussi à satisfaire toutes les contraintes concernant la performance, la scalabilité et la fiabilité dans un même travail. Or, la distinction de ce travail réside dans sa capacité de produire une architecture distribuée en satisfaisant les critères de performance, de scalabilité et de fiabilité sans changer la structure du protocole de BGP tel que décrit dans le RFC 4271 [Rekhter06].

CHAPITRE III

3 A Scalable Parallel Cluster Distributed BGP Architecture for Next Generation Routers

Wissam Hamzeh, Abdelhakim Hafid

Abstract—In classical monolithic router architecture, the Border Gateway Protocol (BGP) engine is implemented as a multi-process centralized function within the controller entity. This architecture does not scale well and its performance decreases when the load increases, forcing multiple processes to compete for the same controller processor. In addition only a limited number of peer connections can be handled. In this paper, we propose a novel scalable distributed architecture for the BGP engine without modifying the core of the BGP protocol as defined in RFC 4271. The proposed architecture is designed according to the "Master-Slave" task separation along with the replication of the Routing Information Base (RIB) on multiple controller cards. In addition, we design and implement a new consistency algorithm for the RIB replication. Our consistency algorithm does not impose any delay on the processing of BGP Updates to perform events ordering. Simulations show an acceptable trade-off between the scalability to a large number of peer sessions and the overhead caused by the communication latency. Furthermore, simulations show that the proposed architecture handily outperforms the actual BGP processing capacity.

[**Status**]: The ideas presented in this paper are largely based on the following published paper: "A scalable cluster distributed BGP architecture for next generation router", IEEE LCN 2009, pp. 161-168. Zurich, Switzerland.

3.1 Introduction

According to the IETF ForCES RFC [20] a router can be designed as a set of modules residing on controller entities distributed into controller cards and forwarding entities distributed into line cards. While line cards are duplicated to ensure high rate of forwarding, the controller card still runs a set of modules that interact to supply the FIB (Forward Information Base) with the right reachability information. As shown in Fig. 3.1, the role of BGP in a monolithic architecture depends largely on the Routing Information Manager (RIM) that is responsible for managing its Routing Information Base (RIB) and updating the FIB in the line cards (LCs). Any two routers forming a TCP connection to exchange BGP routing information are called peers or neighbours. From a conceptual point of view the BGP protocol is composed of four main modules:

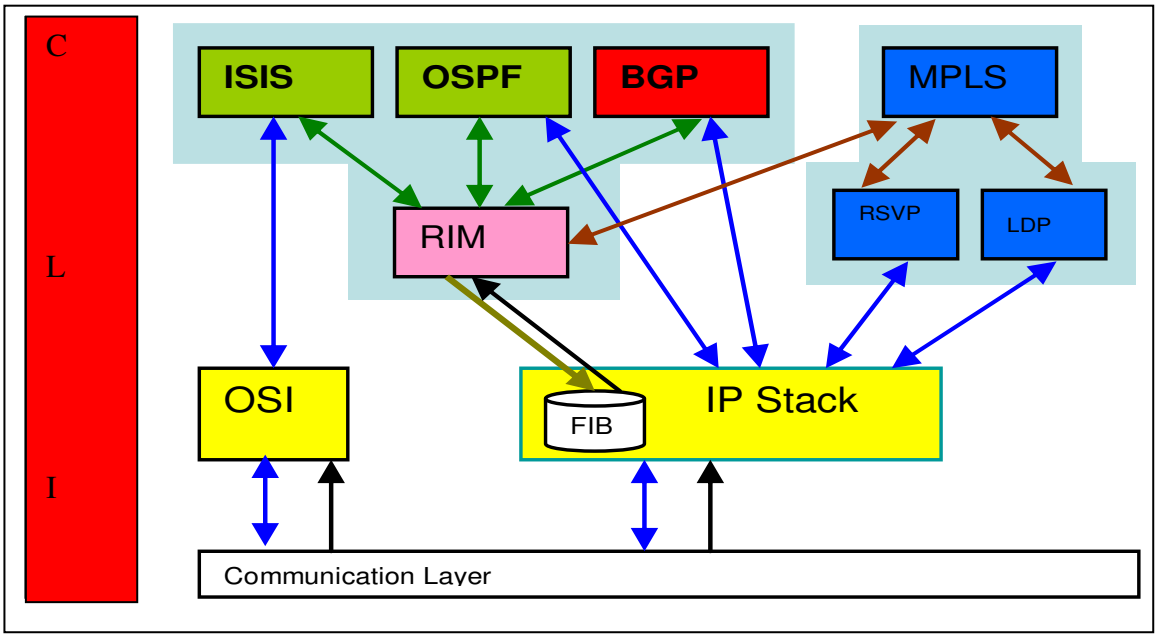


Figure 3.1 Classical Router monolithic architecture

(1) Finite State Machine (FSM): responsible for maintaining the BGP peering sessions i.e. connections state, keepAlive messages, timers etc.; (2) Inbound policy: responsible for filtering the incoming learned routes from adjacent peers according to the inbound administrative policy; (3) Routing Information Manager (RIM): Performs a path selection algorithm on all incoming routes; it selects the best path, stores it in the RIB and updates

the FIB; and (4) Outbound policy: applies the outbound policy on best selected paths before sending them to adjacent peers. BGP scalability can be evaluated in terms of the number of peer sessions and the number of routes that can be supported. BGP has proven to support, a maximum of few hundreds of peers and to maintain well over 100,000 routes [21]. The dramatic growth of the routing table decreases the packet forwarding speed and requires more router memory. The introduction of CIDR [23] has reduced the growth rapidity of the routing table, by enabling more route aggregation in a single prefix. Consequently, the routing table size problem is alleviated at least for short term. Nevertheless, the limited number of peer sessions is still a bottleneck. Our proposed solution consists of splitting the BGP functionalities to be run on multiple controller cards designed using master/slaves paradigm. A master is the card that hosts the RIM, while a slave card hosts FSM, Inbound Policy and Outbound policy modules. A master manages a set of slaves. The slaves apply the inbound policy; forward accepted routes learned from external BGP peers to their master and propagate best routes to adjacent peers. A master (1) calculates best routes learned from the adjacent peers via its slaves; (2) listens to Interior Gateway Protocols (IGPs) such as OSPF or ISIS for any changes within the same Autonomous System (AS); and (3) maintains the consistency of the RIB replicas. We have developed an algorithm to maintain the coherence of the routes among the RIB replicas; the algorithm is based on a relaxed consistency model where we categorize the received updates as *causal* and *concurrent*. Although these notions are similar to those used in causal consistency [17], they are applied in a way that fits the context of inter-domain routing.

Our contribution differs from existing related works in two points: (1) BGP functionalities can be executed in parallel. For example, exchanging the routing table is an extremely demanding task for a centralised BGP module; in our proposal, one slave and one RIM can perform this task while the other slaves and RIMs can handle in parallel different tasks from other peer sessions. We expect our BGP architecture to handle thousands of peers sessions instead of the few hundred as in the case of central BGP (2) RIB replication: According to our proposed solution, only a part of the traffic which crosses the slave input policy will be admitted to the second level verification in the RIB. Besides, only routes that are proven to be best routes in their local group will be multicasted to the rest of the RIMs that then update their replicas. This will enhance the reliability, performance and increase

the space scalability of BGP. The paper is organized as follows. Section 3.2 presents the related work. Section 3.3 reviews the basic concepts of BGP. Section 3.4 introduces the proposed BGP architecture. Section 3.5 proposes a novel algorithm to ensure consistency of RIB replication. Section 3.6 evaluates the proposed architecture via simulations. Finally, Section 3.7 concludes the paper.

3.2 Related Work

During the last decade, several studies have been conducted on router scalability. The concept of a massive clustered router, initially introduced in [12], has become a commercially attractive idea for major router vendors. Recent commercial high-performance routers (cisco [2], juniper [3] and avici [4]) based on distributed multi-chassis solutions are built using this concept. However, the distribution of the routing engines is mostly based on the replication of the control elements as a backup redundancy to support NSR (Non Stop Routing) [10]. Major research efforts have focused on distributed router architectures (e.g., [6], [7], [8]) but none has proposed a distributed architecture of BGP with RIB replication. Hosgand et al., [6] made the design and implementation of a distributed router by decomposing some functionality from control plane to the forwarding plane. The internal communication is performed via a special protocol. Kun et al. [7] proposed a theoretical approach to distribute the BGP protocol in a tree-based manner without assessing the implementation complexity or the communication overhead. Zhang et al. [8] proposed a design of fully distributed BGP architecture based on agent communication. There are also several studies [9], [11], [13] aiming to improve scalability and reliability of BGP. XORP [11] is designed as an open source router with particular emphasis placed on the extensibility. In [9] the authors aim at reducing BGP complexity by proposing the separation of the session maintenance task from the BGP protocol. The work presented in [13] can be considered as the closest to our contribution in this paper. The router is composed of a set of multiple routing BGP engines each responsible for a set of BGP sessions. All received messages are processed and the results are broadcasted to all other processing units. Even though the general schema of the model is presented, the detailed description is missing such as the internal communication protocol being used and

the generated overhead. Furthermore, the consistency of the RIB replicas is not studied and the internal protocol used for communications between the processing units is not defined.

3.3 BGP Protocol

In this Section we present a brief overview of the BGP protocol and the scalability limitations that motivate the distributed BGP Architecture.

3.3.1 BGP overview

The BGP protocol uses TCP as the underlying transport protocol to exchange information with adjacent BGP peers. BGP peers exchange five types of messages to open, establish the connection and propagate routing reachability information. The role of the BGP protocol is to provide the forwarding engine with loop-free inter-domain reachability information. Once a BGP session is established between two peers, the BGP protocol spends most of its time exchanging network reachability information (NLRI) by means of Update messages. These routes learned from the adjacent peers are used to construct the BGP RIB at controller card. A condensed version of the RIB containing only the active (best) routes is downloaded to line cards periodically so that it can be used by the forwarding engine. At startup the BGP peers exchange their full routing tables, once this operation is ended, incremental updates are sent whenever routing table changes (new route is added or existing one is removed) occur. KeepAlive messages are sent periodically to ensure that the connection is alive between BGP peers. Notification messages are transmitted whenever an error is occurred at any side of the connection. Route refresh message is sent on-demand from one peer to its adjacent asking for specific routes already advertised.

3.3.2 Motivation

BGP scalability can be evaluated in terms of the number of peer sessions that can be supported by the controller card and the maximum number of routes that can be handled in the routing table. In major backbone routers, the RIB has been found to change rapidly. As links oscillate in response to error, routing protocol messages are exchanged consequently;

this may cause the routing table to change continuously. Changes may include addition and deletion of prefixes, and modification of next-hop information for existing prefixes. Studies show that these changes in the prefixes often occur at a peak rate of a few hundred prefixes per second and at the average rate of more than a few prefixes per second [22].

In order to achieve scalability in terms of peering sessions, we propose a new distribution of BGP that preserves its functionality as defined in [5]. The proposed distribution increases considerably the number of peer sessions that can be handled and enforces the reliability by replicating the RIB on several controller cards. Since replicas of the RIBs should be kept coherent, we expect a larger number of internal messages to be generated in order to synchronize the RIBs. As it will be shown in Section 3.4, the increase in the number of internal messages does not affect dramatically the performance; this overhead is a small price to pay for the scalability and reliability of BGP operations.

3.4 Distributed BGP Architecture

Instead of having one controller card that hosts among other, the BGP engine, there will be large number of controller cards acting as slaves and masters. The distribution encapsulates the FSM, the input and output policy modules on a controller card namely “slave”; the RIM process and the RIB coexist on a controller card namely “master”. The RIM module runs on the master; it performs the decision process for global best routes selection among the routes learned by its set of slaves, synchronizes its RIB with other masters and resolves next hop reachability. The master is not unique; new masters are added as the load increases; they manage new slaves and synchronize the RIB with other masters. The RIB is maintained by M masters each managing N slaves (M<N). Hence, a total routing capacity of MxN controller cards replaces the single card used in monolithic architecture.

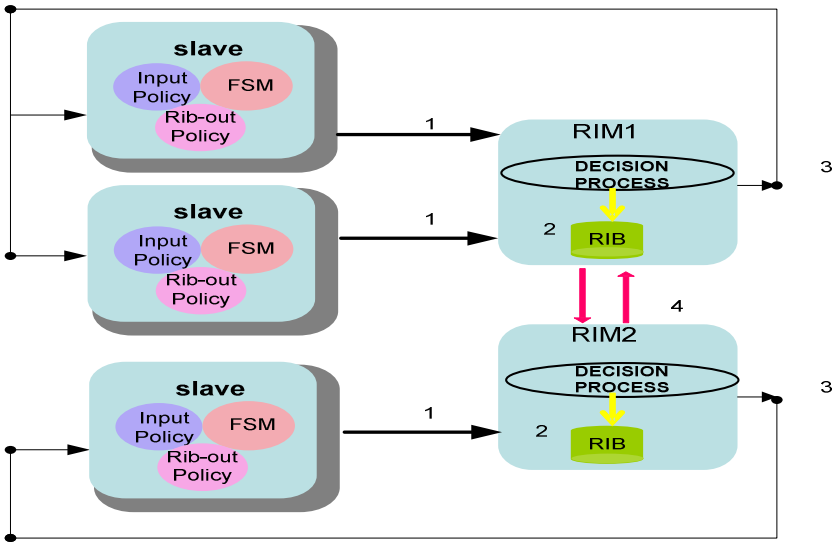


Figure 3.2 Distributed BGP Architecture overview

Fig. 3.2 shows an example of the proposed distribution; it consists of two master cards running the RIMs. The first master manages two slaves while the second master manages only one slave. Upon receipt of incoming routes from its slaves, the RIM: (1) Computes the best routes, (2) then redistributes these routes to its slaves, (3) and finally propagates these routes to the other RIMs. The communication protocol, master-to-masters and master-to-slaves, is a simple reliable multicast that satisfies two conditions: (1) A message sent is received by a destination; and (2) Two messages sent from the same source are received by the destination in the same order as they were transmitted; the connection slave-master is a TCP connection.

In the following, we present the definitions of terms used throughout the paper to describe the various blocs constituting the RIB:

- Adj-Rib-In : the set of all raw routes that have been advertised to the local BGP speaker by its peers.
- Mod-Rib-In : the set of all accepted routes that passed the inbound policy module.
- Loc-Rib: the set of all active routes.
- Adj-Rib-Out : It contains the routes that passed the output policy module and advertised to specific peers by means of the local speaker's UPDATE messages.
- Cache: the set of routes which are second or third best in the decision process.

- Group-Rib-In : A newly introduced set in our proposed contribution; it is the set of all best routes from a group of slaves under a given master leadership. Every master selects the best routes from its local routes in Mod-Rib-In, and only these selected routes will be considered in the global decision process. Locally selected routes by a given master will be set in its corresponding group list. The best from all group lists will be the global best routes. These routes are referred to as "Active" routes and are used by the FIB in the forwarding engine; the routes in Group-Rib-In are referred to as the group best routes.

Although they are mentioned separately, this does not imply that Rib-Ins must be implemented in separate data structures which might increase the memory consumption. A widely used implementation of BGP routing table, Sklower [27], permits one instance with multiple pointers referring to each data set.

We define a field 'Type', included in all messages exchanged internally between RIMs, to define the message type. The RIM-to-RIM message format will be explained in more details in section 3.5.

In the next section, we will describe the interactions, among slaves and masters, following the reception of an advertised (i.e., Update ADD message) or withdrawal (i.e., Update Withdraw message) routes.

3.4.1 Update message

Fig. 3.3 illustrates the components that constitute a master and a slave in the proposed architecture.

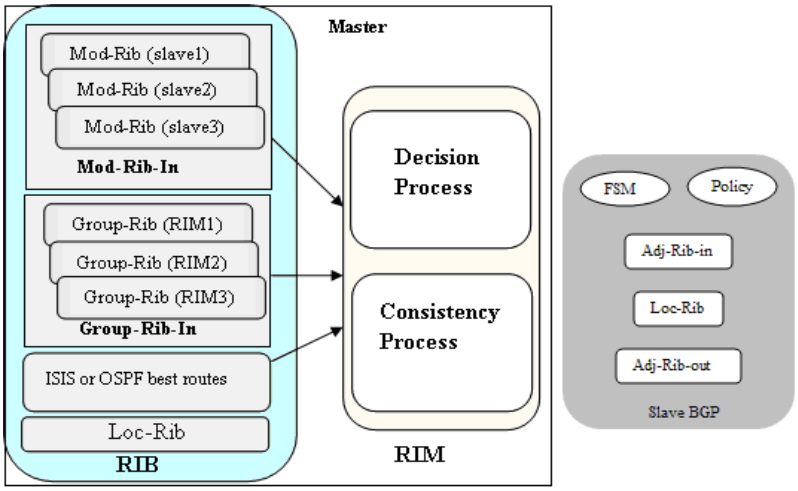


Figure 3.3 BGP distributed engine

Upon receipt of a given announced route *r* from an adjacent peer, the slave makes use of the inbound policy module to process r. If the route passes this process, the slave sends it (modified or as is; this depends mainly on the policy of the router) to its master. The RIM then, checks whether the route is best among these in the local group. If the response is yes, it will be stored in the Group-Rib-In. Then, the RIM proceeds by executing the decision process to compare all routes pointing to the same prefix from all groups. This is done by parsing all routes in the Group-Rib-In, looking for routes with the same prefix as *r* in each group; the comparison of r with all groups' best routes determines whether r is global best or not. If it is not, then it will be multicasted to all RIMs with a message Type set to 'L' meaning that the route is locally accepted in its group. Moreover, if the route is best for all groups then it will multicasted to all RIMs with the Type set to 'A' as Active route; in this case, every RIM updates its RIB image and the active route is redistributed by each master to its slaves.

Upon reception of redistributed routes, a slave aggregates, if suitable, the routes. Then, it updates its Loc-Rib and executes the outbound policy module before advertising the routes to its adjacent peers.

3.4.2 Withdrawal mechanism

Withdrawal mechanism in typical BGP implementation is triggered either because of a session failure or due to reception of an update withdraw message.
In the proposed architecture, when a BGP session goes down, the following steps are executed:

- Initially, the slave detects the TCP connection termination with the external peer; the slave sets all the routes learned from that peers as 'invalid'. Then, sends a control message specifying the deletion event to the master.
- All routes associated with that peer are cleared from the Adj-Rib-In at both slave and master sides.
- The master recalculates its best routes (group best and global best) for the destinations of the invalid routes of the deleted peer.
- The master accordingly removes the routes associated with the dropped peer; then, it redistributes the newly elected routes to its slaves and multicasts a control message to all masters, announcing the withdrawal event.
- Upon reception of the message, the masters perform the same process. The whole mechanism terminates, when all slaves for all masters remove invalid routes from their Loc-Rib and announce replacement routes to their peers.

When receiving a withdraw message from its peer, a BGP slave first applies the route flap dampening process [14]. Then, it sends to its master an internal message with the Type set to 'R', along with the corresponding route to remove. The master applies the Consistency algorithm presented in section 3.5 and executes the required action.

3.5 Consistency problem definition and solution

In this Section, we define the problem of consistency resulting from the replication of the RIB on multiple master cards. Then, we propose a solution that resolves the inconsistency within the proposed architecture.

3.5.1 Consistency definition

In a typical BGP implementation [1], inconsistencies can occur between the RIB and the FIB in the same router; inconsistencies may include: missing prefix, extra prefix and wrong Next-Hop. The problem of RIB inconsistency may also appear in different routers in the Internet. In same router case, commercial routers use a "consistency checker" process to regularly compare the FIB entries with the latest adjacency information from the RIB and subsequently log inconsistencies [2]. In addition, several studies have been carried out to detect and resolve inconsistency in different routing tables in the Internet [15]. Since the proposed architecture is based on RIB replication, it raises the same challenging issue; indeed, the challenge is how to maintain consistency among the different RIBs replicated in the different master controller cards.

For better understanding of the inconsistency issue, let us consider the following example; first we define the following terms to be used throughout the text:

r_i : route announced or withdrawn for a prefix i

r_i^- : withdrawal route for prefix i received from external peer

r_i^+ : announced route for prefix i received from external peer

$r^+_{i,j}$: announced route for prefix i received from j^{th} RIM

$r^-_{i,j}$: withdrawal route for prefix i received from j^{th} RIM

Let us assume that three RIMs with IDs 1, 2 and 3 receive respectively at time t_1 three distinct update messages for routes { r_i^+ , s_i^+ , r_i^-}; r_i and s_i have same Next-Hop and AS (Autonomous System) list, but they may differ in other attributes. Let us consider a naïve implementation where each RIM processes the message upon its reception from a slave and propagates the same message to the remaining RIMs; since there is no central entity to assure message sequencing, routes may be transmitted among RIMs in different orders. Consequently, RIM_1 may remove r_i and upgrade s_i from its cache to be the active route; while in the other RIMs, the route r_i will be treated as active and the second best route s_i will be in the cache as shown in Table 3.1. If, later on, the route s_i is withdrawn, the first RIB will generate a black hole for the prefix i.

TABLE 3.1 RIB'S INCONSISTENCY

Time	RIB 1	RIB 2	RIB 3
T1	r^{+}_{i}	s^{+}_{i}	r^{-}_{i}
T2	$s^{+}_{i,2}$	$r^{-}_{i,3}$	$r^{+}_{i,1}$
T3	$r^{-}_{i,3}$	$r^{+}_{i,1}$	$s^{+}_{i,2}$
Active	r_i (removed),	r_i	r_i
Cache	s_i (upgraded)	s_i	s_i

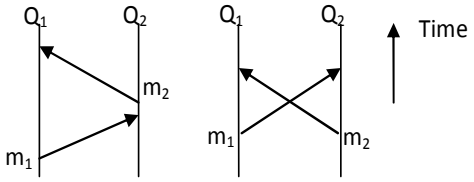
This implementation may lead to some serious problems as incoherent routes will be propagated to different adjacent peers or black-holes will be created in the Internet.

3.5.2 Consistency algorithm

Although the "Consistency checker" approach is used to resolve inconsistency between RIB and FIB in the same router, it is inappropriate for RIB replication on multiple master cards. Indeed, an inconsistency between RIB tables can last several minutes before it can be detected, thus creating significant black-holes in the Internet. In addition, applying this process induces performance degradation as the number of RIBs increases. Unlike the work presented in [15] and the "Consistency checker" in [2], our approach is based on synchronizing the RIBs for every update in order to have the same routes in all RIBs. This consistency model is built upon the logical clocks principle.

Logical clocks have been first introduced by Lamport [16] to define a form of serializability in distributed systems. In many distributed systems, totally ordered multicast protocol is often associated with replication to ensure data coherence. The replicas are kept coherent by executing all operations in the same order; vector clock protocol [19] is an example of such coherent protocols. A number of middleware systems provide a totally ordered and causally ordered reliable multicast at the communication layer [25]. It has been argued whether this implementation of multicast, scales well in large distributed systems [18]. For instance, consider the BGP RIB's replication. A totally ordered multicasting protocol at the communication layer will consider any two messages as related events, even though the

messages might be independent and process a completely different routes. Therefore, an additional cost, in terms of synchronization, will be incurred to order events that do not necessarily need to be ordered. In BGP, the knowledge of the transmitted data among RIBs will make the synchronization restrictive to only these related (relevant events) messages; this will allow relaxing the communication multicast protocol to be simply a communication vehicle to multiple destinations.



(a) m_1 and m_2 are causally related (b) m1 and m2 are concurrent

Figure 3.4 causal and concurrent messages

Our consistency model is designed specifically to fit the BGP implementation by restricting the messages, to be ordered, to only these that relate to the same routes. Unrelated messages can then be executed in any order according to causal consistency [17]. As in the causal consistency model, we define two types of messages: Causally related messages and Concurrent messages (Fig. 3.4). The causality is formed between two routes to the same prefix via the same next hop. To classify the received messages according to their causality, a **Hold-back priority queue** is used by each RIM. Withdrawal messages received from slaves and all messages received internally from other RIMs are parked for a certain time T (T is the bounded time limit to receive an ACK from all destinations). In addition, an ADD message received from the slave will be executed without delay by the receiving RIM, and the message will be set in the queue just for counting as will be shown next. Whenever a message is received by the RIM from a slave and the Hold-back priority queue already contains a message with the same route, then the two messages are considered to be causally related. If two UPDATE messages, for the same route, are issued from two distinct RIMs and the messages are not causally related, they are considered to be concurrent. Since the communication delay is finite and bounded, all concurrent messages will be delivered

within the bounded time interval T. As a rule of thumb the algorithm is based on the following principle:

"Causal update messages with distinct types (ADD, WITHDRAW) must be executed by all RIMs in the same order, while the execution of concurrent UPDATE messages are subject to a heuristic rule convenient to BGP processing". In order to have the same sequencing of messages in every queue, we incorporate the same technique used in [24]; every RIM keeps an up-to-date array of sequence numbers for all RIMs; each entry in the array represents the logical clock of each RIM. The array is maintained as follows:

(1) Initially all the values in the array are set to zero in every RIM.

(2) The local sequence number of every RIM increases monotonically; it is incremented for every message received from a slave, handling best group route or active route.

(3) The entire array of sequence numbers is piggybacked in every message multicasted by a RIM.

(4) Upon receipt of a message, the incoming array of sequence numbers is compared to the local array; each entry of the local array is set to the maximum of the corresponding value in the local array and in the piggybacked array.

Assuming that every message in the queue contains a single route; in our contribution, we define a new metric, called 'Weight', which is associated to each message in the queue to define its execution priority. For a given message 'm' with route r_i received from a slave, its weight is computed as follows:

If the Hold-back priority queue, does not contain a message to a route s_i equals to r_i, then the weight of the message will be set to one:

$$W(m) = 1 \quad /*\textit{weight of } r_i \textit{ equals to 1}*/$$

If messages m' with route s_i equals to r_i existed in the queue, then the weight associated to r_i will be:

$$W(m) = max(\forall W(m')) + 1$$

When a message is received from a RIM, its weight is set according to the comparison of sequence numbers with all messages to the same route in the queue. For instance, consider

three RIMs: P, Q and R maintaining the RIB consistency. Let m and m' be two messages issued from the two RIMs, P and Q respectively with the same route s; and let $V\_SN_m[p,q,r]$ and $V\_SN_{m'}[p,q,r]$ denote their corresponding piggybacked arrays of sequence numbers. The weight of the arrived message m at the RIM R (i.e., same for all destinations) is set as follows:

If the queue at R does not contain a message to route s then:

$$W(m) = 1$$

If the queue at R contains already the message m' then the weight of m is set according to the following three equations:

$$V\_SN_m[q] \geq V\_SN_{m'}[q] \ \wedge V\_SN_m[p] > V\_SN_{m'}[p] \Rightarrow$$
$$W(m) = W(m') + 1 \text{ (i.e., m' preceded m).} \quad (1)$$
$$V\_SN_m[p] \leq V\_SN_{m'}[p] \ \wedge V\_SN_m[q] < V\_SN_{m'}[q] \Rightarrow$$
$$W(m) = W(m') - 1 \text{ (i.e., m preceded m').} \quad (2)$$
$$V\_SN_m[p] > V\_SN_{m'}[p] \ \wedge V\_SN_m[q] < V\_SN_{m'}[q] \Rightarrow$$
$$W(m) = W(m') \text{ (i.e., m and m' are concurrent).} \quad (3)$$

When the holding timer expires for a message in the queue, the RIM triggers a process to execute, the message along with all its concurrent messages in the queue (i.e., messages with the same weight and with the same route); to this end, the process counts the number of the UPDATE messages which have the same weight and applies the following counting rule:

If count (s+) ≤ count (r-) execute (r-); remove all s from Q

If count (s+) > count (r-) remove all r- from Q; execute (all s)

In this way every RIM executes the same messages in the same order without the repeated execution of concurrent messages. For better understanding of the algorithm presented in Table 3.2, we consider the example depicted in Fig. 3.5; it shows a snapshot of the communication held between two RIMs during a given time interval. The first RIM sends two messages a, b while the second RIM sends one message x.

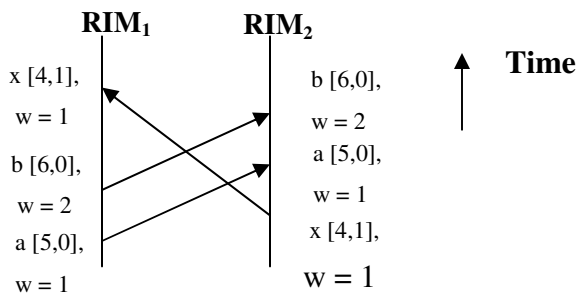


Figure 3.5 illustrating the arrays of sequence numbers

We assume that all messages are about the same route and that message 'x' is a Withdrawal message while 'a' and 'b' are ADD messages; initially, the sequence number in RIM_1, upon receipt of 'a', was equal to 4 and the sequence number in RIM_2, upon receipt of 'x', was equal to 0. In addition, we assume that RIM_1 and RIM_2 did process all the messages exchanged before 'a' and 'x'. Thus, the arrays V_SN that correspond to the clocks of both RIMs before the reception of 'a' and 'x' were equal to [4, 0] at both RIMs; upon receipt of messages 'x' from the corresponding slave, RIM_2 stores the message in the Hold-back priority queue for a waiting period; simultaneously, RIM_1 receives the two ADD messages 'a' and 'b', processes them and parks them in the queue for later comparison with other concurrent messages. If no concurrent messages are received during the waiting period, they will be discarded from the queue. Upon receipt of message 'a' from RIM_1, RIM_2 performs the comparison of sequence numbers of the piggybacked array in 'a' and the local array of sequence number attached to 'x'. Since the value of $V\_SN_a[1]$ is greater than $V\_SN_x[1]$ and the value of $V\_SN_a[2]$ is less than $V\_SN_x[2]$, then as per (3), the two messages are concurrent and assigned the same weight value (i.e., w = 1). After a while, the message 'b' is received at RIM_2; similarly, by applying (1), RIM_2 detects the causality between the messages 'b' and 'a' and sets the weight of 'b' to the greatest weight plus one (i.e., w = 2).

The same process is performed at RIM_1 for the message 'x'. Since the concurrency condition in (3) is satisfied between 'x' and 'a', RIM_1 assigns to 'x' the same weight as 'a'. When the hold waiting time expires for message 'a', RIM_1 parses all the messages with the

same route and same weight as ‘a’ in the queue and applies the counting rule. The number of ADD messages is equal to the number of Withdraw messages (‘a’ and ‘x’); accordingly all ADD messages are removed and the Withdraw message ‘x’ is executed. In the other hand, after the hold waiting time expires for the message ‘x’, RIM_2 extracts all messages with same route and same weight as ‘x’ (i.e., ‘a’ and ‘x’). Using the same counting rule, the route will be withdrawn. Finally, when the timer of ‘b’ expires, ‘b’ is processed and its route is added at both RIMs.

3.5.3 Implementation

Our RIM-To-RIM message exchanged, has the following format:

Sender_ID	Route	Type	V_SN

Where Sender_ID is the ID of the RIM that multicasts the UPDATE message; Route is the route information (the route prefix and the attributes included by the slave); Type is the type of the message (‘R’, ‘A’ or “L’); where ‘R’ means, the message contains a withdrawal route; ‘A’ means, the message contains an active route and ‘L’ means, the message announcing a local best group route as described in section 3.2; and V_SN is the array of the sequence numbers of all RIMs. A possible implementation of the proposed consistency model is detailed in the pseudocode shown in Table 3.2 where two threads: producer and consumer collaborate to implement the approach. The producer thread is responsible for executing the received messages from slaves and RIMs. The messages with ADD route from a slave are executed upon reception and redistributed without delay to other slaves. Withdrawal messages from a slave and any incoming messages from a RIM are parked in the Hold-back priority queue, with the right weight as described in the previous section. The consumer thread executes the messages in the queue according to their weights.

TABLE 3.2 RIB REPLICATION CONSISTENCY PSEUDOCODE

Algorithm: Producer thread

Input : route r_i from slave **Output** : *Queue 'Q' and message multicast*

Case 1: /*message received from slave*/

Begin

1. ($r^-_i \vee r^+_i$) is received from a slave at time t)
2. W (r_i) = 1 /*initially we suppose that no causality exists*/
3. SN = SN+1 /*increment sequence number*/
4. If (r+)
5. Begin
6. Execute (m); /*decision process and redistribution */
7. End if
8. Lock Q
9. If $\exists s_i \in$ Q where r = s then /*r and s are causally related*/
10. Begin
11. W (r_i) = max ($\forall$W (s_i)) + 1
12. End if
13. Insert m (Id, r_i, type, W (r_i), V_SN, t) into Q,
14. Unlock Q
15. multicastToRIMs(m (N, r_i , type, W(r_i), V_SN);

16. End case 1

*17. **Case 2 : /*message received from master*/***

18. Begin

19. $m(r^-_{i,k}) \vee m(r^+_{i,k})$ is received from RIM K;
20. /*Merge the V_SN of the sender with the local V_SN*/
21. For i : = 1 to n /*n is the number of RIMs*/
22. V_SN[i] = max (m.V_SN[i], V_SN[i])
23. Lock Q
24. If $\exists$ $s_i \in$ Q where $s_i = r_i$ then
25. Begin
26. Weight_Max = max ($\forall$ W (s_i)) /*m' is the message of Weight_Max*/
/*j is the sender id of m'*/
27. If $\exists$m', m.V_SN[j] $\geq$ m'.V_SN[j] $\wedge$ m.V_SN[k]> m'.V_SN[k]
28. Begin
29. Insert (m (K, r_i, type, Weight_Max+1, V_SN, t)) into Q
30. End if
31. Else If $\exists$m', m.V_SN [j] < m'.V_SN [j] starting from lowest weight
32. Begin
33. If $\exists$ m', m.V_SN[k] > m'.V_SN[k])
34. Insert (m (K, r_i, type, m'. Weight, V_SN, t)) into Q
35. Else
36. Insert (m (K, r_i, type, m'. Weight-1, V_SN, t)) into Q

37. End if
38. End if
39. Else
40. Insert (m (K, r_i, type, 1, V_SN, t)) into Q
41. Unlock Q

End case 2

END Thread

Algorithm: Consumer thread

Input :waiting time, Queue, **Output** : message execution

---

1. Begin Thread
2. Boolean Exec = FALSE;
3. *Lock Q*
4. Select oldest message m;
5. If (m.route = r_i^-)
6. Begin
7. Find the set S of all routes with prefix i $\in$ *Q* where s = r_i;
8. If $\exists$ s $\in$ S where W(s) < W (r_i^-) remove (s);
9. If $\exists$ s $\in$ S where W(s) = W (r_i^-) then
10. Begin
11. If count (s+) $\leq$ count(s-) remove all s from Q; Exec = True;
12. If count (s+) > count(s-) remove all s- from Q; execute (all s+);
13. End if
14. Else Exec =True;
15. End if /* r_i^- */
16. If (m.route = r_i^+)
17. Begin
18. Find the set S of all route with prefix i $\in$ Q where s = r_i;
19. If $\exists$ s^- $\in$ S, W(s) < W (r_i^+) remove all s-; Exec = True
20. If $\exists$ s^- $\in$ S where W(s) = W (r_i^-) then
21. Begin*
22. If count (s-) < count (s+) remove (s-); Exec = True
23. If count (s-) $\geq$ count (s+) remove all s from Q; execute s-
24. End if
25. Else Exec=True;
26. End if /*r_i^+ */
27. Unlock Q
28. If (Exec)

Execute (message m) /*done by the dispatcher thread*/

END Thread

3.5.4 Implementation analysis

Relevant events ordering mechanism in distributed systems is used to order small subsets from large set of events. Typical implementations relying on vector clock to detect consistency suffer from two drawbacks: (1) All events must be delayed to detect causality, (2) no distinction of relevant events. Aggrawal et al. [28] proposed chain decomposition technique to detect dependency among small percentages of events. The approach showed efficiency in shared memory systems; however, in physically distributed systems the author concludes that vector clocks might perform better for small number of processes. In BGP, the probability of obtaining dependent events (same route messages) at the same time in the Hold-back priority queue is very small. For this reason, delaying all events a given time before processing, is not a scalable solution and would impact the BGP convergence time. Our implementation (table 3.2) does not impose any delay on the execution of update messages received from slaves unless for withdraw messages. Delaying the execution of withdraw messages is an acceptable cost to pay in BGP. Indeed, BGP similarly uses the MRAI (Minimum Route Advertised Interval) timer to postpone route withdrawal for few seconds to detect route flapping. The producer's pseudocode in table 3.2, reveals that when an Update route (r+) is received from a slave, three actions are taken by the RIM: 1) the message is executed (line 6) ; then 2) the message is parked in the Hold-Back queue to detect causality (line 13); and finally 3) the message is multicasted to all RIMs (line 15). On the other hand, when a Withdrawal route is received from a slave or any route update from another master it must be parked in the hold-back queue and its weight which define the causality with other messages will be adjusted accordingly (lines 26-40).

The consumer thread is triggered periodically to consume messages from the Hold-back queue. When triggered, the consumer thread fetches the tail of the queue to get the oldest message. After extraction, the consumer thread searches the queue for related messages to detect causality (line 7 and line 18). We distinguish two cases: causal and concurrent. All causal messages will be executed in order according to their weight. Concurrent messages are subject to counting rules (line 11-12, 22-23). After detecting causality, the message execution is done implicitly in later stage after quitting the critical section (line 27). This process is implemented by a third thread namely "dispatching thread".

3.6 Experimentation and results analysis

In order to evaluate the proposed architecture, we implemented at user level a distributed system consisting of a set of masters running RIM modules and a set of slave modules that generate BGP traffic. The main goal of the evaluation is to study the communication latency, according to the number of slaves and masters, and how it affects the total processing time. We are only interested in the BGP traffic crossing one slave up to the advertisement to external peers. Every group of slaves initially connects to a master via TCP to send routes received from the external peers; it receives results asynchronously from the master via a UDP multicast connection. Each received message is acknowledged by every slave. The RIMs intercommunicate via a simple multicast protocol where each message is acknowledged by the receiving RIM. For the purpose of our experiments, we have chosen a realistic fixed UPDATE BGP message contained in a packet of size 256 bytes. The experimental benchmarking process consists of 16 slaves with for RIMs each RIM is running on a dual core 4GHZ machine running Linux Fedora 10, all interconnected with a 300Mbps router enabled to perform the multicast. Our aim is to extract relevant measurements about three important issues: (1) Evaluate the time taken to completely transfer a BGP table to multiple RIMs during the router's startup phase; (2) Study the BGP performance speedup for simulated daily workload traffic, when multiple RIMs are running in parallel; and (3) Study the Hold-back queue behaviour which is directly correlated with the scalability. The waiting time of every message parked in the Hold-back queue was set to 200ms, which is a sufficient time to receive an ACK from the RIMs. According to the proposed algorithm, all received messages with feasible routes originating from a slave are directly processed and broadcasted by the RIM to its slaves and to other RIMs. However, the Withdrawal routes and all routes originating from RIMs stay in the Hold-back queue and are processed as described in the algorithm presented in section 3.5. The RIM is designed in multithreaded blocks that run asynchronously; a receiver thread handles the incoming traffic; a consumer thread executes the decision process and the consistency algorithm; a dispatcher thread handles the sending of all notification messages to all other RIMs.

3.6.1 BGP routing table transfer performance

In this test scenario, we simulated the routing table transfer. Each slave stores the received messages from its RIM, in a special data structure to emulate the Loc-Rib of BGP. Our aim is to test the speed at which a router can update the Loc-Rib in all slaves and all replicated RIBs in the master cards. To perform this scenario, we dedicated one slave to inject a large number of announcement messages to the RIM. Then, we started the experiment by gradually introducing the slaves and reporting the number of BGP messages received, in terms of packets per second. We conducted the test in three phases: (1) one RIM managing sixteen slaves; (2) Two RIMs, each managing eight slaves; and (3) Three RIMs, where two of them managing five slaves and one managing six slaves.

Fig. 3.6 (phase 1), shows the variation of the number of packets, received by one slave, while increasing the number of slaves managed by the RIM. In particular, Fig. 3.6 shows that one salve is able to receive the whole Loc-Rib with an average speed of 26000 packets/s or 6.5MB per second (26000 x 256bytes). Thus, for a routing table of 100 MB, one slave receives the whole table and updates its Loc-Rib in 15 seconds. It is clear that the rate (packets/s per slave) drops slightly and gradually as the number of slaves increases.

Figure 3.6 One RIM with 16 slaves

The reported rate drop was expected since the number of ACKs (sent by slaves) increases with the number of slaves. Thus, for a BGP table size consisting of 400,000 messages

transfer (100MB), the 16 slaves and the RIM, receive the routing table at a rate of 10,000 packets/s or 2.5MB per second. This means that, the routing table of 100 MB is received and replicated in the 16 slave (Loc-Rib) and the RIM (Loc-Rib) in 40 seconds. Fig. 3.7 shows the performance results when using two RIMs. With an 8-slave group under each RIM, we observe that the 8 slaves of the first RIM receive the BGP table at a rate of 16000 packets/s (x=8 in the X-axis) while the 8 slaves of the second RIM receive the BGP table at a rate of 3800 packets/s. The graph shows the received rate by the two groups of slaves starting at the ninth column; it means that the first group of slaves receives the whole BGP table from their first RIM in a shorter time (i.e., 25 seconds) than the second group. The second group of slaves receives the whole table from RIM_2 in 100 seconds. This overhead is due to the latency induced while RIM_2 is waiting for messages to arrive over the network from RIM_1. The difference in the rates between the two groups of slaves can be attributed to the threaded structure of the RIM. In fact, at RIM_1, the dispatcher thread receives one ACK from RIM_2 compared to eight ACKs received by the consuming thread from slaves for each message. Hence, the kernel allows more CPU slice time to be allocated to the consuming thread and therefore more packets per second are sent to slaves of the first RIM.

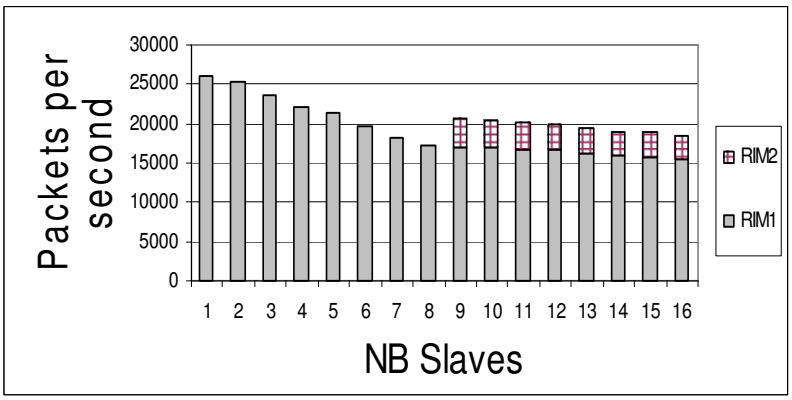


Figure 3.7 Two RIMs with 8 slaves each

Fig. 3.8 shows the performance results when using three RIMs. Each of two RIMs manages five slaves while a third RIM manages six slaves. The first five bars in the graph represent the rates by which the slaves of the first RIM receive the routing table; the

textured bars starting at x=6 represent the rates of the second RIM slaves; the white bars starting at x=11 represent the rates of the third RIM slaves.

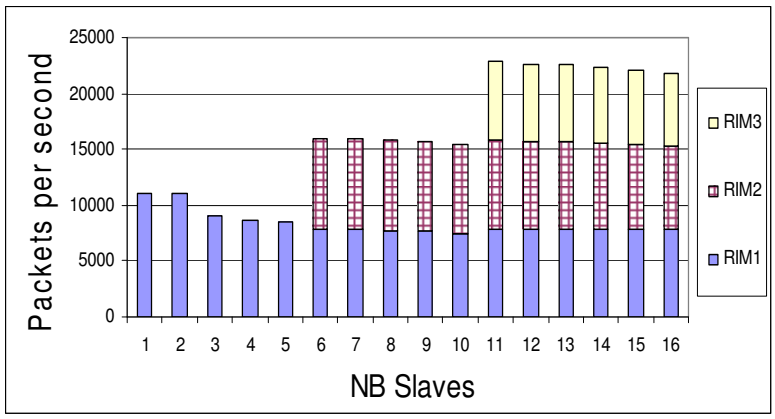


Figure 3.8 Two RIMs with 5 slaves, 3rd with 6 slaves

We have reported approximately 8000 packets/s at each RIM's slave. The fact that interprets the cause of such equivalent distribution of the rates at the different RIMs can be attributed also to the threading structure of the RIM. Since the dispatching thread is receiving more ACKs than in the previous case, all threads are sharing nearly a quasi-equal CPU slice time during the execution; thus resulting an equally distributed rates at the groups of slaves. Consequently, we replicated the routing table to sixteen slave cards and three master cards in 50 seconds. While the transfer made by one slave and one RIM is achieved in 15 seconds.

3.6.2 BGP parallel performance

In the simulations above, we considered the case of one RIM receiving the messages from one slave (generator) and redistributing them to its slaves and other RIMs. In this section, we consider the case where every RIM receives the messages in parallel from a dedicated slave (generator).

In [22] it has been shown that the daily BGP workload is about few hundred messages per second. In order to simulate an equivalent workload, we tested three traffic generators with capacities of 500, 1000 and 1500 BGP update messages/s respectively. All messages are

chosen to announce new routes; we started the generators in parallel on 4 RIMs, each managing a set of 4 slaves. Fig. 3.9 shows the results. For the 500 messages/s generator at each RIM, practically no performance degradation is observed in the 4 RIMs. The slaves received the messages at a rate of 2000 messages/s (speedup = 4). For the 1000 messages/s generator at each RIM, the slaves received the messages at a rate of 3300 messages/s (speedup = 3.3). With 1500 messages/s, the slaves receive the messages at a rate of 4800 messages/s (speedup = 3.2). In [26] it is shown that a commercial router can handle up to 3332 messages/s with no RIB modification. By comparing this result to those depicted in Fig. 3.9, we conclude that our architecture outperforms the tested commercial router, while preserving the coherence of the replicated RIB on different distributed controller cards.

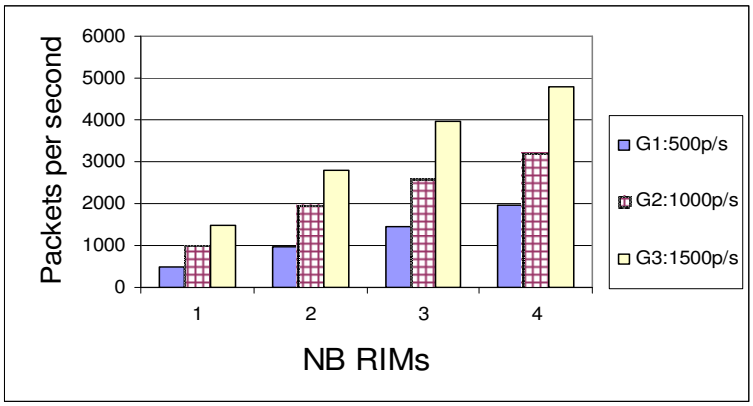


Figure 3.9 Four RIMs with 4 slaves each (3 generator test)

3.6.3 Queue behaviour

During the tests, we observed that the maximum peek size of the Hold-back queue was maintained at 3000 messages when four RIMs involved in the computation. The theoretical size of the queue when no processing operation applied on its elements is given by: r x T; r is the uniform rate of messages/s and T is the waiting time of the message in the queue.

In our test case, the theoretical size must be equal to 1000 messages (r=5000 and T=200ms). By comparing the theoretical and empirical values (1000 vs. 3000), we conclude that the overhead added by the consistency algorithm is acceptable, since the

number of messages contained in the queue can be consumed in few seconds by the consuming thread.

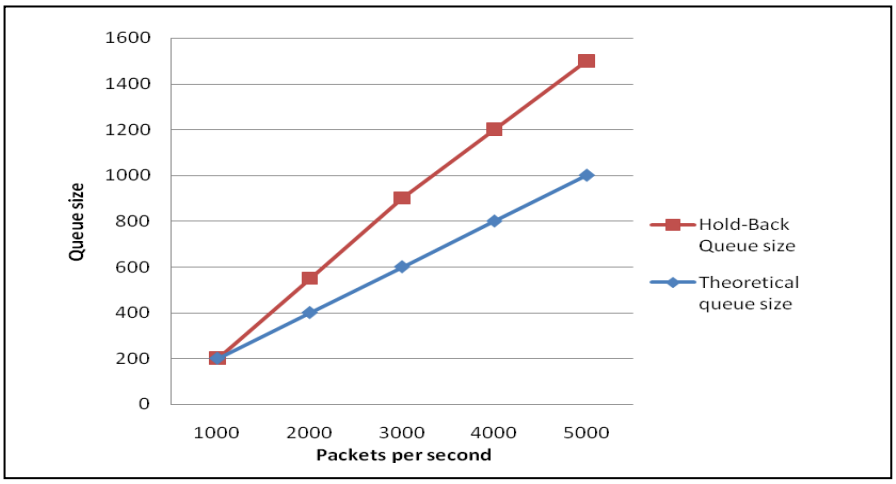


Figure 3.10 Hold-Back priority Queue growth

In Fig. 3.10, further tests are done to evaluate the growth size of the Hold-back queue and compare it to the theoretical size while increasing the incoming traffic from slaves on two RIMs.

3.7 Conclusion

In this paper, we have introduced a new architecture for distributing the BGP protocol applicable to next generation routers. The distribution is based on a master-slaves paradigm, and the replication of the RIB on several master cards. Scalability and reliability are the key goals of the architecture. In addition, we proposed an algorithm to maintain the consistency of the RIBs replicated on multiple master controller cards, at an acceptable cost. This cost is offset by the benefit of the replication achieved. In this contribution, a simulated BGP table of 100 MB is transferred and replicated in three RIMs and 16 slaves in an average time of less than 50 seconds. In addition, a better performance results compared to commercial router, were obtained when testing daily workload traffic equally distributed on 4 RIMs. We believe that these results prove the effectiveness and applicability of our approach in next generation routers.

3.8 References

[1] http://www.zebra.com.

[2] *"Next generation networks and the cisco carrier routing system".* 2004. http://www.newsroom.cisco.com/dlls/2004/networks_and_the_cisco_carrier_routing_system_overview.pdf.

[3] *"Building a Higly-Available Enterprise Networks with Juniper Networks"*, 2010. http://www.juniper.net/us/en/local/pdf/whitepapers/2000257-en.pdf.

[4] http://www.ipinfusion.com, "a redundant architecture for routing protocols, IP infusion white paper", 2002.

[5] Rekhter, Y., "A Border Gateway Protocol 4 (BGP-4) ", RFC 4271, 2006.

[6] Hagsand, O., M., Hidell, P., Sjodin. "Design and Implmentation of a Distributed Router." *IEEE Int'l Symposium on Signal Processing.* Athens, Greece, 2005. 227-232.

[7] Kun, W., W., Jianping, X., Ke. "A Tree-Based Distributed Model for BGP Route Processing." *High Performance Computing and Communications.* 2006. 119-128.

[8] Zhang, X, P., Zhu, X., Lu. "Fully-distributed and highly-parallelized implementation model of BGP4 based on clustered routers." *IEEE ICN.* Reunion, France, 2005. 433-441.

[9] Cavalli, A., D., Vierra. "Working around BGP: An improvment of BGP session maintenance." *IEEE International Conference on Networking and Services.* Silicon valley, California, USA, 2006. 41-49.

[10] Kaplan, H. *"Part 3, in the Reliability Series, NSR, Non-Stop Routing Technology".* http://63.111.106.66/technology/whitepapers/reliability_series/NSRTechnology.pdf; Avici, 2002.

[11] VYATTA community: http://www.vyatta.org; (XORP).

[12] Halabi, S. *"Pluris massively parallel routing".* White papers, http://www.academ.com/nanog/feb1998/parallel/index.html; Pluris Inc., 1998.

[13] See-Woong, M., C., Bung-gu, P., Yong-Seok. "Apparatus for distributive processing BGP". Patent 7330474. 2008.

[14] Villamizar,C., Chandra, R., Govindan, R."BGP Route Flap Damping", RFC 2439, 1998.

[15] Wang, L., D., Massey, K., Patel. "FRTR: A scalable mechanism for global routing table consistency." *IEEE International Conference On Dependable Systems and Networks.* Washington DC, USA, 2004. 465-474.

[16] Lamport, L. "How to make a multiprocessor computer that correctly executes multiprocess programs." *IEEE Transactions on Computers* vol.28 (1979) : 690-691.

[17] Ahmad M., J..E., Burns, P. Hutto, G., Neiger. "Causal Memory: Definitions, Implementation, and Programming ." *Technical Report GIT-ICS-93/95, Georgia Institute of Technology, Georgia, Atlanta*..

[18] Saltzer, J. "End to End arguments in system design." *ACM Transactions*, 1984 : 277-288.

[19] Birman, K., A., Shiper, P., Stephenson. "Lightweight Causal and Atomic Group Multicast." *ACM Transactions on Computer Systems* vol.9 (1991) : 272-314.

[20] Yang, L., R., Dantu, T., Anderson, R. Gopal."Forwarding and Control Element Separation", RFC 3746, 2004.

[21] BGP Routing Table Analysis; http://thyme.apnic.net/current/.

[22] Labovitz, C., R., Malan, F., Jahanian. "Internet routing stability." *IEEE/ACM Transactions on Networking* vol.6, no. 5 (1998) : 515-528.

[23] Fuller, V., et al., "Classless inter-domain routing (CIDR): an address assignment and aggregation strategy", RFC1519, 1993.

[24] Fidge, C. "Timestamps in message-passing systems that preserves partial ordering." *Australian computer science communications* vol.10 (1988) : 56-66.

[25] ISIS project: http://www.cs.cornell.edu/Info/Projects/ISIS/ISISpapers.html.

[26] Qiang, W., Y., Liao, T.,Wolf, L.,Gao. "Benchmarking BGP routers." *IEEE International Symposium on Workload Characterization.* Boston, MA, USA : IISWC, 2007. 77-85.

[27] Sklower, K. "A tree-based routing table for Berkeley Unix." *USENIX Conference* . Berkeley, CA, USA, 1991. 93-99.

[28] Agrawal A., Garg K.V. "Efficient dependency tracking for relevant events in concurrent systems." *Journal of Distributed Computing* vol.19 (2007) : 163-183.

CHAPITRE IV

4 A Highly Reliable Cluster-Based Distributed BGP Architecture with Graceful Restart Capability

Wissam Hamzeh, Abdelhakim Hafid

Abstract—The implementation reliability of the interdomain protocol BGP in core routers still represents a major challenge facing the stability of the whole Internet infrastructure. In monolithic architecture, if the BGP instance inside the router fails, all peers' connections will shutdown and the new reachability state will be propagated across the Internet in non trivial convergence delay. During this convergence period, traffic can still be forwarded to some unreachable destinations causing the creation of black-holes in the Internet. Most carrier core routers rely on classical primary-backup scheme to increase resiliency; although this scheme provides high degree of availability, it limits the system scalability. In this paper, we propose a fault-tolerant mechanism for the BGP distributed architecture presented in [2] and a mathematical model to evaluate the reliability of the resulting architecture. The mathematical model shows that the resulting fault-tolerant architecture achieves a significant gain in system availability evaluated in terms of MTTF (Mean Time To Failure) compared to monolithic and client-server BGP architectures while it considerably increases the scalability of BGP.

[Status]: This paper is submitted to Journal of Computer Communications.

4.1 Introduction

In the past several years, BGP succeeded in maintaining a good level of reachability in the Internet. Nevertheless, this successful trend may no longer last in the future without making substantial enhancements to cope with the exponential traffic growth in the Internet. Presently, there is a serious concern in the scientific community about many aspects of BGP that impact the Internet infrastructure, such as the unprecedented inflation of routing tables, the increase of ASes (Autonomous Systems), the traffic growth and the continuous routes instability. These factors highlight the need to improve the BGP architecture inside routers by adopting a distributed architecture without modifying the core of BGP. Recent contributions [3, 4, 5, 6] have shown that many aspects related to the dynamics of BGP, such as iBGP anomalies, route flapping and convergence delays are not clearly understood. Route flapping represents a major problem that impacts the stability of the Internet; it arises from the oscillations of BGP. Update messages cause heavy computations in the routers and increase the convergence time. This convergence delay increases the likelihood to produce black-holes in the Internet where packets are forwarded to inexistent destinations. A study conducted in [4] has revealed that 32.7 % of the errors in the Internet are essentially related to router reliability issues; the rest of errors are due to software configuration, human errors, bugs in routing protocol implementations, malicious attacks and bad update message formats. Moreover, Wang et al. [9] show that a single inter-domain link failure can generate hundreds of withdrawal update messages which may last few seconds. A solution to minimize route oscillations and black-holes is to increase the router reliability so that internal failures, such as controller card and session failures can be masked and fixed locally in transparent manner to the adjacent peers. The progress achieved to increase router reliability is the separation of controller and forwarding engines on controller and line cards presented in ForCES RFC [12]. This solution enables the router to continue the forwarding process (Non-Stop Forwarding) even in the presence of a controller card failure. However, if no backup controller card exists to replace the failed one to update the Forward Information Base (FIB), Line Cards (LCs) will continue to forward packets to some obsolete destinations; this makes the router to be a black-hole. The BGP graceful restart mechanism (BGP-GR [11]) which is based on peer collaboration has

been adopted by the IETF community as a helpful solution to minimize the failure impact of the BGP sessions; however, it is still far from reaching wide acceptance from Internet Service Providers (ISPs). Indeed, it has been reported [10] that BGP-GR capability is often disabled by ISPs; they prefer an immediate refresh of the reachability information, despite its cost, instead of taking the risk of using paths in the FIB learned from the failed peer router that might become inaccessible during its restart. Generally, to protect a system against data losses and failures, redundancy must be added to the original processing units and data storage by using some replication model. Different replication techniques have been developed in the literature; they are mainly based on passive and active replication. Passive replication model is based on the primary-backup technique, where multiple nodes (Controller Cards) work together to mask a failure; when the primary node fails the backup node is automatically activated to resume computation from the last point where the primary node was interrupted. For its simplicity, this failover solution is widely deployed in current core routers; however, it is not scalable. Indeed, to increase reliability in a router by adopting the passive replication model, controller cards must be doubled for the same processing capacity; this degrades the router scalability. In opposition to passive replication model, the active replication is a form of redundancy where replicas perform different parallel tasks on the same replicated data. When using active replication, mechanisms must be deployed to ensure that consistency among the distributed replicas is correctly maintained. Multihoming [14] is a common form of redundancy frequently used by ISPs; it requires a redundant link for each connected peer, causing a lot of resources to be allocated and contributing to a considerable growth in routing tables. In this paper, we propose a new fault-tolerant mechanism to be integrated with the distributed BGP architecture presented in [2] in order to produce a highly reliable distributed BGP architecture. The key contributions in this paper can be summarized as follows: (1) BGP-GR is used as a potential solution to increase reliability in the BGP distributed architecture at the slave side; (2) the removal of the BGP single point of failure; and (3) an analytical model that evaluates the reliability of the resultant fault-tolerant distributed BGP architecture.

The remainder of the paper is organized as follows. Section 4.2 presents the related work. Section 4.3 reviews the distributed BGP architecture in [2]. Section 4.4 introduces a high level description of the fault-tolerant mechanism. In Section 4.5, an algorithmic approach

of the fault-tolerant mechanism is presented. Section 4.6 evaluates analytically the reliability of the proposed architecture. Section 4.7 concludes the paper.

4.2 Related work

Rerouting is the basic recovery mechanism in BGP to mask a session failure. Generally, a failure of a peer router causes a modification in the adjacent peer's routing table and a propagation of a new replacement route in the Internet. The advantage of rerouting is its simplicity and its ability to discover an alternative forwarding path as soon as the session fault is detected. However, its drawback can be seen by the delay to make the propagation of the new path in the entire Internet network. In order to minimize the negative side effect of rerouting, most existing BGP resiliency contributions either (1) use the graceful restart mechanism introduced and adopted by the IETF community [11] which is implemented in most today's carrier routers ; or (2) modify BGP semantic to add reliability [6,7,8]; the basic idea is to create redundant tunnels between the pool of routers inside same autonomous systems; these tunnels can be used to forward traffic to other peers in case of a router failure; these promising solutions [6, 7, 8] aim at improving BGP resiliency; however, they involve changes to the global routing architecture and protocols; they may also cause additional growth in BGP routing tables which is not desirable. In this paper, we propose a new fault tolerant mechanism based on redundancy without changing the semantic or the external visible behaviour of the BGP protocol.

4.3 Architecture overview

BGP scalability can be measured in terms of the number of connected peers that can be handled and the size of the routing table. The reliability of a router is measured by its capacity to survive from failures. The BGP distributed architecture proposed in [2] is built to satisfy both requirements: Scalability and Reliability.

4.3.1 BGP module separation

The proposed distributed architecture implements a split-replicate model to increase both scalability and reliability of BGP. The BGP daemon is split into two sets of modules where each set resides in a separate controller card. Thus, multiple BGP daemons will be executed on multiple controller cards namely masters and slaves. A BGP daemon can be executed on one master and several slaves. A slave controller card (1) manages the FSM (Finite State Machine) with external peers; (2) manages/applies the inbound and outbound policies; and (3) maintains the set of raw and advertised routes from/to external peers and forwards accepted routes to its master to perform best path decision process.

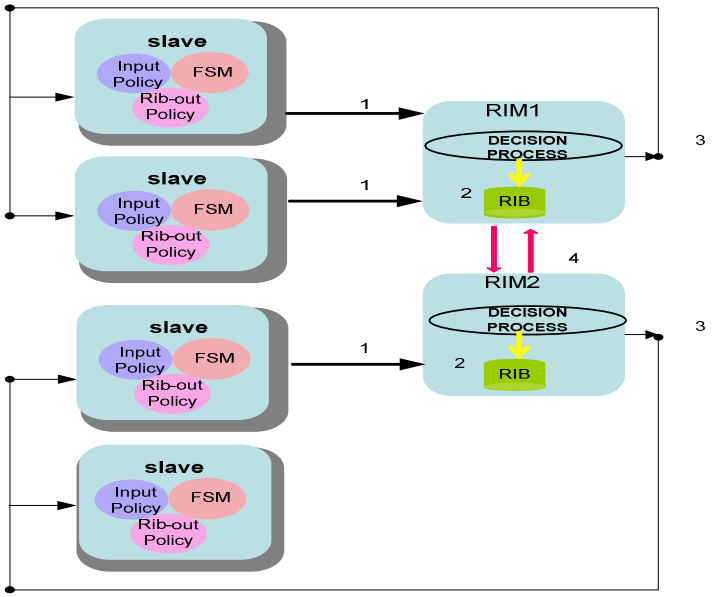


Figure 4.1, Distributed Architecture Overview

A master controller card hosts the RIM (Routing Information Manager) process which stores the set of all routes and the selected best routes from other masters. As shown in Fig. 4.1, the first master card manages two slaves while the second master manages only one slave. A master may receive routes from its slaves and from other masters. Upon receipt of incoming routes from its slaves, the master computes the best routes by applying the decision process and redistributes these routes to its slaves; then, it propagates these routes to other masters. Now, upon receipt of best routes from other masters, the recipient master applies the best path selection process on incoming routes. In case a route is confirmed as

best route, the master propagates it to its slaves. The slaves in turn apply the outbound policies and advertise the route to external peers. If a route received from other masters is found to be group best (i.e., a best route within the master's group that sent it), then it will be kept by the receiving master without propagation to slaves. The group best route will be helpful to the RIM in order to select an alternate best route, in case the later is withdrawn, without communicating with other RIMs.

The internal communication protocol, master-to-masters, and master-to-slaves, is a simple reliable multicast. Throughout the paper, the words RIM and master will be used interchangeably to denote the process running at the master controller card.

4.3.2 Replication model

According to the conceptual definition in [13] a BGP RIB is composed of three data sets: Adj-Rib_In, Adj-Rib-Out and Loc-Rib all stored in the same controller card. In [2] these sets are split and replicated on multiple master and slave controller cards; a new set namely Group-Rib-In is added in the master controller card to enhance the selection of new route when unreachable route is removed from the RIB. These sets are defined as follows: (1) Adj-Rib-In: the set of all raw routes that have been received by the slave from the peers; (2) Mod-Rib-In : this set at the master card contains all accepted routes that passed the inbound filtering policy by the slave; (3) Group-Rib-In : a newly introduced set of routes in the distributed architecture; it is the set of all best routes in the group of slaves; for instance if the master receives two routes to the same destination from two different slaves, only the best route of the two will be stored in the Group-Rib-In. A master holds the group of best routes of all masters; the best route from all groups will be considered as global best (active) route that will be used by the FIB; (4) Loc-Rib: the set of all active routes replicated at each slave and downloaded periodically to LCs; and (5) Adj-Rib-Out : this set at the slave controller card, contains the routes that passed the output filtering policy and advertised to specific peers using UPDATE messages.

4.4 BGP fault tolerant mechanism

In this paper, we consider three types failures: (1) link, connecting two border routers failure; (2) slave failure; and (3) master failure Fig. 4.2.

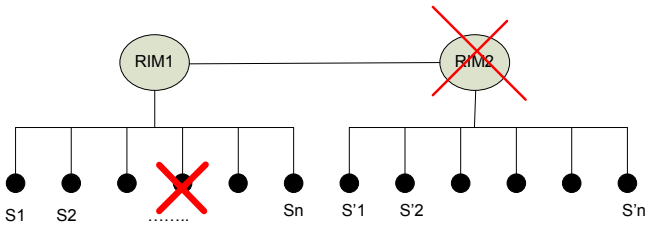


Figure 4.2 Faulty components in BGP distributed architecture

In practice, it is rare that two controller cards fail at the same time; thus, we limit our study to single crash failure. In addition, we do not handle the case of a planned restart nor the partitioning errors that break the communication among RIMs and Slaves.

4.4.1 BGP Graceful restart (BGP-GR)

In 2007 the IETF community has adopted the BGP Graceful Restart [11] as a valid solution. The basic idea of the BGP-GR consists of the collaboration among peers to help the failed/restarted router to re-establish its normal state with a minimal loss. This mechanism can be explained briefly by the sequence of actions taken by the router and its neighbours when a session failure is detected. If neighbours detect the lost BGP session, they do not immediately withdraw the routes originally advertised by the failed router, as in normal re-routing; instead, they mark these routes as “stale”; if an OPEN message (new connection establishment) is received during a short period (5-10 seconds) then the failed router is considered by its peers to be partially failed at controller plane. Therefore, the peer will continue to use the stale routes as valid destinations for some graceful period (~120 s). After that, when the failed router recovers completely, it receives the missed routes followed by an “End-of-RIB” marker from all of its neighbours. Only then, it recomputes its own routes, sends these new routes to its neighbours and removes any remaining stale routes from its BGP routing table. The benefit of using the graceful restart mechanism in BGP is that it gives the failed router (downed in the control plane only), a time to restart

without immediately triggering routes withdrawal. To this end, BGP peers must be equipped with the capability to negotiate the Hold-Up timer that gives the downed router sufficient time to restart without withdrawing its advertised routes. However, the major disadvantage is the increase in convergence time due to the delay in forwarding the new state. Even with primary-backup redundancy deployed in most core routers, using BGP-GR will cause a considerable delay in forwarding the new state. As an example, let us consider a realistic router having 50 connections with adjacent peers and adopt a primary-backup fault-tolerant mechanism at the control plane. If the controller card fails, the backup will need in average 250 seconds (5 x 50 seconds) to recover all lost connections before it starts receiving BGP Updates. Thus, with single backup redundancy, the failed router will still need more time (> 250 seconds) to re-establish its original state. During this period, all peers will be using the same old reachability information advertised by the failed router. This period of time in core routers is considerably high. During this graceful period, black holes and routing loop may easily appear in the Internet. All these factors have led to practical concerns about the use of BGP-GR. It has been reported [10] that BGP-GR capability is rarely turned on [10] by Internet service providers. In our proposed architecture, we integrate the BGP-GR as a part of a fully redundant architecture.

4.4.2 Message logging

Message logging is a technique largely used in distributed systems to mask lost messages whenever a failure disrupts normal message transmission or execution. Message logging is widely used to increase reliability in NACK-based reliable transmission protocols and to increase system fault-tolerance in general. In the following sections, we describe the methodology and implementation to incorporate message logging technique in the context of the proposed distributed BGP architecture to increase its fault tolerance.

4.4.3 Methodology

In order to incorporate a fault-tolerant mechanism to the BGP distributed architecture in [2] that recovers the system from a fault of any of its components, the following challenging issues must be addressed: (1) lost messages between slaves and masters when

anyone of them fails; (2) reconstruction of data that was available in the faulty process (slave/master) before its crash; and (3) consistency of the redundant data carried by BGP Updates during the crash

The proposed solution is to include, in addition to redundancy, the *message logging* technique. The basic idea of message logging is that, lost messages (messages received but not relayed to other slaves or RIMS) can be retrieved by the sender and then retransmitted and replayed correctly by the receivers. This guarantees that the same sequence of re-executed messages that should have been executed before the crash yields the same results after recovery.

4.4.4 Implementation

The message logging technique can be implemented via the use of a Hold-back table that logs messages for a certain time T. The failure detection time is bounded (lack of three heartbeat messages); thus a failure can be detected in time t ($t < T < 2$ minutes). When a failure occurs, a message parked in the queue for a period greater than T will be considered as already executed; otherwise, it must be re-executed by the backup. Consequently, an asynchronous purge process runs periodically to remove messages with time $t > T$ and keeps the others for an eventual reprocessing in case of failure. In most cases, the sender (RIM or slave) does not check whether a re-executed message has already been processed before the crash; thus, when replaying logged messages we should expect duplicate execution of some messages that have already been executed before the crash. In BGP, the re-execution of same update messages will not generate erroneous results. Thus, the execution correctness after the failure is guaranteed.

4.5 Fault tolerant design and algorithm

Generally, the main steps taken to mask a fault in a system are: Detection and Recovery. In the following sections, we propose a detection and recovery mechanism for the three major failures namely link failure, slave failure and master failure.

4.5.1 Communication link failure

A BGP session formed between a pair of peer routers is built on top of a TCP reliable transport connection. A detection of a link failure may signal either a failure of the remote peer or simply a crash of the underlying TCP connection. In both cases, the slave node that manages the failed link collaborates in conjunction with the external peer at the remote end of the connection to handle the error correctly. In case of a failed TCP link, this mechanism is interpreted in our architecture by the following sequence of actions: (1) Failure detection: The slave detects the failed link in an early stage (lack of 3 consecutive KeepAlive messages) from the peer router; and (2) Recovery: Upon failure detection, the slave starts reconnecting as described in [11]. When the peer detects a reconnect OPEN message, it drops the previously failed connection and accepts the new connection without modifying its RIB.

4.5.2 Slave failure: Detection and recovery

In current BGP implementation [1], when a connection is dropped, the live router has no idea whether the cause of the error is attributed to a link failure, a partial or a complete crash of the remote router at the other end of the connection. The BGP-GR idea is that, live peer router grants a surplus of time (5seconds) to the downed router to sends its OPEN message as a sign of aliveness and longer time (120 seconds) to using the stale routes originally advertised by downed router. This mechanism suffers from long delay in core routers (see Section 4.5.1).

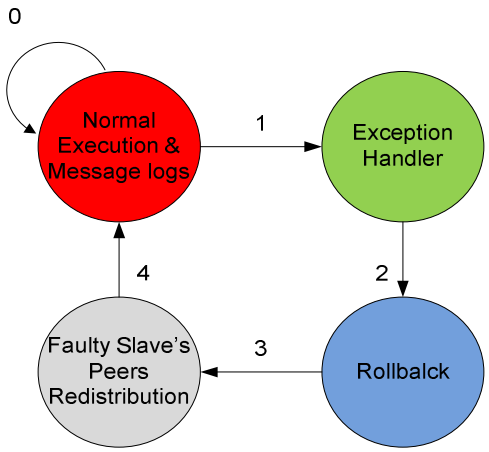


Figure 4.3 State machine of the RIM to mask a slave failure

In our architecture, a BGP-GR is applied by the slave. When a slave controller card fails, the RIM that manages this slave is equipped with a mechanism to redistribute the peers managed by the downed slave process to other live slaves in a transparent manner to external peers. This procedure is composed of two steps: (1) failure detection: the failed slave is detected by the RIM via its detection mechanism such as heartbeat mechanism that sends periodically KeepAlive messages. (2) Recovery: following the failure detection, the RIM fetches all routes that have been best to announce or to withdraw in its message log table; then it redistributes the peers' address list of the disconnected connections along with the missed announcements in a round robin fashion to the remaining live slaves (see Fig. 4.3).

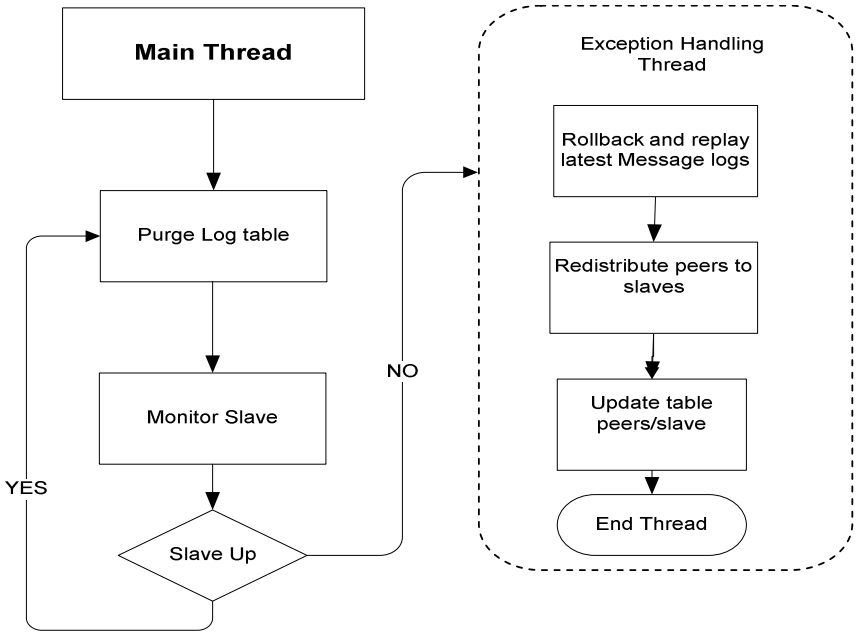


Figure 4.4 Slave resiliency algorithm executed by the RIM

Once the exception thread terminates all peers addresses will be distributed to other existing live slaves. In parallel, the slaves, that received the new addresses, apply individually the BGP-GR mechanism to reconnect with their newly distributed adjacent peers (see Fig. 4.4). The advantage of our approach over classical BGP primary–backup architecture is that, the delay induced by using BGP-GR will be reduced considerably. To appreciate this delay difference, let us consider the same example presented in section 4.5.1 and apply it to our proposed architecture composed of several RIMs and up to 16 slaves per RIM. In the classical architecture, the backup controller card needs to apply the BGP-GR mechanism to 50 connected adjacent peers; this would require from BGP a delay to be around 250 seconds to restore its normal state and resume its functional state. In the distributed architecture [2], the 50 connections will be distributed to 16 slaves that apply in parallel the BGP-GR mechanism. In this way, the recovery delay is reduced to as low as 15 seconds (i.e., the overall delay, 250 sec, divided by the number of slaves that perform the BGP-GR process collectively). Moreover, unlike monolithic architecture, in the distributed

architecture when slaves are busy applying the BGP-GR mechanism with their external peers; the remaining slaves belonging to other RIMs are not affected.

4.5.3 Master failure: Detection and recovery

Master controller card resiliency is more complex than slaves' resiliency because of the central role a RIM plays within the distributed architecture. Recall that, a RIM receives data from its slaves, applies the best route selection process and relay the selected routes to other RIMs. Moreover, the RIM receives data from other RIMs and relay it to its own set of slaves. This central role requires a tight collaboration between RIMs and slaves to detect and recover a RIM from crash failure.

Master failure detection

The failure detection mechanism is applied at both sides (slaves and RIMs) simultaneously to detect a RIM failure. Common failure detectors, such as the heartbeat technique, can be used. To this end, RIMs are classified in a logical arrangement as primaries and backups. As shown in Fig. 4.5 (a), we assume that RIMs are connected in a circular ring. Every RIM will be a primary to its set of slaves and a backup to its neighbouring RIM in the ring according to clock-wise direction. Every set of slaves under a master leadership knows a priori about the designated backup. Similarly, every backup RIM knows *a priori* the address of the primary RIM that it should replace during the failover process. As soon as the two neighbours of the failed RIM detect, from the heartbeat detector, a flag signalling the failure of a RIM, they propagate this new information immediately to their corresponding neighbours (Fig. 4.5 (b)). Recursively, all RIMs detect the RIM failure in a short delay and adjust their neighbourhood heartbeat (see Fig. 4.5 (c)) accordingly. This mechanism can be repeated as long as there is *at least* one RIM alive.

Master recovery

As soon as the failure detection message signalling a RIM crash is received, the receiving backup RIM stops the execution of messages in its Hold-back queue until it receives an End-Of-Log signalling the end of the recovery process.

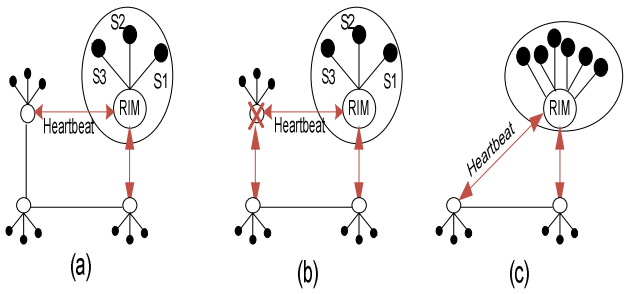


Figure 4.5 RIM fault detection and recovery mechanism

Meanwhile, when the designated backup RIM detects the failure of a primary RIM, it executes the following actions: (1) sends a failure notification to its neighbouring RIM in the ring announcing the failure of the primary RIM (see Fig. 4.5 (c)); (2) accepts the connections from orphan slaves of the failed RIM; and (3) starts receiving, executing and multicasting the log messages from the slaves. After receiving all logged messages and the Adj-Rib-Ins, the backup RIM sends an End-Of-Log Marker message to all RIMs signalling the end of the recovery process. As soon as the End-Of-Log is received, all RIMs start processing the messages from their local queues as in normal execution and adjust their new primary/backup relationships.

4.5.4 Consistency

In order to maintain a consistent view when executing the log messages following a RIM failure, special attention must be paid to make all RIM replicas execute messages in the same order they were received. Recall that, in normal BGP execution [2], the consistency implementation is based on the causality relationship among the received messages. This consistency mechanism is implemented by using Hold-Back priority queue to park the messages for a time period T in order to detect causality among messages. The proposed mechanism in [2] is proved to maintain consistency of the RIB replicas as long as the execution is fault-free. Unfortunately, this mechanism may not lead to the desired consistency level in the presence of faults at any of the existing RIMs. For better understanding, let us consider the following scenario where three RIM processes are involved in managing the BGP RIB replicas and one of them crashes at a given time t. Let

us assume that, during the crash, the Hold-back queue contains the set of routes shown in Table 4.1.

TABLE 4.1 RIB'S REPLICAS DURING A FAILURE

Time	RIM 1	RIM 2	RIM 3
T1	r^{+}_{i}	r^{-}_{i}	r^{-}_{i}
T2	CRASH	$r^{-}_{i,3}$	$r^{+}_{i,1}$
T3	CRASH		$r^{-}_{i,2}$
Active	CRASH	None	None

Where:

r_i^- : withdrawal route for prefix i received from external peer

r_i^+ : announced route for prefix i received from external peer

$r+_{i,j}$: announced route for prefix i received from j^{th} RIM

$r^-_{i,j}$: withdrawal route for prefix i received from j^{th} RIM

According to the consistency mechanism, when the Holdup timer of the message expires, a consumer thread will execute all causally related routes by applying a counting rule [2]. Now, since the multicast in the proposed architecture, for efficiency purpose is not atomic, the multicast message carrying route r^{+}_{i} just before the crash of RIM1, might be partially delivered to some RIMs but not to all of them. Thus, in Table I, RIM3 has received route r^{+}_{i} but not RIM2. At execution, RIM2 and RIM3 remove route r^{+}_{i} because there is a withdrawal route to the same prefix, at the same time, in the Hold-back queue.

After recovery, the salve that has sent initiated the route to RIM1 and kept it in its message logs, sends it again to the backup for replay in the recovery phase. This will generate an incorrect result since route r_i will be added to the RIB while it should be removed because it is unreachable.

The execution ambiguity and similar erroneous results that might show up during a crash are caused by the (1) Existence of withdrawal messages during the failure crash and (2) Lack of atomicity in the multicast protocol.

To deal with any execution ambiguity that might occur after a crash, the RIM keeps a history of all executed withdrawal messages in a separate log table for a given time period

t; when incoming log messages from slaves are replayed, each RIM checks whether the update message carries some routes to some withdrawn prefixes in its executed withdrawal messages. If the response is yes, the route will be removed from the RIB and the corresponding update is discarded. Moreover, when a log message from a slave is a withdrawal message, the route must be deleted and all log updates referring to this withdrawn route must be discarded. Although, by applying this scheme, we might remove, in very rare cases, some reachable routes, we guarantee a correct and consistent view in all replicated RIBs. Additionally, removed routes can be easily replaced during the normal operation of BGP.

4.6 Reliability and analytical evaluation

To evaluate the reliability and the availability of the resultant distributed architecture, we derive a mathematical model that unambiguously expresses the reliability of the whole system in terms of redundancy.

4.6.1 Reliability model

A(t) is the average fraction of time the system is up during the time interval [0,t]. In this paper, to measure the reliability of the architecture in [2] with the fault tolerant mechanism, we limit the study to compute the reliability and Mean Time to Failure (MTTF). According to our approach, the necessary and sufficient condition for the proposed BGP distributed architecture to be functional, is to have *at least* one master controller card and *at least* one slave controller card under any master leadership both functional until a given time t. Let *r(t)* be the reliability of the process running on an individual controller card; it is defined as the probability that the controller card (master or slave) be functional at least until a given time t. Now, let us compute the reliability of the entire system composed of *m* masters and *s* slaves processes arranged as described earlier (see section 4.6). For this purpose, we define the following reliability events:

A: denotes the event that at least one RIM process is alive until time t

B: denotes the event that at least one slave process is alive until time t

The reliability, θ (t) of the whole system is the probability that the system composed of m masters and s slaves, is functional during the interval [0,t]. Since the failure in a master controller card does not imply a failure in a slave and vice versa, the two events A and B are considered independent. Therefore, the probability θ (t) is defined as follows:

θ (t) = P (system is functional) = P (A) x P (B) (1)

Equation (1) shows that the system reliability is a function of (a) the reliability of each master/slave processes at the controller card; (b) the total number of slaves; and (c) the total number of RIMs. We know that, every time we add a master controller card, practically a new RIM backup is added and new slaves are added as well. Thus, we can infer intuitively that the reliability is much more impacted by the number of RIMs than from the number of slaves. Our aim is to define a formula for the reliability model that mathematically expresses this fact. To compute the system reliability we assume that: (1) the number of slaves in a cluster based distributed architecture is by far bigger than the number of cluster heads (i.e., RIMs); thus, $s >> m$ (a realistic configuration may consist of up to 16 RIMs each managing 4 to 16 slaves); (2) the number of slaves per RIM is the same for all RIMs to ensure a good level of load balancing on all RIMs; and (3) RIMs and slaves are expanded exponentially as follows:

$m=2^k \quad and \quad s=2^{l+k}$ Where: l and k are integers and $l > k \geq 0$

The rationale behind the last assumption is that, we want to express the reliability for large number of cards (at least 4 RIMs) and to make the formula mathematically simpler to expand. This being said, even without this assumption, our findings still apply.

The reliability R of *m* RIMs is the probability, P(A), that *at least* one RIM is alive. This probability can be calculated using the binomial distribution:

$$P(A) = R = \sum_{i=1}^{m} \binom{m}{i} r^i(t)(1-r(t))^{m-i} = 1-(1-r)^{2^k} \qquad (2)$$

The reliability of slaves is the probability P (B); our aim is to compute an upper bound probability for a large number of slaves. For this purpose, this probability denoted by $p^{(l)}$ (is the probability that at least one slave, from 2^{l+k} slaves, is alive) can be computed as follows:

$$p^{(0)} = 1-(1-r)^{2^k} \qquad \text{(1 slave per RIM; total slaves = } 2^k\text{)}$$

$$p^{(1)} = 1-(1-r)^{2x2^k} = [1-(1-r)^{2^k}][1+(1-r)^{2^k}]$$

$$= p^{(0)} \times [1+(1-r)^{2^k}]$$

$$p^{(2)} = 1-(1-r)^{4x2^k} \quad = p^{(0)} \times [1+(1-r)^{2^k}][1+(1-r)^{2^{k+1}}]$$

.

.

$$p^{(l)} = p^{(0)} \times [1+(1-r)^{2^k}][1+(1-r)^{2^{k+1}}]...[1+(1-r)^{2^{k+l-1}}]$$

In order to transform the last equation into a geometric progression sum, we multiply and divide its Right Hand Side (RHS) by the term:

$$[1+(1-r)][1+(1-r)^2]....[1+(1-r)^{2^{k-1}}]$$

Thus, we obtain:

$$P(B) = p^{(l)} = p^{(0)} \times \frac{\prod_{i=0}^{k+l-1}[1+(1-r)^{2^i}]}{\prod_{i=0}^{k-1}[1+(1-r)^{2^i}]} \qquad (3)$$

By expanding and developing the term ($\prod_{i=0}^{k+l-1}[1+(1-r)^{2^i}]$) we obtain the following equation:

$$\prod_{i=0}^{k+l-1}[1+(1-r)^{2^i}] = \sum_{i=0}^{2^{k+l}-1}(1-r)^i \qquad (4)$$

To show the validity of Equation (4), let us consider a distributed architecture that consists of 2 RIMs (i.e. k = 1) where each RIM manages 8 slaves (i.e. l = 3); in this case, 2^{k+1}-1 = 3, (1-r) = α < 1 and the LHS of Equation (4) can be expressed as follows:

LHS = $(1+\alpha)(1+\alpha^2)(1+\alpha^4)(1+\alpha^8)$

$= (1+\alpha+\alpha^2+\alpha^3)(1+\alpha^4)(1+\alpha^8)$

$= (1+\alpha+\alpha^2+\alpha^3+\alpha^4+\alpha^5+\alpha^6+\alpha^7)(1+\alpha^8)$

$= (1+\alpha+\alpha^2+\alpha^3+\alpha^4+\alpha^5+\alpha^6+\cdots\alpha^{15})$

$= \sum_{i=0}^{2^4-1}\alpha^i$ = RHS

Obviously, the Right Hand Side (RHS) of Equation (4) is a geometric progression with rate 1-r. Since r < 1 (probability), the RHS of Equation (4) can be written as follows:

$$\sum_{i=0}^{2^{k+l}-1}(1-r)^i = \frac{1-(1-r)^{2^{k+l}}}{r} \qquad (5)$$

The denominator part of P(B) in Equation (3) can be approximated (given than α^i decreases when i increases) by the sum of the terms up to the power $2^{k-2}+2^{k-1}= 3*2^{k-2}$. Therefore, we distinguish the two formulas for small and big values of k:

$$\prod_{i=0}^{k-1}1+(1-r)^{2^i} = \sum_{i=0}^{2^k-1}1+(1-r)^i,\ k<2$$

and

$$\prod_{i=0}^{k-1}1+(1-r)^{2^i} \approx \sum_{i=0}^{3\times 2^{k-2}}1+(1-r)^i,\ k\geq 2$$

Since the last two sums are geometric progressions and r < 1, then

$$\sum_{i=0}^{3\times 2^{k-2}}(1-r)^i = \frac{1-(1-r)^{3x2^{k-2}+1}}{r} \qquad (6)$$

By combining Equations (5) and (6) in (3) we obtain:

$$P(B)=p^{(l)}=p^{(0)}\times\frac{1-(1-r)^{2^{k+l}}}{1-(1-r)^{3\times 2^{k-2}+1}}\quad ,k\geq 2 \qquad (7)$$

Equation (7) shows that for $l = 0$ we have equal number of slaves and RIMs (1 slave per RIM) and, as expected, the reliability of slaves is equal to the reliability of the RIMs: $[1-(1-r)^m]$.

Since r (t) < 1 and the number of slaves (2^{l+k}) is much bigger than the number of RIMs (2^k), Equation (7) can be simplified and rewritten as follows (e.g., if k=2 and l=3, then $(1-r)^{2^{k+l}} = (1-r)^{32} \approx 0$):

$$P(B) = p^{(l)} = \frac{p^{(0)}}{1-(1-r)^{3\times 2^{k-2}+1}} = \frac{P(A)}{1-(1-r)^{3\times 2^{k-2}+1}} \qquad (8)$$

Obviously, $P(B) = p^{(0)} = R$, when $k < 2$, because the term

$$\frac{\prod_{i=0}^{k+l-1} 1+(1-r)^{2^i}}{\prod_{i=0}^{k-1} 1+(1-r)^{2^i}}$$ in Equation (3) tends to 1 for $k < 2$.

Consequently, the reliability of the entire system is obtained by combining Equations (8) and (1):

$$\theta_k = \begin{cases} R^2 & , k < 2 \\ \dfrac{R^2}{1-(1-r)^{3\times 2^{k-2}+1}} & , k \geq 2 \end{cases} \qquad (9)$$

Equation (9) is very important since it computes the upper bound reliability of the system in terms of the number of available RIMs regardless of the amount of existing slaves (under the assumption s>>m).

Assuming that all controller cards have the same reliability function r(t), Fig. 4.6 plots the reliability r(t) of a single module controller card (x-axis) versus the reliability of the whole

system for different numbers of RIMs. We clearly observe that the overall reliability increases with the number of RIMs; however, this increase is not significant for high values of r(t) (>0.90). If the reliability of a single controller card drops to as low as 50%, the minimum number of master controller cards needed to reach a reliability above 99% is 4 (i.e. k = 2, 3 or 4). Conversely, to reach a system reliability of 99.999 while the individual reliability of controller cards is high, the minimum number of master controller cards needed is 2 (k=1).

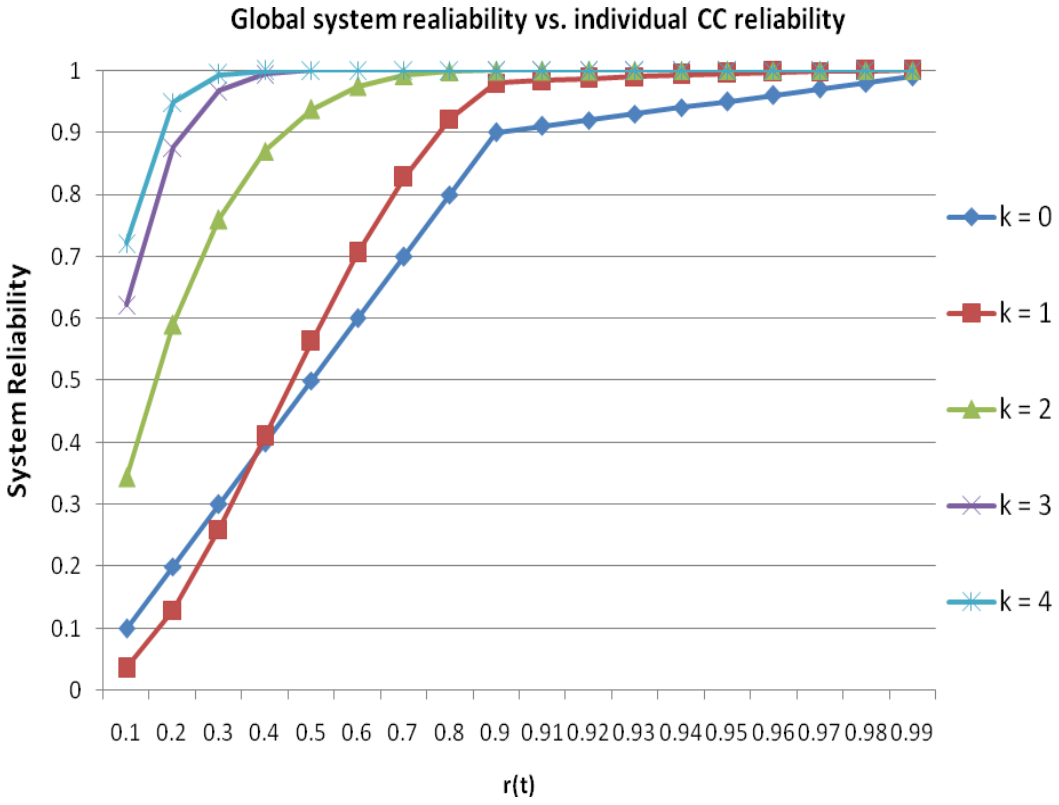


Figure 4.6 Overall system reliabiliy vs. a invidual reliability r(t)

4.6.2 Mean Time to Failure MTTF

The Mean Time to Failure is the common metric that measures the capacity of the system to stay available during a period of time. Our aim is to calculate the MTTF for the values k = 0 and k = 2 (1 RIM vs. 4 RIMs) to see the impact of redundancy on the availability. Assume that, the individual controller card reliability r(t) has the common

exponential distribution with a mean parameter equals to the constant failure rate λ; then the reliability r(t) and MTTF can be written as follows:

$$r(t)=e^{-\lambda t} \quad and \quad MTTF=\int_0^{\infty} r(t)\,dt=\frac{1}{\lambda} \quad t\geq 0 \qquad (10)$$

One RIM (k = 0)

Replacing k = 0 in Equation (9) we obtain:

$$\theta_0 = r^2$$

According to Equation (10), MTTF is defined as follows:

$$MTTF(\theta_0)=\int_0^{\infty} r^2(t)\,dt=\int_0^{\infty} e^{-2\lambda t}\,dt=\frac{1}{2\lambda} \qquad (11)$$

Equation (11) proves that MTTF of a system with only one RIM and multiple slaves (client-server architecture) is equal to 1/ 2*λ which is half the original MTTF of an individual card. In our distributed BGP architecture, this expected result reflects the fact that when no redundancy exists for the RIB storage part in BGP, the availability of the whole system will be reduced. Now, to evaluate the effect of redundancy, we calculate MTTF for a system that consists of 4 RIMs (k=2).

Four RIMs (k = 2)

For k = 2, the system reliability in Equation (9) can be expressed as follows:

$$\theta_2 = 1-(1-r)^4 = 4r-6r^2+4r^3-r^4 \qquad (12)$$

By applying the same computation as in Equation (10), we compute MTTF for a system composed of 4 master controller cards:

$$MTTF\ (\theta_2) = \int_0^{\infty} 1-(1-r)^4\,dt = \frac{4}{\lambda}-\frac{6}{2\lambda}+\frac{4}{3\lambda}-\frac{1}{4\lambda}=\frac{25}{12\lambda} \qquad (13)$$

With four RIMs (k=2), MTTF of the system is nearly double the MTTF of an individual controller card. More specifically, with only 4 RIMs, the availability of the proposed distributed architecture is twice the availability of a classical BGP monolithic architecture.

4.7 Conclusion

This paper presents a fault-tolerant mechanism that can be integrated with the distributed BGP architecture in [2]. The resulting architecture is a highly reliable cluster-based distributed architecture for the BGP protocol. Additionally, we have developed, in this paper, an analytical model that evaluates the reliability of the resulting fault-tolerant architecture. This model is very helpful for computing the availability gain of the distributed BGP implementation versus the availability of monolithic implementation of BGP. Indeed, by computing MTTF, we proved that the availability of the distributed BGP architecture with four RIMs, each managing multiple slaves, can be twice the availability of a monolithic BGP architecture while increasing the performance and scalability.

4.8 References

[1] *The open source zebra project* http://www.zebra.com

[2] Hamzeh W., A., Hafid. "A scalable cluster distributed BGP architecture for next generation routers." *IEEE LCN.* Zurich,Switzerland, 2009. 161-168.

[3] Govindan, R., Reddy, A. "An analysis of Internet inter-domain topology and route stability." *IEEE INFOCOM.* Kobe, Japan, 1997. 850-857.

[4] Labovitz, C., A., Ahuja, F., Jahanian. "Eperimental study of Internet stability and wide-area network failures." *International Symposium on Fault-Tolerant Computing.* 1999.

[5] Bornhauser, U., P., Martini, M., Horneffer. "Root Causes for iBGP Routing Anomalies." *IEEE LCN.* Denver, Colorado, USA, 2010. 488-495.

[6] Timothy, G., G., Wilfong. "An analysis of BGP convergence properties." *ACM SIGCOMM.* Cambridge, Massachusetts, USA, 1999. 277-288.

[7] Kushman, N., S., Kandula, D., Katabi, B. M., Maggs. "RBGP: Staying connected in a connected world." *NSDI.* 2007. 341-354.

[8] Bonaventure, O., C.Filsfils, P.Francois. "Achieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures." *IEEE/ACM transactions on networking* 15, no. 5 (2007): 1123-1135.

[9] Wang, F., L., Gao. "A backup route aware routing protocol-Fast recovery from transient routing failures." *IEEE INFOCOM.* AZ, USA, 2008. 2333-2341.

[10] NISCC (UK Gvt.) Best Practices Guidelines: BGP (note 5 on page. 8), April 2004.

[11] Sangli, S., E. Chen, R. Fernando, J. Scudder, Y. Rekhter. "Graceful Restart Mechanism for BGP", RFC 4724, 2007.

[12] Yang, L., R., Dantu, T., Anderson, R. Gopal."Forwarding and Control Element Separation", RFC 3746, 2004.

[13] Rekhter, Y. "A Border Gateway Protocol 4 (BGP-4) ", RFC 4271, 2006.

[14] Abley J. "IPv4 Multihoming practices and limitations", RFC4116, 2005.

CHAPITRE V

5 DRTP: A Distributed Parallel Approach for BGP Routing Table Partitioning

Wissam Hamzeh, Abdelhakim Hafid

Abstract—**The rapid growth of routing tables represents a major challenge facing the scalability of BGP and indeed the whole Internet infrastructure. In this paper, we introduce DRTP, a novel parallel distributed algorithmic scheme for partitioning the BGP routing table on multiple controller cards, where we exploit parallelism to improve both the lookup speed and the scalability of the RIB (Routing Information Base). The proposed scheme improves the lookup performance by letting unrelated tasks, such as Best Match Prefix (BMP) lookup and BGP decision process, to be executed in parallel at different controller cards. Benchmarking experiments on real BGP data show that our proposal outperforms classical central lookup mechanisms with a reasonably acceptable cost, while it increases considerably the space scalability of the BGP routing table. Moreover, the analytical evaluation proves the effectiveness of DRTP in obtaining a significant parallel performance speedup with acceptable tradeoff of internal synchronization cost.**

[**Status**]: This paper is submitted to the journal of Computer Networks. The ideas presented in this paper are largely based on the following published paper: “A Distributed Parallel Approach for BGP Routing Table Partitioning In next Generation Router”, IEEE LCN 2010, pp. 480-487. Denver, Colorado, USA.

5.1 Introduction

BGP router scalability can be assessed according to two metrics: the routing table size and the number of connected peers. During the last several years, the BGP routing table has been growing exponentially. The introduction of CIDR [1] by the end of the last decade succeeded in reducing the expansion rapidity of the routing table by enabling route aggregation in a single prefix. While this mechanism slowed down the growth of the routing table for short period, this later has regained its exponential growth trend by the year 2004 [14]. The routing table is expected to reach a size of 1,000,000 entries within the coming few years. Besides, current core routers may easily attain hundreds of connected BGP peers simultaneously. In order to augment the router capacity and absorb the expected load increase, recent contributions propose to decompose the BGP functionalities at the control plane [4]. During the last decade, several studies have been conducted on router scalability; major research efforts have focused on distributed router architectures (e.g., [6], [10], and [11]) but none has proposed a distributed architecture in which the BGP routing table is partitioned at the control plane. In this paper, we elaborate a novel scheme called DRTP (Distributed Router Table Partitioning) aiming at partitioning the BGP routing table at multiple controller cards, so that lookups and updates can be performed in parallel on multiple cards. This partitioning scheme offers scalability in terms of memory usage by enabling a steady growth of the BGP routing table with an acceptable communication overhead without modifying the core of BGP. Our key contribution is the distributed partitioning of the BGP routing table and the parallelization of the best match prefix (BMP) search algorithm in the BGP context.

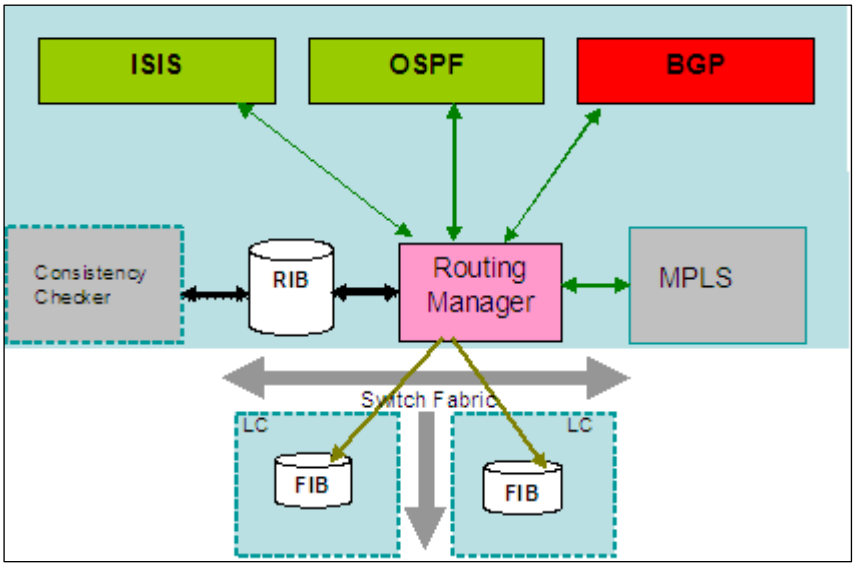


Figure 5.1 General view of router components

According to the IETF ForCES RFC [20], core routers consist generally of two components: control plane and forwarding plane (see Fig. 5.1). The control plane is responsible for building the RIB from reachability information and mapping the reachable routes into the FIB (Forwarding Information Base) at line cards (LCs) in the forwarding plane; the forwarding engine at each LC classifies and forwards packets. At control plane several protocols are running as processes competing for the same routing processor to build the Routing Information Base (RIB). The RIB is constructed using multiple tables associated with each of these protocols. Mainly, the IGP (Interior Gateway Protocol: ISIS and OSPF) table holds reachability information about destinations inside an Autonomous System (AS) coming from ISIS or OSPF, and the BGP table holds inter-domain reachability information. The BGP RIB (or BGP table) includes prefix entries which define the reachable destinations, attribute entries which describe the characteristics of BGP prefixes and filter lists that define the AS policies for accepting or rejecting incoming routes. In core routers, the BGP RIB is accessed most frequently by three asynchronous processes: (1) the best path selection process; (2) the inconsistency checker process; and (3) the next-hop update process which is invoked upon IGP reachability changes from the ISIS or the OSPF protocols. In the first process, BGP continuously receives update messages about different destinations from adjacent peer routers whenever the Internet topology changes. If the message type is a withdraw, then the table is accessed to remove the

corresponding prefixes; if the message is an announcement for new routes then BGP executes a complex algorithm to select the best path among existing routes and the received ones. To this end, BGP performs lookup into the RIB to make next-hop resolution for the learned prefixes then invokes the decision process to make the comparison for best path selection. As a result, BGP decides whether the received route can be considered as first best, second best or simply rejected in case its next-hop is not reachable. The second process [10] fetches regularly the BGP table to detect route inconsistency between FIB and RIB. This is done by scanning periodically all the prefixes in the FIB to find out whether they are still valid in RIB or not. The third process reacts to changes in the IGP topology; that is, if a next-hop address for a BGP route resolved via IGP is down, the routing manager process must walk down the BGP RIB and remove all routes that use this address as next-hop. In this paper, we propose a solution to remove the bottleneck at the BGP RIB by allowing multiple controller cards work collectively to replace the centralized paradigm of the BGP RIB. The proposed DRTP increases (1) the scalability of the BGP routing table by making a horizontal division on prefix length and a vertical division on prefix range of the whole table and (2) the lookup performance by exploiting parallelism and distributing adequately the load on multiple controller cards. The overhead introduced by DRTP to process addition and deletion of prefixes, is minimal. Indeed, as shown in [18], routers receive update messages at high frequencies; fortunately, most of these messages involve only changes in the routes and do not add or delete prefixes. This feature minimizes the synchronization among distributed nodes and therefore increases the parallel speedup.

The remainder of the paper is organized as follows. Section 5.2 presents related work. Section 5.3 presents the details of the proposed partitioning scheme. Section 5.4 presents an algorithmic solution of the proposed approach. Section 5.5 presents a complexity and analytical evaluation of DRTP. Section 5.6 evaluates DRTP. Section 5.7 concludes the paper.

5.2 Motivation and related work

5.2.1 Motivation

In major backbone routers, the RIB has been found to change frequently. As links go down and come back, routing protocol messages may cause the routing table to change continuously. Changes include addition and deletion of prefixes; in addition, the modification of next-hop information for existing prefixes in IGP topology incurs deletion of prefixes in the BGP routing table. These changes often occur at the peak rate of a few hundred prefixes per second and at the average rate of more than a few prefixes per second [18]. This reflects only the number of feasible prefixes to be added or removed. Besides, BGP keeps a large amount of backup routes to the same network entry in its RIB for a future replacement in case of withdrawal of best routes. Even though, BGP routing computation at control level is less time critical than forwarding at line card, the controller card may easily become a critical bottleneck when the BGP routing table becomes huge; our contribution proposes a solution to overcome this bottleneck. Although our proposed partitioning scheme focuses on RIB partitioning at control plane, we believe it can be easily extended to partition the FIB at line cards to increase its forwarding capacity. Our contribution differs from existing work, since it exploits parallelism by using multicast to communicate with the distributed nodes (nodes are controller cards). The proposed scheme allows an increase in the lookup capacity by taking benefit of the ultra-fast switch fabric in next generation routers, where communication latency drops to as low as 20 ns [21]. The partitioning scheme that we propose in conjunction with the multicast, guarantees that multiple nodes perform the search in parallel and only one partition node (i.e., controller card) answers the lookup or update query.

5.2.2 Related work

Most related work mainly focuses on partitioning the routing table at line cards (LCs) to increase the lookup capacity. Related work can be classified in three categories: (1) logical partitioning [2]: partitions the routing table into several partitions on the same node to increase the BMP (Best Match Prefix) lookup speed; (2) physical partitioning [3]:

partitions the routing table physically into several LCs; and (3) physical control plane partitioning [4, 16]: partitions the routing table physically into several controller cards.

Scudder et al. [16] propose a partitioning scheme that uses the range of prefixes to distribute the RIB aiming to increase space scalability at control plane. Although, details of the partitioning algorithm are not present, the idea consists of segmenting the RIB into multiple servers, saying two for simplicity; one server serves routes matching the prefix 0.0.0.0/1 while the second server serves routes matching the prefix 128.0.0.0/1. The client process, such as the routing manager or the consistency checker, needs to examine the prefix in the lookup or update query in order to know the range to which it belongs and therefore the server into which the query should be directed. Although the scheme is simple, its performance is limited. Indeed, the traffic distribution of update messages per prefix range is very irregular and not well balanced [14]; in fact, much more lookups are performed on high-range prefixes (192.0.0.0) than low-range prefixes (0.0.0.0). Partitioning routing tables based on prefix range will not provide the desired load balancing and thus will not allow for fully exploiting parallelism; however, it increases the space scalability of the routing table.

Virtual aggregation [22] is a long-term proposal for FIB size reduction. The mechanism focuses on a partitioning scheme of the router table where every ISP (Internet Service Provider) router inside an AS is responsible for managing a given range of prefixes of the routing table. Different ISPs inside a same AS collaborate in exchanging the messages through virtual tunnelling. The key advantage of virtual aggregation is that it can be deployed independently inside ASs; however, it requires special changes in routers to implement the forwarding of virtual prefixes.

Tzeng [3] proposes a technique inside the router, called SPAL (speedy packet lookups), that partitions the FIB; it uses SRAM cache to share the learned locality of the forwarded routes at different line cards. In SPAL, the routing table partitioning is done via a program that makes an exhaustive search to find pivot bits that optimally divide the routing table. Although SPAL proved efficient in fragmenting the routing table at line cards into merely equal partitions, unfortunately this approach may exhibit two drawbacks if applied at the controller level: (1) the partitioning method to find the pivot bits is time consuming and its complexity depends on the prefix length which limits its efficiency for the IPv6 128-bit

prefixes; (2) before carrying out the partitioning method, the approach assumes the presence of the whole routing table, which is not the case at the controller level since the BGP daemon builds incrementally its routing table from peer routers. Besides, the pivot bits can lead to ephemeral balanced partitions that may last only few days. Indeed the routing session lifetime may last several weeks (even months) before a reboot; during this period and due to the rapid and continuous changes in the routing table, the previously calculated pivot bits may no longer lead to the same initially desired optimal partitions. Akhbarizadeh [2] presented a technique to partition the routing table into subsets that can be searched in parallel; all partitions are replicated at all line cards (LCs). The size of the routing table at each LC node is not reduced when increasing the number of LCs. Hence, the scalability of the routing table in terms of size was not considered.

5.3 Proposed BGP table partitioning scheme

In this Section, we start by discussing the characteristics of the BGP routing table, such as its structure, growth trend and prefix properties. Then, we present our proposed partitioning solution for the BGP routing table.

5.3.1 BGP routing table structure

A BGP route is defined as a unit of information that pairs a destination prefix with the attributes of a path to this destination. BGP stores a huge amount of routes in its routing table to provide correct reachability information to the LCs. The various blocs constituting the BGP routing table are: (1) Adj-Rib-In: the set of all raw routes that have been advertised to the local BGP speaker by its peers; (2) Loc-Rib: the set of all active routes; and (3) Adj-Rib-Out: it contains the routes that passed the output policy module and advertised to specific peers by means of the local speaker's UPDATE messages.

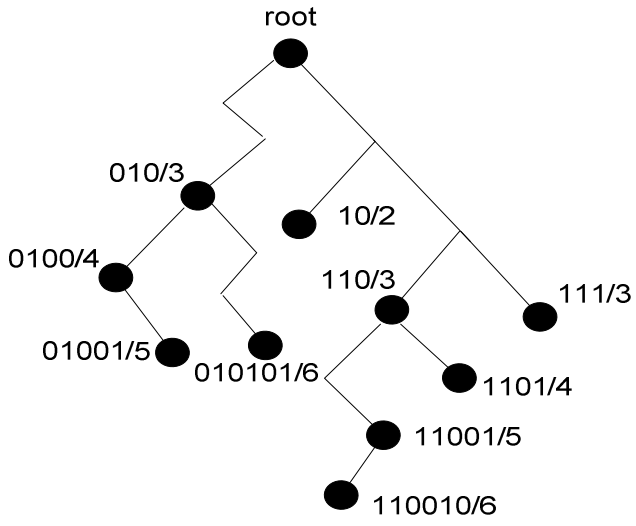


Figure 5.2 BGP radix-tree routing table structure

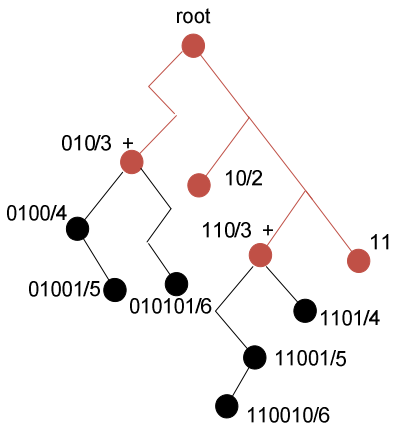
Several existing contributions propose approaches to represent efficiently the routing table in a convenient data structure suitable to perform the BMP lookup. The most popular approach is the usage of trie or radix-tree [23] (see Fig. 5.2). Although the conceptual model described in RFC 4271 [12] distinguishes between Adj-Rib-In, Loc-Rib, and Adj-Rib-Out, this does not imply that an implementation must maintain three separate copies of the routing information. The implementation choice (3 copies vs. 1 copy with pointers) is not constrained by the BGP protocol; it is up to the implementer choice. In this paper, we assume that RIB is implemented as one copy with pointers. We classify the prefixes in a BGP routing table according to three categories: (1) Stand-alone prefix: a prefix that has no subset or superset in the BGP routing table; (2) Less specific (Sub- root) prefix: a prefix that has a least specific prefix with subsets in the BGP routing table; and (3) More-specific prefix: a prefix that is a subset of some other prefix. For example, consider the ten prefixes in the routing table shown in Fig. 5.2. The prefixes 10/2, 111/3 are stand-alone prefixes while 010/3, 110/3 are less specific prefixes, and the rest are more-specific prefixes.

5.3.2 Prefix length-based distributed partition

Classical sequential search for fixed length key using hash partition is trivial. However, since the search is performed in terms of best matching, the solution is more tedious, because of the prefix overlapping property [7]. For better understanding, let us consider a sample routing table (see Fig. 5.3) composed of ten prefixes that can be distributed according to their lengths on two nodes A and B (nodes are controller cards); prefixes with lengths 1 to 3 are stored in their home node A and prefixes with lengths 4 to 6 are stored in their home node B. Let us consider a naive distributed implementation consisting of one dispatcher node representing any process that transmits a BMP lookup query for prefix 10111/5. Since the length of the prefix is 5, by performing a simple hashing, the dispatcher transmits the query to node B. Clearly, the lookup will fail since 10* (in node A) is a longest prefix match. This example shows that partitioning the routing table according to only prefix length with simple hashing is not a desirable solution.

A[1-3]	B[4-6]
10*	0100*
010*	1101*
110*	11001*
111*	01001*
	110010*
	010101*

(i)



(ii)

Figure 5.3 (i) Prefix length-based partition (ii) Tree representation, prefixes with + sign are "juncture prefixes"

Indeed, the shortcomings of prefix length-based partitioning are due to the fact that the entries of the routing tables are inherently not disjoint [7]. Srinivasan et al. [8] propose a technique to transform the routing table prefixes into a set of disjoint prefixes by using the concept of 'Controlled Prefix Expansion'. This technique can be used efficiently in the case of same node logical partitions. However, it is not appropriate in the case of partitions over physically distributed nodes; indeed, managing every expanded prefix will generate a considerable overhead in terms of space and exchanged messages among the distributed nodes.

5.3.3 DRTP

In opposition to existing partitioning schemes, our approach for partitioning the BGP table is achieved in two dimensions: the length and the range of prefixes. In this section, we first present the definitions and notations used in the remainder of this paper; then, we discuss parallelism concepts of interest before presenting the proposed scheme, called DRTP, which distributes efficiently the BGP routing table into multiple distributed nodes.

Definitions and notations

In this section, we define the terms that will be used in the remainder of this paper. Let P and Q be two partitions of an original routing table T, where P holds prefixes of length smaller than L_1 and Q is a subsequent partition holding prefixes of length within the interval $[L_1, 32]$ (case of IPv4) or $[L_1, 128]$ (in case of IPv6). Let p and q be two prefixes in two partitions P and Q:

Definition 1: We say that a prefix p in a partition P spans a prefix q in a subsequent partition Q, iff:

Length (p) < Length (q) and p is a match of q.

Definition 2: Let p and q be two prefixes that belong respectively to two nodes P and Q. We define the prefix p to be a *juncture* between the partition P and the subsequent partition Q, iff: $p \in P$, $\exists q \in Q$ such that p spans q and there is no x, such that x spans q and x is more specific to p.

Definition 3: Two prefixes p and q are *disjoint* iff p is not a prefix of q and q is not a prefix of p.

Lemma 1 (Non-transitivity): Let p, q and r be three prefixes that belong to three partitions P, Q and R (i.e. length (p) <length (q) <length (r)); if p is a juncture to q (p is a best match to q in P) and q is a juncture to r (q is a best match to r in Q) then p cannot be a juncture to r.

Proof: the proof is done by contradiction. Let us assume that p is a juncture to r. Since p is a juncture to q then q is more specific than p and since q is a juncture to r then q spans r. This contradicts the juncture definition because if p is a juncture to r, then there is no prefix that spans r and is more specific to p. Thus, the assumption is false and p cannot be a juncture to r.

Distributed parallel length-based partition

Parallel processing coupled with an efficient distributed partition scheme can significantly improve the BGP performance because it removes the central processing bottleneck of current BGP serial implementation. Two directions can be investigated to parallelize the BGP functionalities. The first direction is the functional decomposition, also

called *task-level parallelism* where separate functions are distributed on physically distributed processing nodes [4] or local threads [19]. The principal gain from using task-level parallelism is the speedup and bottleneck removal. The second direction is data decomposition, also known as *data-level parallelism;* it breaks down the data on multiple processing units. Rather than having one processing unit performing all computations, data decomposition suggests a distribution of the computations on n processing units, each performing $1/n^{th}$ the work in the ideal case. The key benefit of this approach is a tradeoff between performance and storage scalability of the whole system. In this paper, we propose to use data-level parallelism to partition the BGP routing table on multiple processing nodes, in which every node performs the same task on its local data partition. In data-level decomposition, if data partitions are disjoint and tasks are interdependent inside the same partition node with no need to inter-node synchronization, parallelism would be desirable and linear speedup can be achieved. Unfortunately, this is not always the case in real problems because many algorithms are conceived to run in a serial manner and inadvertently introduce dependencies that impacts parallelism. This is the case of best matching prefix calculation in BGP routing tables. To overcome this limitation and exploit parallelism, we propose to remove such dependency through a simple transformation to eliminate the problem of prefix overlapping. Therefore, distributed node partitions of the routing table will be able to find out autonomously the lookup results with minimal inter-nodes synchronization. After this transformation is performed, the parallel best matching prefix lookup on distributed nodes will be possible and the system will behave as if the partitions are disjoint. Consequently, parallel search on the transformed routing table can be realized by a dispatching process (e.g., routing manager) that multicasts the lookup query to all distributed nodes. Upon receipt of the query, every recipient node searches for the query prefix in its local data partition; a unique successful answer can then be returned by one node. Moreover, route updates will be performed in parallel as soon as the multicast query reaches the partitions (see next section for details).

Methodology

The basic idea of the solution is to replace single unicast message query to the entire RIB with a multicast to multiple nodes. These nodes form a group of partitions with common multicasting address and each partition holds a portion of the RIB. In Fig. 5.4, the RIB is divided into two partitions and each partition resides in a separate node called horizontal node. “nodes”, “horizontal nodes” and “partitions” are used interchangeably throughout the paper.

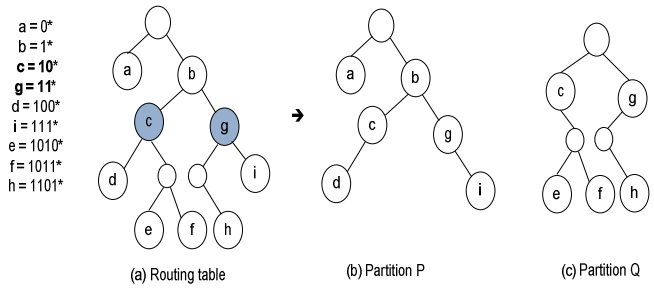


Figure 5.4 Length-based partitions with juncture replication

The proposed transformation of the RIB radix-tree is based on replicating a specific subset of prefixes into multiple distributed partitions. Indeed, by analyzing the structure of the routing table, we identified a set of prefixes, called juncture prefixes or simply ‘junctures’ of key importance to solve the dependency problem. Juncture prefixes are prefixes in a partition spanning other prefixes in subsequent partitions. In fact, when performing lookup in a prefix length-based partition of the routing table, we are not concerned with dependent (overlapped) prefixes whose length is smaller than the juncture’s length inside a same partition. Indeed, only when a lookup best match is a juncture prefix, a special synchronization must be done. For example, let us consider the case when partitions P and Q (see Fig. 5.4) receive at the same time (via multicast) a BMP lookup query to the prefix “00010/4”. Since the longest match of this prefix is “0/1” which is a stand-alone prefix and not a juncture, then P is sure that no longer match exists in Q; thus, P can answer the query. However, if both partitions receive lookup query to the prefix “1101/4”, then the longest match at P is the juncture prefix *g* (“11/2”); in this case, P is not sure whether g is the only and unique best match since g spans another set of prefixes in Q and one of these prefixes may represent the best match of “1101/4”. Thus, if P sends back its local best match (g),

the query transmitter will receive ambiguously two match prefixes to the same query: “11/2” from P and “1101/4” from Q. A possible patch solution would require the transmitter to select, among the multiple matching results (“11/2” vs. “1101/4”), the longest one to be the best match of its query (i.e., “1101/4”). Clearly, such a solution is not desirable because of its poor scalability. To eliminate the juncture dependency among nodes, we investigated two approaches: (1) controlled prefix expansion [8] of the juncture prefixes; and (2) replication of the whole juncture prefixes with an adequate management of the replicas. Our investigation led us to not consider controlled prefix expansion because it will generate a significant overhead; indeed, the amount of the prefixes to add and the number of internal synchronization messages will be considerably high.

The replication-based approach consists of the replication, when needed, of the juncture prefixes. Indeed, by replicating a juncture along with its children from node P to node Q (see Fig. 5.4), Q will be able to answer any lookup query concerning a prefix best match of the juncture; if the query concerns a prefix match of one of the juncture’s children, then Q will not answer this query; in this case, P (home node of the juncture) will answer this query. Reciprocally, if P (home node of the juncture) receives a query concerning a prefix match of the juncture, will not answer the query; in this case, Q, that has the replicated juncture and its children, will answer and return the query result. Clearly, this simple replication mechanism provides results as if prefixes among distributed nodes are disjointly partitioned; it requires no extra changes to prefixes in the routing table. We implemented this mechanism by making use of two auxiliary tables, called *disable table* and *juncture table;* disable table is a radix-tree data structure that holds the juncture prefixes at the home node; juncture table is a radix-tree data structure at replica nodes holding the replicated juncture prefixes along with their children.

DRTP distribution

In order to produce efficient routing table partitions, we started by studying the characteristics of BGB routing tables. Indeed, by inspecting the prefix length distribution of a BGP routing table, valuable information can be retrieved about the nature of the prefix distribution as well the aggregatability level of the prefixes within the routing table. Studies

[14] have shown that the partition of BGP routing tables corresponding to the length interval [24-32] in IPV4, hosts more than 55% of the routing table while the partition corresponding to the length interval [8-16] contains no more than 3% of prefixes. Moreover, a study on a routing table [14] has revealed that in general, 40% of the prefixes are standalone prefixes, 5% are less specific prefixes and 55% are more-specific prefixes. Bu et al. [17] found out that prefixes of length 17 to 25 are the fastest growing prefixes in routing tables. Moreover, Sun et al. [15] reported that prefixes in general tend to be disjoint prevalent; that is the number of overlapping prefixes is quite small with respect to the complete routing table. Based on these findings, DRTP starts by partitioning prefixes in a first dimension, according to their lengths in a horizontal partition as shown in Fig. 5.5. Although the prefix length-based distribution is not well balanced, it reflects a systematic growth trend, which is common to all routing tables as proved statistically in [14]. Therefore, by suitably regrouping prefixes according to their prefix length interval, we will be able to produce efficient partitions applicable in any real routing table. In DRTP, the initial set of prefixes in the routing table is decomposed into multiple subsets each, containing a list of prefixes falling in a given length interval; each subset is then assigned to a node, called its home node. This partition scheme produces a group of K nodes where K< L and L is the number of prefix lengths available in the routing table. Note that, the length interval associated to each node is chosen based on known (empirical) prefix length distribution [14, 15, 17]. For example, in IPV4, a well-balanced partition can be obtained if prefixes of length inside the interval [8-23] are regrouped in one partition node and prefixes of lengths bigger than 23 are regrouped in a second partition node. Once the partitions of the routing table are computed, we determine the juncture prefixes. This gives a good indication whether a longest match of the searched/queried prefix, in which the juncture is best match in a given partition node, exists in a subsequent partition.

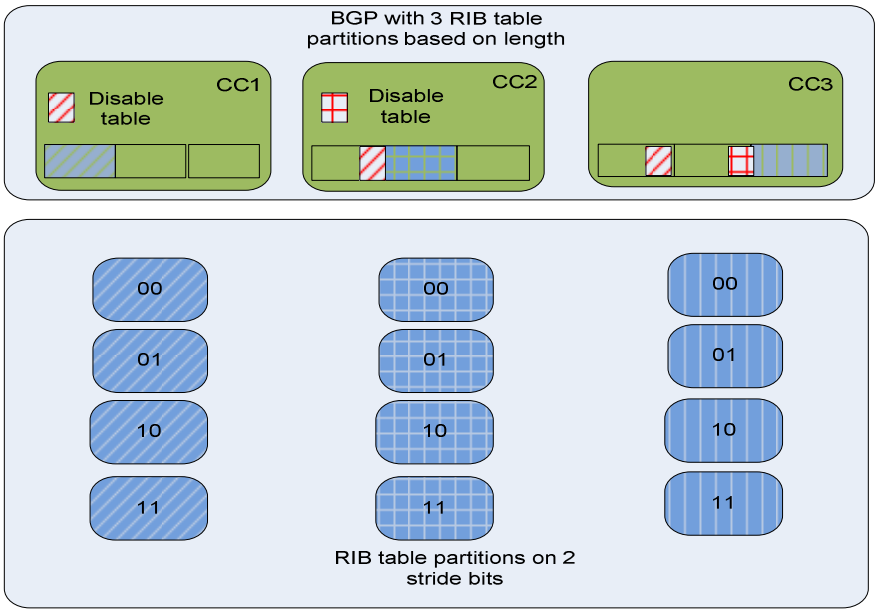


Figure 5.5 BGP Routing table partitioning at control plane

For better understanding, let us consider the case of only two partitions (see Fig. 5.3). Let us assume a prefix p, which belongs to a home node A, is a juncture to another prefix q in node B; according to DRTP, p will be replicated in B; if p has children in the level just after the p level, then these children will be replicated in B as well. For instance, if prefix 110* in A is a juncture of another prefix 110010* in B, then 110* must be replicated in B. Consequently, A removes the juncture (only and not the children) from its trie and stores it in an auxiliary data structure called "*disable table*" while B inserts the received juncture prefix along with its children (if any) in a table called "*juncture table*". Thus, the nodes can update the replicated juncture elements in parallel. Given a modification in a juncture prefix, the two nodes must collaborate to synchronize their tables. Recall that, as BGP keeps several alternative paths to the same route in its Adj-Rib-In, in DRTP, only home nodes keep track of multiple paths to their prefixes, while nodes hosting the replicated juncture prefixes need to keep only best route of that juncture prefix.

After building all nodes horizontally (Fig. 5.5), a second level of partitioning is done vertically according to fixed stride bits. A stride s is the set of the most significant bits, in a prefix, which defines its range. With a stride of s bits, 2^s horizontal groups can be constructed (Fig. 5.6).

DRTP architectural view

The DRTP parallel architecture (Fig. 5.6) consists of two key components: (1) horizontal and vertical partitions; and (2) the use of multicast to send the query to a group of horizontal nodes that correspond to the stride bits of the query prefix. In [4] the BGP routing process is split into two portions: front-end and back-end. The front-end process communicates and receives routes from external BGP routers while the back-end process applies best path decision process to select best routes. In DRTP, the same paradigm is applied, with the only difference that back-end process is replicated in all RIB partitions to perform the decision process in parallel. When the front-end receives a valid route, it multicasts this route to the appropriate node where back-end node triggers best path selection process and compares the received route with existing ones and updates RIB accordingly. To appreciate the benefit of using the juncture property, let us consider the example where we have two horizontal nodes A and B (see Fig. 5.3). Upon receipt of the multicast message, A will not answer the query if the query prefix best match is a juncture prefix in the disable table. In that case, A knows that B contains a longer match of the searched prefix, and if not, at least it will have this matched prefix itself because it is already replicated in B; thus, B can answer the query. Simultaneously, B will answer the query if the query prefix best match is the juncture and not its children; indeed, if it matches the children, then B will know that A has a longest match. It is worth noting that junctures are the only prefixes where a multicast BMP query requires the synchronization of both A and B in order to answer the query; apart from juncture prefixes, all prefixes are disjointly distributed in A and B and only one node has a valid match at a time. Consequently, if R is the total number of partition nodes and s is the stride bits, then only $R/2^s$ will be involved in the lookup or update for a given prefix; this is a small and constant number that makes DRTP scalable. For example, if we consider two home nodes that correspond to two length intervals and stride length s = 2, we will have a total of 8 partitions (i.e., 8 partitions) composed of four multicast groups (see Fig. 5.6).

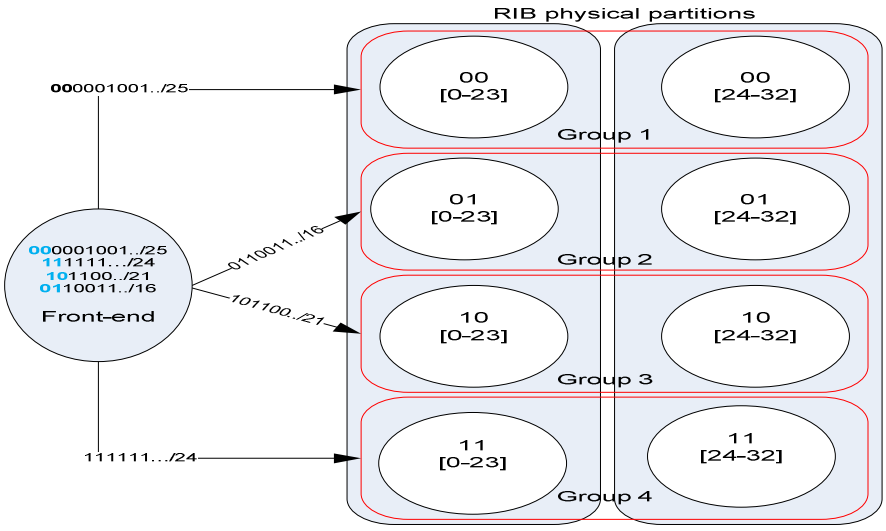


Figure 5.6 Two horizontal and four vertical RIB partitions in DRTP

Any lookup query must then be mulicasted by the front-end process to a multicast group composed of two nodes (Fig. 5.6). Since the routing table is disjoint prevalent [15], we expect the size of the disabled table, holding junctures in each node, to be relatively small compared to the size of the whole partition (see Section 5.6).

Distributed BMP lookup in DRTP

This section presents an example of the BMP lookup process using DRTP. In table I, prefixes are assigned, according to three length intervals, to nodes A, B and C with stride s = 2. Let us assume that the front-end process receives from an adjacent peer an update message that encapsulates the following prefixes in its Network Layer Reachability Information (NLRI):

{101/3;111/3;0110/4;00010/5;000010/6;0000100/7;010010001/9;1011110000/10;1100001110/10;1110000001100/13;0000100010001000010000100/20;01111100000000000000/19;10111111111111111111111/21;11100000011001111111110/21}.

The front-end process classifies the prefixes according to their stride bits and encapsulates all same stride bits prefixes in a separate message; then, it transmits the newly formed messages to the corresponding groups. The recipient nodes execute best route decision process, insert the prefixes and synchronize their juncture tables accordingly. As illustrated in table I, the duplicated juncture prefixes in A, B and C are shown using bold font.

TABLE 5.1 RIB VERTICAL DIVISION ON 2-BIT STRIDE AND 3 HORIZONTAL PARTITIONS ON LENGTH

S =b_0 b_1	A[1-6]	B[7-18]	C[19-32]
00	00010* **000010***	**0000100*** **000010***	000010001001000010000* **0000100***
01	0110*	010010001*	0111110000000000000*
10	**101***	1011110000* **101***	10111111111111111111* **101***
11	**111***	1100001110* **1100*,111*** **1110000001100***	11100000011001111111* **1110000001100***

Let us assume that, later on, the dispatcher receives a BMP lookup for prefix 000010001000100001000/21; in this case, the dispatcher multicasts an internal message to three nodes with strides 00; upon receipt, the back-end processes of the three nodes look in parallel for that prefix. Node C {00} finds a prefix match (see Table I) in its main trie and thus sends back the results (i.e., answer) to the dispatcher. Meanwhile, nodes A {00} and B {00} find a match in their respective disable tables; thus, they terminate the search without returning any results, since they know that a longer match exists elsewhere. DTRP guarantees that a lookup query is answered by only one node.

5.4 Algorithms

The BGP routing table changes frequently; this leads to a continuous change in prefixes, such as addition and removal. This will definitely impact the distribution of junctures in the partitions. A key operation in our contribution is the management of the juncture prefixes replicated in multiple distributed nodes. In this section, we present the details of the operation of DRTP processing Insert, Withdraw and Lookup queries.

5.4.1 Insert

The add route in typical BGP implementation consists of applying the Breaking-tie algorithm, between a received route p and some existing best match route r found in the trie. Let assume we have three horizontal nodes N0, N1 and N2 managing prefixes of length in intervals [a,b],[b+1,c] and [c+1,d] respectively (a<b<c). Since the prefix is

multicasted to all horizontal nodes, each receiving node can be: (1) a home node of the prefix; (2) a node managing lower length prefixes than the received one; (3) a node managing higher length prefixes than the received one. The routing and juncture tables are built incrementally via update messages received from peer routers. During the update process in DRTP, two messages DUP (DUPLICATE) and JUN (JUNCTURE) are exchanged internally. A DUP message is directed from nodes managing higher length prefixes than p (e.g. N1) to the home node of p (e.g. N0). The purpose of the DUP message is to notify the home node N0 that its prefix p is found to span a prefix in N1. Thus, p must be considered as a juncture. Reciprocally, a JUN message containing a prefix p and its children prefixes is sent from a node managing lower length prefixes (e.g. N0) to a node managing higher length prefixes (e.g. N1) in order to announce a new juncture prefix.

Table 5.2, shows the pseudo-code of the prefix insert processing in DRTP. The algorithm consists of three blocs (starting at lines 1, 5 and 21 respectively), where each node executes in parallel its part according to the length of the incoming prefix: (1) upon receipt of an insert query, with a destination prefix p, the home node (N1) of p applies the decision process (classical BGP execution) and inserts the prefix in the routing table (line1-3). If the exact match of p already exists in the main trie or disable table then the decision process will only update route attributes. If p is a new prefix that has, according to its length, N1 as home node then nodes N0 and N2 that received the query in parallel (query multicasted to the three nodes) apply the BMP operation in their tables; accordingly a JUN or DUP message will be sent to N1, if p is found to have a juncture relationship. In the worst case scenario, the total number of messages that needs to be exchanged per route update during the update process is two (1 JUN and 1 DUP, see lines12-13 in table 5.2).

Upon receipt of DUP, a home node (e.g. N1) considers the prefix p (piggybacked in DUP) as juncture prefix; it removes p from its main trie and inserts it in its disable table; it also sends a copy of this prefix children to the node that already sent the DUP (N2). In opposite, when a JUN message is received by N1, this later inserts the piggybacked juncture prefix along with its children in its juncture table for future lookup use. Although the following algorithm treats all cases for multiple horizontal nodes, by dealing with only two partitions a significant part of the algorithm will be reduced.

TABLE 5.2 PARALLEL ALGORITHM TO UPDATE A ROUTE IN DRTP

Algorithm: insert

Input: prefix p **Output**: message sent; RIB updates

```
/*Let MT denotes the Main Trie*/
/*Let DT denotes the Disable Table*/
/*Let JT denotes the Juncture Table*/
1.  if (LowerBound ≤ length (p) ≤ UpperBound) /* home node*/
2.  begin
3.    Normal  BGP decision process and route insert;
4.  end
5.  if (length (p) > UpperBound) /*A node that manages lower
                                    length prefixes than  p */
6.  begin
7.    q = BMP (p)
8.    if (q == NIL or q ∈ DT)
9.      do nothing
10.   else if (q ∈ JT)  /* this node in the middle; Lemma1*/
11.   begin
12.     send (JUN, p, q) to p's home;
13.     send (DUP,p) to q's home
14.   end
15.   else if (q ∈ MT) /*New juncture detected*/
16.   begin
17.     send (JUN, p, q) to p's home;
18.     set q in DT as disabled to p's home node
19.   end
20. end
21. if (length (p) < LowerBound) /*A node that manages higher length prefixes than p*/
22. begin
23.   q = BMP(p)
24.   if (q == NIL and p does not span a prefix r)
25.      do nothing;
26.   else if (p spans a prefix r and p is best juncture)
27.   begin
28.     send (DUP, p) to home node of /*New juncture found. notify
                                        home node that p is juncture;*/
29.     wait_children(); /*  home node of p sends back the  children of p*/
30.   end
31.   else if (q ∈ JT)
32.   begin
33.     Update_children(); /*verify if p is a juncture child*/
34.   end
end  /*length(p)<lowerbound*/
```

5.4.2 Withdraw

Withdrawal mechanism in typical BGP implementation is triggered either because of a deleted peer or due to the reception of an update WITHDRAW message. In normal BGP execution, when a route is withdrawn (deleted), BGP selects a next best route from its RIB to be a replacement best route of the deleted one. A WITHDRAW message containing the deleted route and its replacement are propagated to adjacent peers to make the necessary update in their routing tables. In DRTP when a WITHDRAW message is received, the front-end process transmits the withdrawn routes to their corresponding home nodes via unicast. The back-end process at each home node then executes the algorithm shown in table 5.3. All received routes are cleared from Adj-Rib-In, and then each home node recalculates its best route replacement. If the prefix of the withdrawn route is found to be a juncture, then an internal update message (UPD) that includes the replacement best route attributes is sent to all nodes where the route prefix is replicated (line 8). If no replacement route is found then the home node removes completely the route from its disable table, elects an alternative juncture prefix which is a BMP of the removed prefix and sends it with its children to all nodes via the internal (DEL) message where the route prefix is replicated (lines12-14); accordingly, the receiving nodes will update their tables with the new juncture information and remove the old one. Now, if the parent of a removed prefix is a juncture and no alternative route is found in the Adj-Rib-In, then the home node sends a delete message to all replicas of the removed prefix (line 4). Finally, if the withdrawn route prefix is neither a juncture nor a child of a juncture, then the home node will simply remove it as in the normal BGP route withdrawal case. As depicted in Table 5.3, the withdrawal of a prefix p will cause the node to generate a message from the home node of p to the node hosting the replica of p, in two cases: if the prefix p to be deleted is a juncture or a child of a juncture prefix. In all other cases, no message is sent. For every deletion, besides the search in the main trie, DRTP looks in the disable table to find whether the parent of the prefix is a juncture. Upon reception of a message from a home node regarding a deletion, the receiving node takes one of the three following actions:

TABLE 5.3 ALGORITHM TO WITHDRAW A ROUTE IN DRTP

Algorithm: withdraw route

Input: route r **Output**: message sent; RIB updates

```
/*Let MT denotes the Main Trie*/
/*Let DT denotes the Disable Table*/
Prefix p = r.prefix;
1.  Elect a new best route (br) from Adj-Rib-In /*Normal BGP action*/
2.  if (p∈ MT && parent (p) ∈ DT && br=NIL) /*p is a child juncture*/
3.  begin
4.    /*remove  juncture child replicated in subsequent nodes*/
5.    send (DEL, p, NIL) to all nodes where p is duplicated; /*send DEL
                          to all replicas to remove p from their juncture table*/
6.  end
7.  else if (p∈ DT and  br ≠ NIL)
8.  begin
9.    /*just update route attributes of the replicated juncture prefix*/
10.   send (UPD,r.attributes,br.attributes) to all nodes where p is replicated;
11. end
12. else if (p∈ DT && br=NIL)
13. begin
14.     q = BMP(p);   /*elect a new juncture replacement*/
15.     put q in DT;
16.     send (DEL, p, q ) to all nodes where p is duplicated;/* this message
17.                          tells replica to remove p and add q to JT */
18. end
19. delete r;
```

(1) Update the juncture prefix attributes by those of the replacement in UPD message; (2) remove the juncture prefix and insert a second juncture piggybacked in the DEL message; or (3) remove a child prefix of a juncture.

5.4.3 Lookup

The parallel lookup algorithm is quite straightforward. Only, the nodes that belong to a same stride group receive a copy of the multicast message to lookup a prefix. The lookup is done in two stages; the first stage consists of the BGP classical lookup procedure. However, if no match is found, then the next stage lookup will be invoked at the juncture table. Since the size of juncture table is small, we expect much fewer lookups to be

performed in the second stage. Table 5.4 shows the pseudo-code of the lookup message processing by DRTP. The nodes that receive a lookup message execute the pseudo-code in parallel. The function prefix_match(a,b) simply returns true, if prefix b is a match of prefix a.

TABLE 5.4 PARALLEL LOOKUP ALGORITHM.

Algorithm: lookup

Input: prefix p **Output**: best match prefix (BMP)

```
1.  /*lookup in disable table*/
2.  d=BMP(p);
3.  /*lookup for a prefix p in the main trie*/
4.  if  ((q=BMP (p)) && len(d)<len(q)) then
5.      return q;
6.  if (len(d)>len(q))  exit(); //bmp in next higher interval partition
7.  /* fetch the juncture table*/
8.  if  (j=BMP (p)) then      //j is a juncture prefix
9.  begin
10.       if (∃ s∈ S, prefix_match (p, s) = true) then //S:set of j's children
11.            exit (); //bmp in lower interval partition
12.       return j;
13. end;
```

5.5 Complexity evaluation

In this Section, we study the complexity of the proposed scheme. Let us assume that we have partitions that consist of k horizontal nodes where the total routing table size is N and m is the number of junctures. We study the overhead complexity in terms of: (1) The number of internal messages exchanged after an insert or a delete; (2) memory; and (3) time complexity. Finally, we study analytically the achieved performance speedup in DRTP.

5.5.1 Overhead: internal messages

In most cases, for non-juncture prefixes, no messages are exchanged internally to synchronize the partitions. On the other hand, for every juncture prefix received, either for withdrawal or for insertion there is a cost to update the structure. The withdrawal

complexity is O (k); this worst case arises when a prefix home node is the first node with lowest length interval and it is a juncture duplicated in all partitions. In case of insert, the worst case scenario for a juncture is when the prefix is a match in all partitions; in this case a home node will receive k messages of type DUP or 1 message of type JUN. Hence, the number of messages exchanged will be O (k) for DUP messages and one or zero message otherwise.

5.5.2 Overhead: memory

Let us consider a routing table with N prefixes including m junctures, to be partitioned horizontally into k partitions according to DRTP. In order to measure the complexity of our scheme in terms of memory overhead, we need to quantify the size of the replicated junctures. In fact, the upper bound is reached when every juncture in node i is replicated to all nodes j, where i=1,2,3..k-1 and j=i+1, i+2, ..k; without loss of generality, if we assume that each node contains m junctures (children included), then the total memory overhead (i.e., extra memory needed to realize DRTP) in the whole system will be:

$$\sum_{i=1}^{i=k-1} i \times m = \text{m+2m+3m+..(k-1)m} = \text{k(k-1)m/2} = \text{O (m}\times\text{k}^2\text{)} \quad (1)$$

Equation (1) shows that the number of partitions and the number of junctures have a direct impact on the overhead in terms of memory. The average memory size per node will be O (N/k+mk) if we assume an equi-probable distribution of prefixes per partition. With a vertical partition on s bit strides the average memory per node is reduced to O ((N/k+mk)/ 2s).

For better understanding, let us consider an example where a routing table with 1 million prefixes contains 5% of its elements as junctures and 4 horizontal partitions; in the worst case scenario, we will have 4*3*(50000)/2 = 300000 replicated prefixes in the 4 horizontal nodes. Now, let us further consider a vertical partition on a 2-bit stride; this will split the horizontal partitions along with the extra replicated prefixes to 4 additional groups; this results in a total of 16 nodes. Consequently, the nodes within each group will contain 25000 additional prefixes (excluding the first node from each horizontal group). Waldvogel [13] evaluated, in the worst case, the increase of prefixes, when using his approach, to be in

the order O (NlogW) where W is the length of the prefix and N is the initial size of the routing table. If we consider W=32 (IPV4 prefix length), that would generate 5 times (5 millions prefixes) the size of the routing table. Although DRTP (physically distributed nodes) and Waldvogel's approach (same node partitioning) are different, we can conclude that O ($m \times k^2$) is an acceptable increase of the routing table for "reasonable" values of m and k.

Interestingly, the complexity of DRTP (O ($m \times k^2$)) is independent of the prefix length (in opposition to [13]); this makes our scheme very attractive to partition the IPv6 routing table where the prefix length is equal to 128-bit.

5.5.3 Time complexity

The time complexity of a lookup operation in DRTP is the elapsed time between the time a packet is transmitted from the frond-end process to the time a response is returned (see Fig. 5.6). The response time can be expressed as the processing time of the lookup query by the back-end process plus the round trip time (RTT) to deliver the reply from the back-end process. The processing time complexity, is expected to be the same as that for the corresponding operation in the underlying radix-tree data structure which is O(W) [23], where W is the length of the prefix. However, a constant factor speedup is expected because each partition in DRTP has only a fraction of the prefixes of the complete routing table. On the other hand, the RTT complexity is closely related to the underlying multicast protocol being used: ACK or NACK based protocol [24]. In ACK based protocols, the number of horizontal partitions has a direct impact on the response time; when the number of these partitions grows, the number of ACK messages for each query will increase accordingly (ACK implosion); this will impact the communication latency. Fortunately, in BGP multiple prefixes with common attributes can be encapsulated in one NLRI (Network Layer Reachability Information) within the update message; this feature increases the communication granularity and decreases the impact of the communication latency. Thus, the ACK implosion effect will be alleviated but not avoided. In NACK based protocols, the communication latency can be significantly reduced; however, due to using log tables to keep messages history, the space scalability at the sender side can be reduced. In order to

minimize the impact of ACK implosion due to multicast, we have elaborated a novel multicast protocol that generates one ACK per multicast message, regardless of the number of receivers. Due to the limited space, we describe the protocol details in a separate paper.

5.5.4 Analytical parallel performance evaluation

In this section, we study analytically the parallel performance of DRTP. Our aim is to study the performance of DRTP compared to classical prefix range-based partitioning schemes of BGP RIBs.

A typical prefix range-based partitioning scheme [16] does not increase the performance speedup (i.e., does not shorten response time) but only the space scalability. In the following, we present a proof of this statement/finding; let us consider the case where RIB is partitioned into *p* nodes each holding a partition according to certain prefix range criteria. Upon receipt of a query from a front-end process, only one node will process the query and returns a response. Let us assume that there exists a function *F (p)* which gives the fraction of time (probability) over which p is busy (processing a query); in this case, we have:

$$\sum_{i=1}^{p} F(p) = 1 \qquad (2)$$

Therefore, the overall speedup denoted by O is given by:

$$O = \sum_{i=1}^{p} F(p) \times S_i \qquad (3)$$

where S_i is the partial speedup achieved by a partition node. In BGP, most computation time to process an update message consists of the sum of the prefix lookup time and the time to update the route by applying best selection process. In a radix-tree RIB implementation, the lookup complexity is equal to O (W) where W is the prefix length of the route to update. Obviously, this complexity is independent of the tree size; this means that the lookup time in a partition node or in the complete routing table is nearly the same. Similarly, the time taken to update a route by applying the breaking-tie algorithm is the same in both cases. Thus, the overall speedup is:

$$S_i = 1 \text{ and } O = \sum_{i=1}^{p} F(p) \times S_i = \sum_{i=1}^{p} F(p) = 1$$

We conclude that the model presented in [16] does not provide a speedup advantage over the centralized model. In opposition, in DRTP instead of having one working node at a time we have *k* nodes distributed horizontally working collectively to perform parallel search and update of a given multicasted route (query). Let us now compute the DRTP speedup expressed in Equation (3). *F(p)* now is the probability that a given horizontal *group* is busy (processing a query) among *p* groups; each group is composed of *k* nodes. A group speedup S_i representing the i^{th} horizontal group is the fraction of the sequential execution time over the parallel execution time. Recall that, in DRTP, all nodes perform lookup for all prefixes in order to detect junctures but only home nodes perform route update of their own prefixes. In addition, once a juncture prefix is detected in a non-home node, a communication message is sent to the home node. If we assume the worst case scenario when a juncture from the first node is duplicated in the remaining (k-1) nodes, then the speedup of the i^{th} horizontal will be:

$$S_i = \frac{(t_l + t_u) \times n_r}{t_l \times n_r + \frac{t_u \times n_r}{k} + (k-1) \times f_r \times C} \qquad (4)$$

where t_l is the lookup time of a route, t_u is the time spent by a home node selecting best path updating a route, f_r is the fraction of routes whose prefixes are junctures, and C is the communication latency to send a JUN or DUP.

$\left((t_l + t_u) \times n_r\right)$ is equal to the lookup and update time to process n_r routes sequentially at one node. $\left(t_l \times n_r + \frac{t_u \times n_r}{k} + (k-1) \times f_r \times C\right)$ is equal to the lookup and update time to update in parallel n_r routes; it consists of three components: (a) $\left(t_l \times n_r\right)$ is equal to the lookup time, by all nodes in parallel, to detect if the incoming routes contain juncture prefixes; (b) $\left(\frac{t_u \times n_r}{k}\right)$ is equal to the time spent, by only home nodes, to update n_r routes; and (c) $\left((k-1) \times f_r \times C\right)$ is the communication delay to exchange JUN/DUP when juncture prefixes are found. This delay represents the worst case scenario when a juncture in the first node is found replicated in all (k-1) subsequent nodes. Note that, the term in (b) assumes the workload is

uniformly distributed horizontally on the k nodes. DRTP will achieve a speedup if the following inequality holds:

$$\frac{(t_l + t_u) \times n_r}{t_l \times n_r + \frac{t_u \times n_r}{k} + (k-1) \times f_r \times C} > 1 \qquad (5)$$

Where: $f_r = \alpha \times n_r$ and α is the fraction of juncture prefixes.

Thus, inequality (6) can be expressed as follows:

$$\alpha < \frac{t_u}{k \times C} \qquad (6)$$

To achieve a speedup, using DRTP, in any group of *k* partitioned horizontal nodes during a given time interval, the percentage of new juncture prefixes found during this interval of time must be smaller than the fraction of the average time needed to perform an update of a route (t_u) over the communication latency (*C*) to send a message between two nodes divided by the number of partitions *k*. This means that, a considerable performance speedup is realized whenever the update time is significantly high and the amount of juncture prefixes is considerably small.

In current BGP RIBs, the number of alternative routes is evaluated to be around ~40 per destination [14]. This would require a longer time to perform best path selection process and consequently the route update. In addition, the hourly rate of juncture prefixes found empirically during our tests (next section) are far less than 0.01% of the total prefixes carried hourly by the BGP Updates; this guarantees a significant speedup will often be obtained using DRTP.

5.6 Experimental results

In order to evaluate DRTP, we implemented the DRTP algorithms and the multicast operations in the Quagga BGP real router [9] for two horizontal nodes. Our aim is to study both the impact of junctures (key elements in DRTP) and the communication overhead

while processing BGP real traffic update messages. In next section, analytical evaluation is carried out for larger number of nodes.

5.6.1 Juncture prefix distribution

We downloaded several real-life BGP RIBs from [14]. We consider the most recent RIB dated in 2009. The size of the RIB is 288732 of distinct prefixes (9 million with all Adj-Rib-Ins). We ran our program to decompose statically the RIB into four (range-based) vertical partitions and two (length-based) horizontal partitions (H1 [1-23], H2 [24-32]). The goal of the experiment is to show the prefix distribution per partition as well to quantify the number and percentage of junctures in the corresponding partitions. The results are depicted in TABLE 5.5; we observe that, the percentage of juncture prefixes is around ~5-10% of the whole routing table. Similar results were determined using different RIBs [14] tested during our benchmarking evaluation.

TABLE 5.5 2-HORIZONTAL (H1 AND H2) AND 4-VERTICAL BGP RIB PARTITIONS

Prefix range	*H1*	*H2*	***Total***	***Junctures***	***%juncs***
[0,63]	11970	12296	24266	1795	7.4%
[64,127]	54038	43124	97162	9601	9.8%
[128,191]	18688	14408	33096	2912	8.8%
[192,255]	50487	83766	134253	12619	9.4%
Total	135183	153594	288732	26927	9.3%

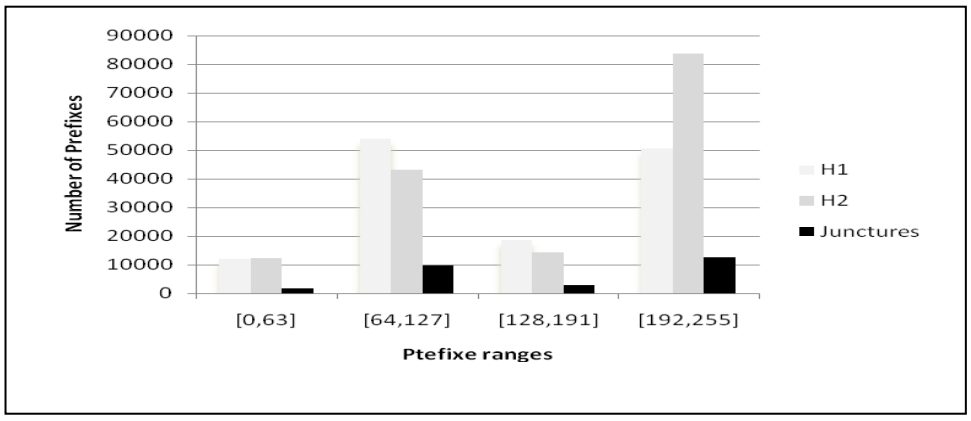


Figure 5.7 BGP RIB (year2009) from route view

Fig. 5.7 shows that prefix length-based horizontal partitioning can yield a nearly-balanced partitions for all ranges (H1 and H2), thus increasing the partitioning level of the routing table. Figure Fig. 5.8 shows the results of (horizontal) prefix length-based partitioning of real data RIBs [14] from 2001 to 2009. We observe two key findings that support our DRTP design choices: (1) prefix length-based partitioning of the RIB is desirable since it produces a nearly-balanced horizontal partitions (H1 and H2) for all tested RIBs under consideration; and (2) the number of junctures and children per juncture prefix is acceptable (i.e., far smaller) with respect to the size of the routing table.

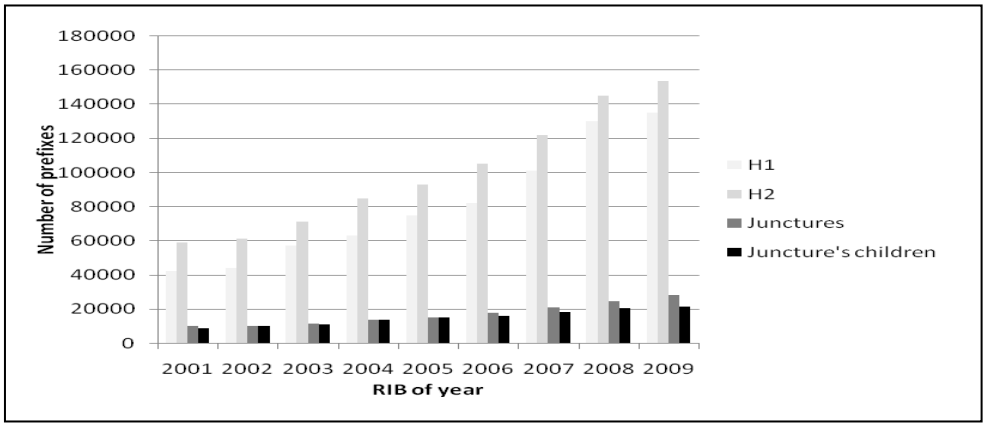


Figure 5.8 Juncture distribution on two horizontal RIB partitions

5.6.2 Incremental update statistics in DRTP

The objective of the current experiment is to empirically (1) the overhead generated by DRTP in terms of internal messages exchanged among partition nodes; and (2) the growth rate of juncture table during a snapshot of three-day BGP update execution. To this end, we downloaded from RouteViews [14] (a) a set of nine RIBs chosen at random dates; and (b) the corresponding update messages applied on these RIBs during the next 3 days. The experiment conducted for each RIB is performed in two steps: (1) the RIB is divided into 2 partitions residing in two routing servers (i.e., 2 nodes) forming a multicast group; each DRTP partition is run on Quagga [9] routing server. The first routing server hosts a horizontal partition managing prefixes with length interval [1-23] and the second routing server is hosting the second partition managing prefix length interval [24-32]. The junctures are replicated and the DRTP code added to the BGP Quagga router is instrumented to extract relevant statistics; and (2) the flow of update messages is sent via multicast to the two servers. We extracted the results (see Table 5.6) by executing the accumulated BGP updates in chunks of one-hour interval time. At the end of the experiment, we computed the hourly average of all collected data. Table VI shows that the number of newly added juncture prefixes (row 1) is small. In addition, the amount of JUN and DUP internal messages that create new junctures is considerably low in comparison to the total number of Updates (row 5) and the total number of prefixes that are updated (rows 6 and 7). More specifically, the number of internal messages to exchange new junctures between the first partition and the next one (the sum of JUN and DUP messages) is about 8 messages per hour; from 50,000 update messages received in one hour, updating nearly 150,000 prefixes, only 8 synchronization messages are exchanged internally representing less than 0.01% of the total updated prefixes. Thus, although BGP receives update messages at high frequencies, most of these updates involve only changes in existing routes attributes and do not add/remove prefixes from the routing table; this leads to an expected low synchronization overhead in DRTP.

Table 5.6 shows also that 670 (for RIB 2009) DEL messages, in average, are exchanged; this means that 5% of the total withdrawn prefixes (approximately 15000 prefixes) are junctures or children of junctures. We observe that, this number (670 messages per hour) is

not proportional to the size of juncture table that contains about 10% of the entries of the total routing table. One possible explanation to this difference is that, junctures are less specific prefixes that are more stable in the Internet and less subject to withdrawal compared to stand-alone and more-specific prefixes.

We also observe from table 5.6 that the amount of updated prefixes in the first partition where prefix length is smaller than 24 is not always proportional to the amount of updated prefixes in the second partition holding prefixes of length bigger than 23. This difference impacts the speedup performance of DRTP; indeed, when the workload distribution on the two horizontal nodes (prefix length< 24 vs. prefix length >23), is quasi-balanced as in the case of RIB 2009, DRTP will generate a significant performance speedup. However, if the workload is not balanced (RIB 2005), performance speedup will suffer from load imbalance because much work will be condensed on one partition leaving the second partition frequently idle; in this case, synchronization overhead will offset the performance gain obtained from parallelism. (See next Section).

TABLE 5.6 BGP STATISTICS WITH DRTP

RIB of year/Hourly average statistics	2001	2002	2003	2004	2005	2006	2007	2008	2009
New Junctures /hr	7	4	7	8	6	4	5	7	3
JUN Messages /hr	5.1	2.18	4.1	5.2	4.75	3.1	3	4	1.5
DUP Messages /hr	1.8	1.81	2.9	2.8	1.25	0.9	2	3	1.72
DEL Messages /hr	217	269	280	267	357	422	580	620	669
ADD Messages/hr	17300	20400	27000	33200	38400	42800	49500	57000	60300
WITHDRAW Messages/hr	1700	1850	2200	2900	4200	4300	4600	4900	53400
Hourly updated prefixes of H1(length <24)	19000	21500	33000	39000	35000	52000	61000	68000	72500
Hourly updated prefixes of H2 (length > 23)	27400	35100	45000	51000	71000	75000	80000	85000	90000

5.6.3 DRTP runtime behaviour

The data collected in table 5.6 shows the global average) results for 3-day BGP execution. The goal of this section is to report results over short time intervals; these results will give a better idea on the runtime behaviour of DRTP. Fig. 5.9 shows the number of messages exchanged internally while processing (using DRTP) the real traffic of BGP updates from 10/26/2001 to 10/27/2001 for the RIB dated on 2001-10-26 downloaded from [14].

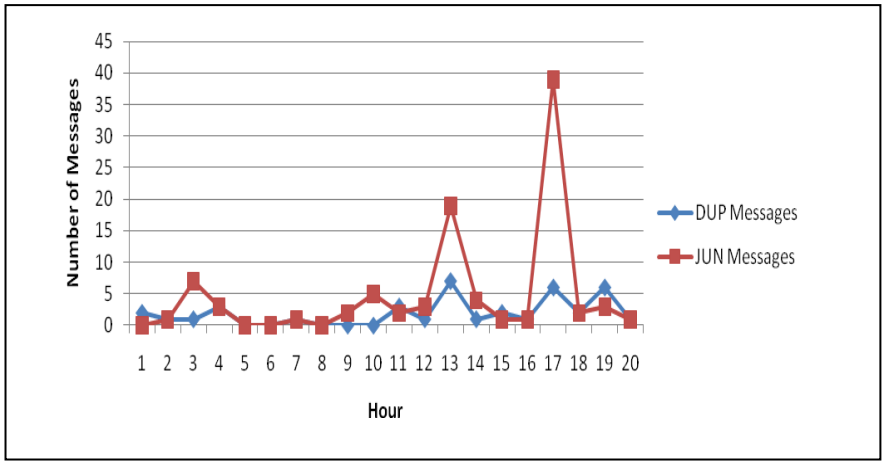


Figure 5.9 Hourly internal messages in DRTP

We started by sending real BGP update messages, accumulated during the execution of BGP in 20-hour length interval, to the two nodes; then, we recorded the number of messages exchanged during each hour.

Fig. 5.9 shows that, the number of messages exchanged internally is negligible compared to the total number of update messages received from peer routers during the same period of time. A peak value (e.g., 39 at 18^{th} hour in Fig. 5.9) of the number of DUP messages, explains a normal phenomena in BGP when a new peer router comes up and generates high-rate BGP traffic (i.e., Update messages) before it stabilizes and the traffic goes back to its normal level. Peak values can also appear because of link oscillations.

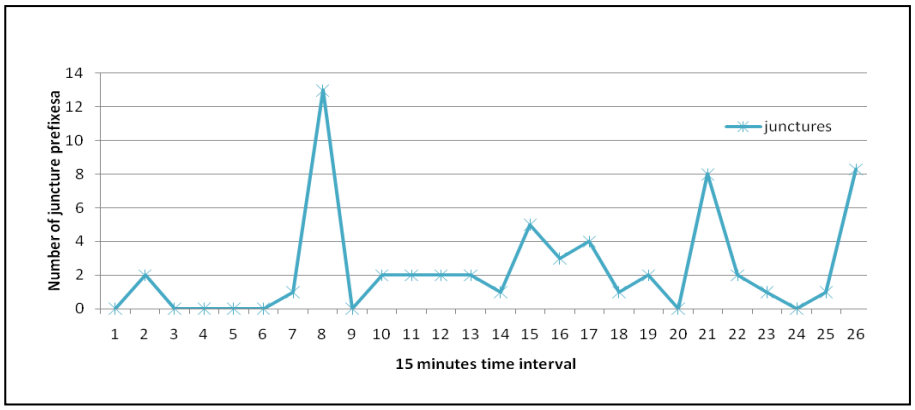


Figure 5.10 Juncture table size growth during 6 hours interval in DRTP

Fig. 5.10 shows the growth trend of the juncture table during a snapshot of 6-hour execution of BGP Updates for two-node partitions of RIB. The number of new junctures is recorded every 15 minutes time interval. We observe that the number of newly found juncture prefixes, which is correlated with the number of internal messages exchanged, is negligible with respect to the global BGP workload volume; indeed, the processing capacity in DRTP is mostly used to perform parallel computation rather than processing new junctures. Fig. 5.11 shows the number of deleted juncture prefixes during a 20-hour interval. We observe that the peak value of the number of deleted junctures, causing the generation of DEL messages, is around 2000 messages per hour or 30 messages per minute. In order to reduce the number of DEL messages, a technique to encapsulate a number of deleted prefixes, over a predefined time period, and then send a single message that contains all these prefixes can be used. This will definitely decrease the overhead introduced by DEL messages.

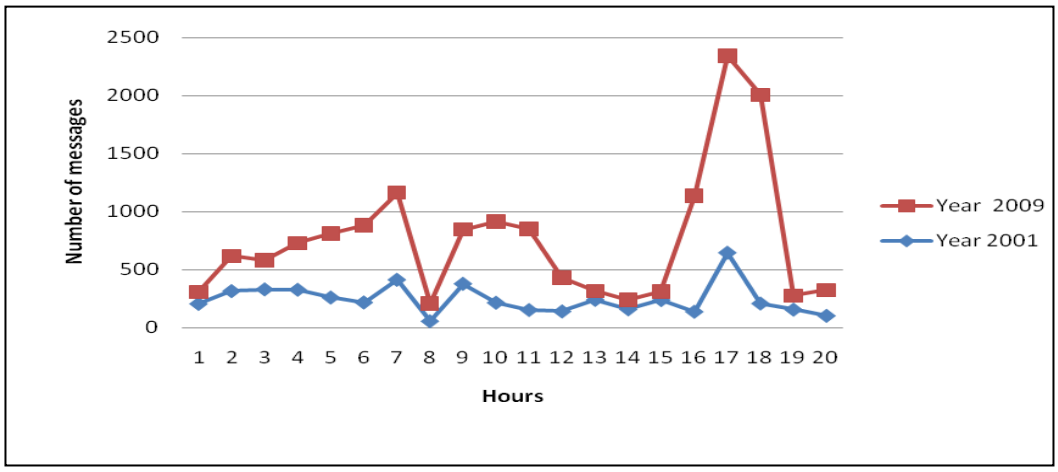


Figure 5.11 Number of internal DEL message during 6-hour BGP data divided into 15 minute execution time interval in DRTP

The peak value (1800) of Y-axis in Fig. 5.11 reflects an event that happens in everyday BGP normal execution, where a failure has occurred in some adjacent peer router and forced BGP to remove a chunk of routes learned from the failed router.

5.6.4 BGP table updates performance

In this section, our aim is to measure the performance speedup of DRTP using two horizontal-length partitions of the routing table where each partition is running on a router instance. The evaluation has been conducted on three Linux machines with 2.6GHz CPU connected by a 300Mbps router with enabled multicast capability. In the centralized case (classical BGP operation), the front-end application sends the update messages (MRT format [5]) to the BGP router via simple UDP unicast with retransmission; with DRTP, the front-end application multicasts the update messages to the group of two partitions. We developed our own reliable protocol that requires one ACK (Acknowledge) per multicast message. We run our experiments using RIBs downloaded from the archive of RouteViews [14] dated on 2001, 2005 and 2009. For each RIB, the front-end injects into the routers a set of updates, collected also by RouteViews, of the corresponding RIB during the next 9 hours (starting from the RIB date). Table 5.7 shows that the number of update messages as

well the number of updated prefixes for the 3 RIBs under consideration. Due to the irregularity of prefix distribution in the three samples, different speedup levels were achieved. We found that in the first and second sets (2001, 2005) there is a moderate speedup offered by DRTP while in the third set (2009) DRTP outperforms the centralized approach by up to 20% (see Fig. 5.12).

TABLE 5.7 CENTRALIZED APPROACH (CE) VS. DRTP

	# Prefix updated		***# of UPDATES***	***Time in sec.***	
Year	*[0-23]*	*[24-32]*	***ADD***	***CE.***	***DRTP***
2001	139100	302900	117400	140	129
2005	106024	460976	345300	416	407
2009	435049	539951	334300	598	481

This difference in the speedup levels can be explained as follows: (1) (1) in the third set, the updated prefixes are better balanced among the two horizontal partitions while in the first two sets the updated prefixes are more condensed in the second partition (length > 23); and (2) the number of prefixes updated in the third set (RIB 2009) is much bigger than the number of prefixes updated for the second set (RIB 2005); however, the amount of BGP messages sent by the front-end process is almost the same in both cases. This means that update messages of the third set encapsulate more prefixes in their NLRI than update messages of the second set. This increase in prefix encapsulation per message (RIB 2009) causes an increase in the computation granularity at both partitions. Thus, the computation to communication ratio becomes higher; indeed, both partitions of the third set will spend most of the time executing BGP and less time receiving messages and sending ACKs to the front-end process (RIB 2005); this clearly explains the speedup achieved in RIB 2009. Interestingly, the number of new junctures found during the experiment is low (6-8 junctures/hour); this means that the communication overhead caused by the amount of internal messages exchanged in DRTP, is minimal compared to the BGP traffic volume.

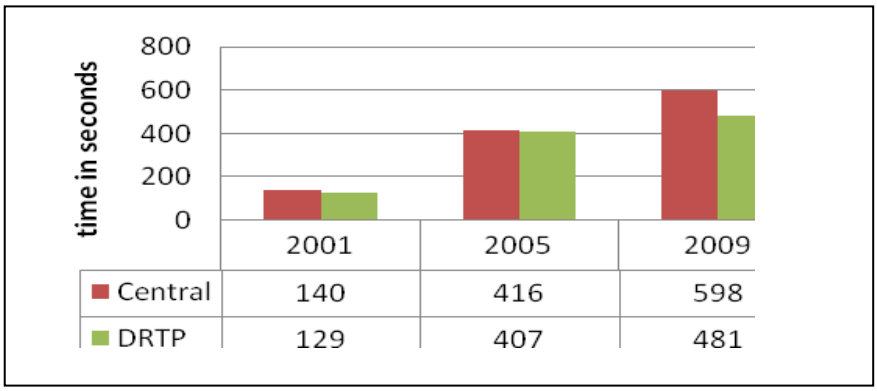


Figure 5.12 Central approach vs. DRTP

We also observed that for extremely irregular distributed prefixes traffic and unbalanced workload per partitions (which is not the case of real life RIBs), the centralized approach might be very slightly faster than DRTP.

5.6.5 DRTP analytical performance

According to Equation (6), the key components that determine whether a speedup can be realized, in a given interval of time, are: (1) the amount of route updates and (2) the percentage of junctures found. To get an approximation about the RIB update processing time t_u of a single prefix, we conducted an experiment that consists of applying, through two successive trials, the same set of BGP updates on the same RIB. In the first trial, we measure the processing time taken to insert all routes that initially did not exist in the RIB. In that case, the processing time is the sum of the lookup time and update time to insert all prefixes. In the second trial, the routes were already inserted in the RIB; thus the measured processing time will count only for the lookup time since no update processing is involved. The difference of the two measurements evaluates the update processing time of the entire set of prefixes carried by the BGP updates. Dividing the difference of the two values by the total number of prefixes being updated, an approximation of the prefix update time t_u will be obtained. We conducted this experiment on several RIBs from RouteViews [14] along with several samples of updates from different time periods. The evaluated prefix update time t_u is varied from (50µs to 100µs). On the other hand, the prefix lookup time measured

in a radix-tree RIB data structure is known to be approximately 2.580µs [26]. In our model, we assume the communication latency C to exchange and process a message between two horizontal nodes is in the order of few milliseconds. Based on these findings, Fig. 5.13 (two horizontal partitions) and Fig. 5.14 (three horizontal partitions) show the analytical speedup achieved for different percentages of junctures (α) and three values of the update time t_u as a multiple of the lookup time t_l. The communication latency C is set to be 100 times the update time in all three cases. From Equation (4) we get the speedup curves shown in Fig. 5.13 and Fig. 5.14 for different percentages of junctures (α) and three values of the update time t_u (i.e., 10* t_l, 20* t_l, and 30* t_l);

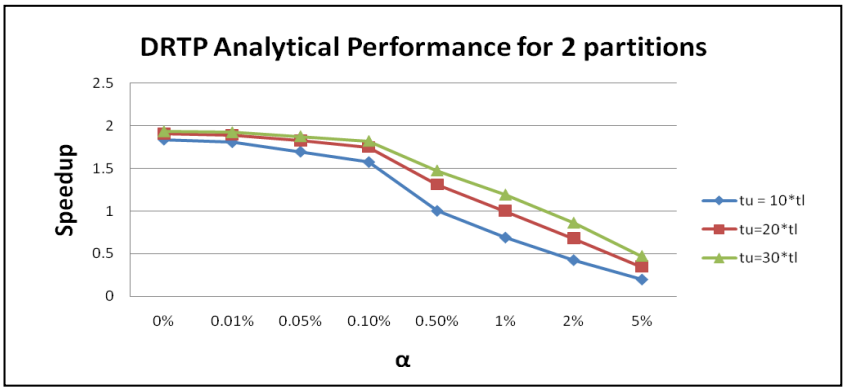


Figure 5.13 DRTP analytical performance in 2 horizontal partitions for different values of α and 3 update time values

Fig. 5.13 and Fig. 5.14 show that a considerable speedup can be achieved when α (rate of the juncture prefixes) is smaller than 0.1 % of the total updated prefixes. When α increases, the speedup deteriorates gradually and starts to become negative when α reaches 0.5%.

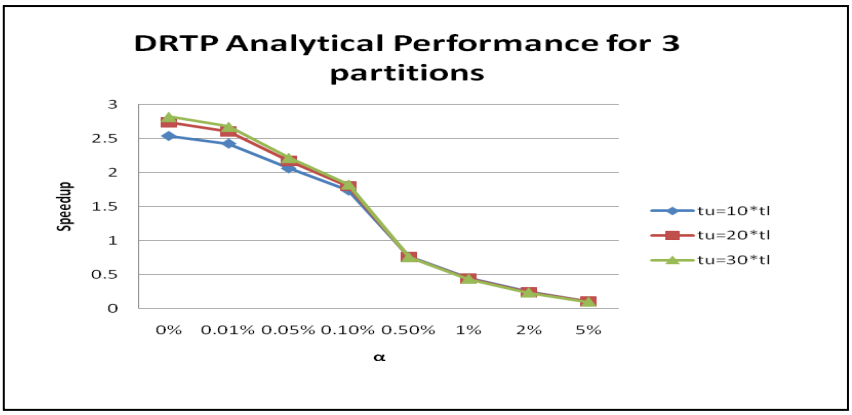


Figure 5.14 DRTP analytical performance in 3 horizontal partitions for different values of α with 3 update time values

Note that, besides the juncture percentage α, the update time, in terms of the lookup time, has an impact on the overall speedup. For example, the first scenario, where $t_u = 10 * t_l$, produces the lowest speedup. This can be explained by the fact that, in DRTP, home nodes perform route updates for their own prefixes and perform lookup for all incoming prefixes in other nodes; thus, the route update which is the computation part of processing a BGP Update, is run in parallel in DRTP while route lookup which represents the BGP sequential part of processing the BGP Update that cannot be parallelized. According to Amdahl's law [25], if the ratio of parallel/sequential parts of the code is high, better performance speedup can be achieved for an increasingly number of nodes. According to this rule, the third scenario where ($t_u = 30* t_l$) will achieve the highest speedup since it generates the highest parallel/sequential computation ratio among the three considered scenarios. It is worth noting that t_u / t_l impacts the speedup when DRTP's internal communication latency is low. However, when it is not the case (the value of α is big) the impact of t_u in achieving speedup is considerably reduced. Practically, it may frequently happen that during a given interval of time, the majority of the BGP messages update only existing routes without adding or deleting juncture prefixes. This feature of the BGP updates can be interpreted in our analytical model by setting α to zero in (see Equation (5)). Therefore, the performance speedup for different update/lookup time scenarios will be shown in Fig. 5.15. Obviously,

when α equals to zero, a good speedup can be achieved in the three considered update/lookup time scenarios. As expected, the best speedup is attributed to the curve of the highest route update time (i.e. $t_u = 30 * t_l$).

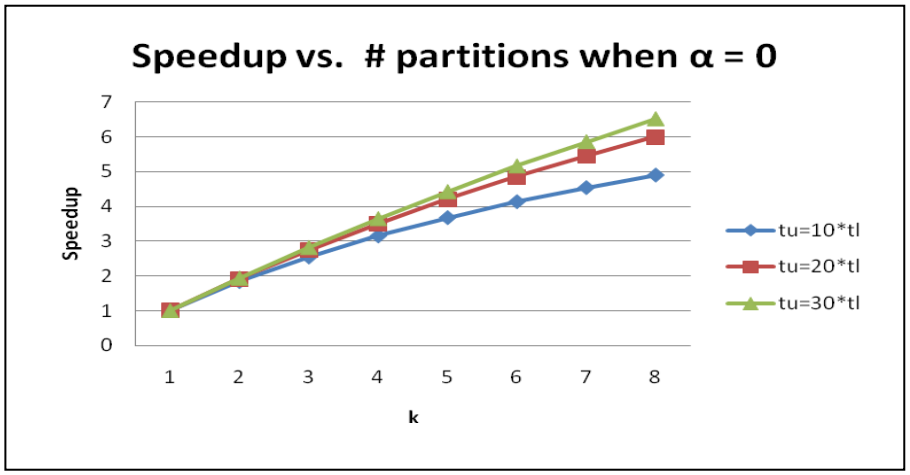


Figure 5.15 DRTP Speedup vs. number of horizontal partitions when no juncture prefixes

As shown in Fig. 5.15, a significant speedup can be achieved during a given interval of time if the amount of juncture prefixes tends to zero and the route update time is significantly high.

5.7 Conclusion

There is a serious growing concern in the scientific and industrial communities about the growth rapidity of routing tables. Existing techniques relying on periodical upgrade of core routers, to cope with the overwhelming traffic evolution, may no longer be a viable solution in the long term; besides, it is a costly process. In this paper, we proposed a novel parallel-distributed scheme, called DRTP, for partitioning the BGP RIB at control plane inside the router without modifying the core of BGP. DRTP is a method to partition the BGP routing table in two dimensions according to prefix length and prefix range in order to increase both space scalability and performance of BGP. The algorithms, we developed to realize DRTP, show that the modifications needed to be applied on BGP are simple. In addition, the lightweight implementation of DRTP can be done incrementally without

impacting the external visible behaviour of BGP. Our experiments and analytical study show the role of DRTP in improving performance while increasing the size/space scalability of the BGP routing table. Moreover, our experiments using real life data confirm the validity of our approach at a reasonable accurate level. We believe that DRTP will enhance the scalability of routing tables, increasing the average lifetime of core routers and consequently sustaining the whole Internet infrastructure.

5.8 References

[1] Fuller, V., et al., "Classless inter-domain routing (CIDR): an address assignment and aggregation strategy", RFC1519, 1993.

[2] Akhbarizadeh, M.J., M.,Nourani. "Hardware-based IP routing using partitioned lookup table." *IEEE/ACM Transactions on networking* vol.13, no. 4 (2005): 769-781.

[3] Tzeng, N.F. "Routing Table Partitioning for speedy packet lookups in scalable routers." *IEEE Transactions on Parallel and Distributed Systems* vol. 17, no. 5 (2006): 481-494.

[4] Hamzeh W., A., Hafid. "A scalable cluster distributed BGP architecture for next generation routers." *IEEE LCN.* Zurich,Switzerland, 2009. 161-168.

[5] MRTD: The Multi-Threaded Routing Toolkit. http://www.mrtd.net.

[6] Hagsand, O., M., Hidell, P., Sjodin. "Design and Implmentation of a Distributed Router." *IEEE Int'l Symposium on Signal Processing.* Athens, Greece, 2005. 227-232.

[7] Ruiz-Sanchez, M., E.W.A., Bierscak, W., Dabbous. "Survey and taxonomy of IP address lookup algorithms." *IEEE Network* vol. 15, no. 4 (2001): 8-23.

[8] Srinivasan, V., Varghese, G. "Fast address lookups using controlled prefix expansion." *ACM Transaction on Computer Systems* vol. 17, no. 1 (1999): 1-40.

[9] The open source quagga project, http://www.quagga.net

[10] *"Next generation networks and the cisco carrier routing system".* 2004. http://www.newsroom.cisco.com/dlls/2004/networks_and_the_cisco_carrier_routing_system_overview.pdf.

[11] *"Building a Higly-Available Enterprise Networks with Juniper Networks"*, 2010. http://www.juniper.net/us/en/local/pdf/whitepapers/2000257-en.pdf; Juniper Inc.

[12] Rekhter, Y. "A Border Gateway Protocol 4 (BGP-4) ", RFC 4271, 2006.

[13] Waldvogel, M. "Scalable High-Speed Prefix Matching." *ACM Transactions on Computer Systems* vol.19, no. 4 (2001): 440-482.

[14] "ftp.routeviews.org/bgpdata", University of Oregon Route-Views Project.

[15] Sun, Q., H., Xiao, Y., Zhou. "A Dynamic Binary Hash Scheme for IPV6Lookup." *IEEE GLOBECOM.* 2008. 1-5.

[16] Scudder A. "Partitioned Routing Information Base". Ed. US patentstorm. Patent 7499459. 2009.

[17] Bu, T., Gaou, D., Taously. "On characterizing BGP routing table growth." *The International Journal of Computer Networks* 45 (2004): 45-54.

[18] Labovitz, C., R., Malan, F., Jahanian. "Internet routing stability." *IEEE/ACM Transactions on Networking* vol.6, no. 5 (1998): 515-528.

[19] Lei, G., L., Mingche, G., Zhenghu. "An Improved Parallel Access Technology on Routing Table for Threaded BGP." *International Conference on Parallel and Distributed Systems.* Guangdong, China, 2009. 198-204.

[20] Yang, L., R., Dantu, T., Anderson, R. Gopal."Forwarding and Control Element Separation", RFC 3746, 2004.

[21] Mollar, L., M., Kauer, A., Adameicky, J., Sinsky, L.,Buhl. "Multicast Optical Switch Fabric with Ultrafast Channel Access Based on Heterodyne Receivers." *IEEE Photonics Technology Letters*, vol. 16, no. 1 (2004):272-274.

[22] Ballani, H., P., Francis, C., Tuan, J., Wang. *"Virtual Aggregation: Making Routers Last Longer with ViAggre".* NSDI, USENIX, 2009.

[23] Sklower, K. "A tree-based routing table for Berkeley Unix." *USENIX Conference* . Berkeley, CA, USA, 1991. 93-99.

[24] Pragsyanmita, P., S.V., Ragavan. "Survey of multicast routing algorithms and protocols." *the 15th International Conference on Computer Communication.* Mumbai, India, 2002. 902-926.

[25] Hennessy & Patterson: Livre. "*Computer Architecture A Quantitative Approach. Third edition".* 2003.

[26] Lampson, B., V., Sirnivisan, G., Varghese. "IP lookups Using Multiway and Multicolumn search." *IEEE/ACM Transactions on Networking* vol.7 (1999): 324-334.

CHAPITRE VI

6 CONCLUSIONS ET TRAVAUX FUTURS

Avec les derniers avancements technologiques des réseaux d'interconnexion, la capacité du traitement des routeurs risque de devenir le goulot d'étranglement qui pourrait nuire à l'évolution de l'Internet dans la prochaine décennie. En effet, la capacité de communication des réseaux est en croissance continue. Grâce à ce développement, les routeurs doivent traiter plusieurs millions de paquets par seconde au niveau de la couche d'expédition. Heureusement, avec une amélioration de communication au niveau du « Switch Fabric », la réalisation matérielle du routeur d'une manière physiquement extensible (multi-chassis) vient alléger le problème. Ceci promet que la capacité d'expédition va tenir la route durant la prochaine décennie. Malgré le fait que le trafic au niveau de la couche d'expédition est beaucoup plus critique que celui de la couche de contrôle, ce dernier peut facilement devenir un sérieux défi dans le futur proche. En effet, le protocole BGP (Border Gateway Protocol) qui consomme le plus de ressources au sein d'un routeur s'exécute d'une façon monolithique avec d'autres protocoles sur la même carte de contrôle. Or, la performance, la scalabilité et la fiabilité de ce protocole ont un impact direct sur le réseau de l'Internet tout entier. Cette déficience provient de plusieurs facteurs; notamment la grosseur de la table de routage, la limitation des ressources dans une carte de contrôle pour supporter un grand nombre de connexions TCP et la croissance du trafic. De même, la façon dont le protocole BGP est conçu comme un protocole vecteur distance (Distance Vector Protocol), il peut engendrer des coûts supplémentaires tels que l'oscillation des routes. Tous ces facteurs nous ont incités à penser à restructurer l'architecture logicielle de BGP d'une manière distribuée au sein des routeurs. L'objectif initial était de concevoir et d'implanter une architecture qui enlève la centralisation de BGP au niveau de la carte de contrôle afin de maximiser sa performance, sa scalabilité et sa fiabilité sans modifier la sémantique du protocole. Cette thèse comporte trois contributions qui forment ensemble une suite logique pour atteindre

l'objectif envisagé. La première contribution (chapitre 3) propose une distribution de BGP au niveau des cartes de contrôle en se basant sur la décomposition des fonctionnalités et sur la duplication du RIB (Routing Information Base). Le but est d'augmenter la scalabilité ainsi que la capacité du traitement de BGP sans modifier sa sémantique telle que décrite dans le RFC de l'IETF [Rekhter06]. Cette distribution enlève toute sorte de centralisation en permettant à BGP de s'exécuter en parallèle sur plusieurs cartes de contrôle. De même, cette distribution produit une architecture en grappe (cluster) composée de plusieurs cartes de contrôle formant une seule entité logique par rapport aux routeurs BGP voisins. Les simulations effectuées sur un réseau local ont démontré un gain remarquable au niveau de la performance ainsi qu'au niveau de la scalabilité. De plus, le surcoût de communication ajouté pour effectuer la synchronisation de la table de routage (RIB) dupliquée est acceptable. Or, avec cette distribution, on a pu augmenter la scalabilité de BGP en termes de nombre de connexions TCP supportées et de la performance via l'exploitation du parallélisme. De même, nous avons proposé un algorithme pour assurer la consistance des données au niveau du RIB dupliqué. Cette même architecture s'avère une solution adéquate pour augmenter la fiabilité de BGP. Dans la deuxième contribution, nous avons exploité cet aspect en proposant un mécanisme de tolérance aux pannes pour l'architecture développée au chapitre 3. Ce mécanisme tire profit de la méthode de réplication de données fournie dans la première contribution pour produire une architecture performante, scalable et tolérante aux pannes. L'architecture résultante a pu augmenter la fiabilité de BGP et enlever le point individuel des pannes. Dans ce même contexte, lorsqu'une panne intervient dans une carte de contrôle durant la phase de l'exécution, celle-ci n'empêchera pas l'exécution de BGP sur les autres cartes. Grâce à la réplication active, les autres cartes seront à la fois disponibles pour effectuer le traitement et en même temps pour masquer une erreur qui peut se produire sur une carte de contrôle. Ceci augmentera la résilience de l'architecture distribuée de BGP sans affecter sa scalabilité. Nous avons prouvé analytiquement en se basant sur la métrique MTTF (Mean Time To Failure), qu'avec cette architecture contenant quatre cartes de contrôle pour la réplication du RIB, la disponibilité de BGP, vue par les routeurs voisins est quasiment doublée comparativement à une architecture

centralisée. Donc, l'architecture distribuée résultante présentée dans la deuxième contribution améliore non seulement la scalabilité et la performance de BGP, mais aussi augmente considérablement son niveau de fiabilité. Dans la troisième et dernière contribution, nous avons abordé un thème critique dans BGP qui est celui du partitionnement de la table de routage sur des nœuds physiquement distribués. Pour atteindre ce but, nous avons conçu et implanté une méthode de partitionnement parallèle et distribuée de la table de routage de BGP au niveau de la couche de contrôle. Cette méthode que nous avons appelée DRTP (Distributed Routing Table Partitioning) s'applique sur une table de routage implémentée avec la structure de donnée « TRIE » (radix-tree). DRTP effectue une division de la table de routage horizontalement selon la longueur du préfixe et une autre verticalement selon son rang. Plus précisément, notre contribution scientifique réside dans la manière de partitionnement horizontale. La difficulté rencontrée pour faire la division horizontale est causée par la nature du chevauchement qui existe dans les préfixes. Cette caractéristique essentielle pour effectuer la recherche (Best Match Prefix) empêche de faire la division de la table de routage, selon la longueur du préfixe, en plusieurs partitions disjointes. Pour surmonter le problème de chevauchement, nous avons identifié un ensemble de préfixes inclus dans la table de routage que nous avons appelé JOINTURES (en anglais, « JUNCTURES»). Ces derniers forment les points d'articulation entre les partitions distribuées de la table de routage. Notre technique de partitionnement est basée sur la duplication adéquate de ces préfixes jointures « JUNCTURES » en conjonction avec l'utilisation de multicast des requêtes sur l'ensemble des nœuds physiquement distribués. Grâce à cette technique, les partitions de la table de routage sur des nœuds physiquement distribuées réagissent quant à la recherche (Best Match Prefix) de la même manière comme si elles étaient disjointes. En recevant une requête via multicast, tous les nœuds effectuent la recherche en parallèle et grâce à la duplication des jointures, une seule réponse sera retournée à l'émetteur de la requête. Nous avons proposé des algorithmes parallèles pour faire de la recherche, la mise à jour et la suppression des routes en BGP sur les nœuds distribués contenant les partitions de la table de routage. Les tests benchmarks appliqués sur de vraies tables de routage ont démontré que dans plusieurs cas, DRTP peut fournir une amélioration de la

performance qui pourrait atteindre 20 % comparativement à une approche centralisée. De même, nous avons développé un modèle analytique pour évaluer le gain de performance parallèle en DRTP; le modèle a révélé qu'un gain de performance sera réalisé si le nombre de préfixes jointures créés en cours d'exécution est infime. Heureusement, nos tests sur de vraies tables de routage divisés par DRTP en deux partitions, ont prouvés empiriquement que les préfixes jointures ne constituent qu'un pourcentage ne dépassant pas 10 % de la taille globale de la table de routage. De plus, le taux de détection de nouveaux préfixes jointures ne dépasse pas 0.1 % de la totalité du trafic reçu par BGP. Ceci minimise le coût de la synchronisation au profit du parallélisme exploité et augmente la scalabilité de la table de routage en terme de la taille.

En conclusion, les trois contributions sont encourageantes pour offrir une architecture pour le protocole de BGP hautement parallèle, performante, scalable, et tolérante aux pannes. Nos contributions ouvrent les portes vers d'autres projets futurs. En fait, les deux premières nous incitent à investiguer dans la distribution des autres protocoles ISIS et OSPF dans la couche de contrôle. La troisième contribution nous permet de croire qu'une réalisation de DRTP sur la couche d'expédition est envisageable. Ceci est faisable en appliquant l'algorithme de recherche parallèle de BMP sur les cartes de ligne tel que décrit par DRTP.

La raison pour laquelle nous croyons que DRTP est applicable dans la couche d'expédition est que la latence du multicast au niveau du « switch fabric » est assez basse dans l'ordre de quelques nanosecondes. Or, dans DRTP, chaque carte de ligne contiendra une portion de la table de routage au niveau du FIB de la même manière expliquée pour la couche de contrôle. Lorsqu'un paquet à expédier est reçu par une carte de ligne, cette dernière prendra la bonne décision d'expédition. Ceci dit, si l'adresse d'expédition existe dans la partie courante de la table dans la carte de ligne, le paquet sera expédié tel que dans le cas normal. Dans le cas contraire, le paquet sera dirigé via multicast vers le groupe des cartes de ligne qui peuvent traiter en parallèle la classification et l'expédition du paquet.

Bibliographie

[Abley05] Abley J., K., Lindqvist, E., Davies, B., Black, V., Gill. "IPv4 Multihoming practices and limitations", RFC4116, 2005.

[Ahmad93] Ahmad M., J..E., Burns, P. Hutto, G., Neiger. "Causal Memory: Definitions, Implementation, and Programming ." *Technical Report GIT-ICS-93/95, Georgia Institute of Technology, Georgia, Atlanta*.

[Amdahl] Hennessy & Patterson: Livre . "*Computer Architecture A Quantitative Approach, third edition"*. 2003.

[Agrawal07] Agrawal A., K.V., Garg "Efficient dependency tracking for relevant events in concurrent systems." *Journal of Distributed Computing* vol.19 (2007): 163-183.

[Akhbari05] Akhbarizadeh, M.J., M.,Nourani. "Hardware-based IP routing using partitioned lookup table." *IEEE/ACM Transactions on networking* vol.13, no. 4 (2005): 769-781.

[APNIC] APNIC. *BGP Routing Table Analysis.* http://thyme.apnic.net/current/.

[Baker95] Baker, F., "Requirements for IP Version 4 routers". RFC 1812, 1995.

[Ballani09] Ballani, H., P., Francis, C., Tuan, J., Wang. *"Virtual Aggregation: Making Routers Last Longer with ViAggre"*. NSDI, USENIX, 2009.

[NISCC] "Best Practices Guidelines: Border Gateway Protocol." NISCC (U.K. Governement), 2004, Note 5, p.8.

[Birman91] Birman, K., A., Shiper, P., Stephenson. "Lightweight Causal and Atomic Group Multicast." *ACM Transactions on Computer Systems* vol.9 (1991): 272-314.

[Bornhaus10] Bornhauser, U., P., Martini, M., Horneffer. "Root Causes for iBGP Routing Anomalies." *IEEE LCN*. Denver, Colorado, USA, 2010. 488-495.

[Bonn07] Bonaventure, O., C., Filsfils, P., Francois. "Achieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures." *IEEE/ACM TRANSACTIONS ON NETWORKING* 15, no. 5 (2007): 1123-1135.

[Bu04] Bu, T., Gaou, D., Taously. "On characterizing BGP routing table growth." *The International Journal of Computer Networks* 45 (2004): 45-54.

[Cavalli06] Cavalli, A., D., Vierra. "Working around BGP: An improvment of BGP session maintenance." *IEEE International Conference on Networking and Services.* Silicon valley, Calif., USA, 2006. 41-49.

[Cisco] *"Next generation networks and the cisco carrier routing system"*. 2004. http://www.newsroom.cisco.com/dlls/2004/networks_and_the_cisco_carrier_routing_system_overview.pdf.

[Fidge88] Fidge, C. "Timestamps in message-passing systems that preserves partial ordering." *Australian computer science communications* vol.10 (1988): 56-66.

[Fuller 93] Fuller, V. *"Classless interdomain routing (CIDR): an address assignment and aggregation strategy"*. RFC1519, 1993.

[Govindan97] Govindan, R., Reddy, A. "An analysis of Internet inter-domain topology and route stability." *IEEE INFOCOM.* Kobe, Japan, 1997. 850-857.

[Hagsand 05] Hagsand, O., M., Hidell, P., Sjodin. "Design and Implmentation of a Distributed Router." *IEEE Int'l Symposium on Signal Processing.* Athens, Greece, 2005. 227-232.

[Halabi] Halabi, S. *"Pluris massively parallel routing"*. White papers, http://www.academ.com/nanog/feb1998/parallel/index.html; Pluris Inc., 1998.

[Hamzeh09] Hamzeh W., A., Hafid. "A scalable cluster distributed BGP architecture for next generation routers." *IEEE LCN.* Zurich,Switzerland, 2009. 161-168.

[Hamzeh10] Hamzeh W., A., Hafid. "A distributed parallel approach for BGP routing table partitioning in next generation routers." *IEEE LCN.* Denver, Colorado, USA 2010. 480-487.

[IPINF02] *"A redundant architecture for routing protocols"*. IP Infusion white paper. 2002. http://www.ipinfusion.com.

[ISIS] *ISIS Project"*.http://www.cs.cornell.edu/Info/Projects/ISIS/ISISpapers.html

[Juniper] *"Building a Higly-Available Enterprise Networks with Juniper Networks"*, 2010. http://www.juniper.net/us/en/local/pdf/whitepapers/2000257-en.pdf; Juniper Networks Inc.

[Kaplan02] Kaplan, H. *"Part 3, in the Reliability Series, NSR, Non-Stop Routing Technology".* http://63.111.106.66/technology/whitepapers/reliability_series/NSRTechnology.pdf; Avici, 2002.

[Kesl03] Keslassy., I., I.et al. "Scaling Internet routers using optics." *ACM SIGCOMM* (ACM Press), 2003: 189-200.

[Kun06] Kun, W., W., Jianping, X., Ke. "A Tree-Based Distributed Model for BGP Route Processing." *High Performance Computing and Communications.* 2006. 119-128.

[Kushman07] Kushman, N., S., Kandula, D., Katabi, B. M., Maggs. "RBGP: Staying connected in a connected world." *NSDI.* 2007. 341-354.

[Labovitz 98] Labovitz, C., R., Malan, F., Jahanian. "Internet routing stability." *IEEE/ACM Transactions on Networking* vol.6, no. 5 (1998): 515-528.

[Labovitz99] Labovitz, C., A., Ahuja, F., Jahanian. "Eperimental study of Internet stability and wide-area network failures." *International Symposium on Fault-Tolerant Computing.* 1999.

[Lamport79] Lamport, L. "How to make a multiprocessor computer that correctly executes multiprocess programs." *IEEE Transactions on Computers* vol.28 (1979): 690-691.

[Lampson99] Lampson, B., V., Sirnivisan, G., Varghese. "IP lookups Using Multiway and Multicolumn search." *IEEE/ACM Transactions on Networking* vol.7 (1999): 324-334.

[Lei09] Lei, G., L., Mingche, G., Zhenghu. "An Improved Parallel Access Technology on Routing Table for Threaded BGP." *International Conference on Parallel and Distributed Systems.* Guangdong, China, 2009. 198-204.

[Mollar04] Mollar, L., M., Kauer, A., Adameicky, J., Sinsky, L., Buhl. "Multicast Optical Switch Fabric with Ultrafast Channel Access Based on Heterodyne Receivers." *IEEE Photonics Technology Letters*, vol. 16, no. 1 (2004):272-274.

[MRTD] *MRTD: The Multi-Threaded Routing Toolkit.* http://www.mrtd.net.

[Pr02] Pragsyanmita, P., S.V., Ragavan. "Survey of multicast routing algorithms and protocols." *the 15th International Conference on Computer Communication.* Mumbai, India, 2002. 902-926.

[Qiang07] Qiang, W., Y., Liao, T., Wolf, L.,Gao. "Benchmarking BGP routers." *IEEE International Symposium on Workload Characterization.* Boston, MA, USA: IISWC, 2007.79-88.

[Quagga] *The open source quagga project.* http://www.quagga.net.

[Rekhter06] Rekhter, Y. "A Border Gateway Protocol 4 (BGP-4) ", RFC 4271, 2006.

[RouteViews] RouteViews Project. http://www.routviews.org/bgpdata.

[Sanchez01] Ruiz-Sanchez, M., E.W.A., Bierscak, W., Dabbous. "Survey and taxonomy of IP address lookup algorithms." *IEEE Network* vol. 15, no. 4 (2001): 8-23.

[Sangli07] Sangli, S., E. Chen, R. Fernando, J. Scudder, Y. Rekhter. "Graceful Restart Mechanism for BGP", RFC 4724, 2007.

[Saltzer84] Saltzer, J. "End to End arguments in system design." *ACM Transactions*, 1984: 277-288.

[Scudder09] Scudder J., D., Ward, P.Jensen. "Partitioned Routing Information Base". Ed. US patentstorm. Patent 7499459. 2009.

[See08] See-Woong, M., C., Bung-gu, P., Yong-Seok. "Apparatus for distributive processing BGP". Patent 7330474. 2008.

[Sklower91] Sklower, K. "A tree-based routing table for Berkeley Unix." *USENIX Conference* . Berkeley, CA, USA, 1991. 93-99.

[Srinivas99] Srinivasan, V., Varghese, G. "Fast address lookups using controlled prefix expansion." *ACM Transaction on Computer Systems* vol. 17, no. 1 (1999): 1-40.

[Sun08] Sun, Q., H., Xiao, Y., Zhou. "A Dynamic Binary Hash Scheme for IPV6Lookup." *IEEE GLOBECOM.* 2008. 1-5.

[Timothy99] Timothy, G., G., Wilfong. "An analysis of BGP convergence properties." *ACM SIGCOMM.* Cambridge, Massachusetts, USA, 1999. 277-288.

[Tzeng06] Tzeng, N.F. "Routing Table Partitioning for speedy packet lookups in scalable routers." *IEEE Transactions on Parallel and Distributed Systems* vol. 17, no. 5 (2006): 481-494.

[Villam98] Villamizar,C., Chandra, R., Govindan, R."BGP Route Flap Damping", RFC 2439, 1998.

[Waldvog01] Waldvogel, M. "Scalable High-Speed Prefix Matching." *ACM Transactions on Computer Systems* vol.19, no. 4 (2001): 440-482.

[Wang08] Wang, F., L., Gao. "A backup route aware routing protocol-Fast recovery from transient routing failures." *IEEE INFOCOM.* AZ, USA, 2008. 2333-2341.

[Wang04] Wang, L., D., Massey, K., Patel. "FRTR: A scalable mechanism for global routing table consistency." *IEEE Conference On Dependable Systems and Networks.* Washington DC, USA, 2004. 465-474.

[XORP] *XORP: VYATTA community.* http://www.vyatta.org.

[Yang04] Yang, L., R., Dantu, T., Anderson, R. Gopal."Forwarding and Control Element Separation", RFC 3746, 2004.

[Zhang05] Zhang, X, P., Zhu, X., Lu. "Fully-distributed and highly-parallelized implementation model of BGP4 based on clustered routers." *IEEE ICN.* Reunion, France, 2005. 433-441.

[ZEBRA] *The open source zebra project* http://www.zebra.com.