Cooperation in networks and end-to-end quality of service guarantees for the Internet

Romain Jacquet

THÈSE / Télécom Bretagne
sous le sceau de l'Université européenne de Bretagne

pour obtenir le grade de Docteur de Télécom Bretagne
En accréditation conjointe avec l'Ecole doctorale Matisse
mention : Informatique

présentée par
Romain Jacquet
préparée dans le département Réseaux, Sécurité et Multimédia
Laboratoire Irisa

Cooperation in networks and end-to-end quality of service guarantees for the Internet

---

Coopération des réseaux et garanties de qualité de service de bout-en-bout dans l'Internet

Thèse soutenue le 8 juillet 2015
Devant le jury composé de :

Dominique Barth
Professeur, Université de Versailles / président

Jean-Louis Rougier
Professeur, Télécom ParisTech / rapporteur

Miklos Molnar
Professeur, Université Montpellier / rapporteur

Géraldine Texier
Maître de conférences, Télécom Bretagne / Examinatrice

Alberto Blanc
Maître de conférences, Télécom Bretagne / Examinateur

Xavier Lagrange
Professeur, Télécom Bretagne / Directeur de thèse

Olivier Dugeon
Ingénieur de recherche (HDR), FT/Orange Labs – Lannion / invité

N° d'ordre : 2015telb0357

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En accréditation conjointe avec l'Ecole Doctorale Matisse

Ecole Doctorale – MATISSE

Cooperation in networks and end-to-end quality of service guarantees for the Internet

---

Thèse de Doctorat

Mention : informatique

Présentée par **Romain Jacquet**

Département : Réseaux, Sécurité et Multimédia (RSM)

Laboratoire : IRISA

Directeur de thèse : Xavier Lagrange

Soutenue le 8 Juillet 2015

Jury :

M. Jean Louis Rougier, Professeur, TELECOM ParisTech, (Rapporteur)
M. Miklós Molnár, Professeur, Université Montpellier 2, (Rapporteur)
M, Dominique Barth, Professeur, Université de Versailles, (Examinateur)
M. Xavier Lagrange, Professeur, Telecom Bretagne, (Directeur de thèse)
Mme, Géraldine Texier, Maître de conférences, Telecom Bretagne, (Encadrante de Thèse)
M, Alberto Blanc, Maître de conférences, Telecom Bretagne, (Encadrant de Thèse)
M, Olivier Dugeon, Ingénieur de recherche, HDR, Orange Labs, (Invité)

Declaration of Authorship

I, Romain JACQUET, declare that this thesis titled, 'Cooperation in networks and end-to-end quality of service guarantees for the Internet' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Contents

List of Figures

List of Tables

Abbreviations

Acronym	Description
AS	**A**utonomous **S**ystem
AF	**A**ssured **F**orwarded
ASBR	**A**utonomous **S**ystem **B**order **R**outer
ark	archipelago
BA	**B**arabasi **A**lbert
BGP	**B**order **G**ateway **P**rotocol
BE	**B**est **E**ffort
BRPC	**B**ackward **R**ecursive **P**CE-based **C**omputation
CDF	**C**umulative **D**istribution **F**unction
DiffServ	**Diff**erentiated **Serv**ice
DCLC	**D**elay **C**onstraint **L**east **C**ost
EU	**E**nd **U**ser
EF	**E**xpected **F**orwarded
eLSR	**e**gress **L**abel **S**witch **R**outer
FEC	**F**orwarding **E**quivalence **C**lass
GPS	**G**lobal **P**ositioning **S**ystem
GLP	**G**eneralized **L**inear **P**reference
HD	**H**ighest **D**egree
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IP	**I**nternet **P**rotocol

IntServ	**Int**egrated **Serv**ice
iLSR	**i**ngress **L**abel **S**witch **R**outer
IRR	**I**nternet **R**outing **R**egistry
LAH	**L**ist **A**bbreviations **H**ere
LSP	**L**abel **S**witch **P**ath
LSR	**L**abel **S**witch **R**outer
MCP	**M**ulti-**C**onstraint **P**ath
NoE	**N**umber **o**f **E**dge
PCE	**P**ath **C**omputation **E**lement
PCEP	**P**ath **C**computation **E**lement **P**rotocol
PFP	**P**ositive **F**eedback **P**reference
QoS	**Q**uality **o**f **S**ervice
RSVP	**R**esource Reser**V**ation **P**rotocol
RCP	**R**emote **C**ontrol **P**lateform
sp	**s**hortest **p**ath
SLA	**S**ervice **L**evel **A**greement
SLS	**S**ervice **L**evel **S**pecification
TE	**T**raffic **E**ngineering
TCP	**T**ansmission **C**ontrol **P**rotocol
VoIP	**V**oice **o**ver IP
VoD	**V**ideo **o**n **D**emand
VSPT	**V**irtual **S**hortest **P**ath **T**ree

Chapter 1

Résumé

1.1 Introduction

L'Internet a récemment vu l'apparition de nouveaux services tels que IPTV, VoIP. Or, ces services, pour fonctionner de manières correctes, ont besoin de qualité de service (QoS). la QoS doit être considérée comme un problème de bout-en-bout car l'Internet est composé d'une interconnexion d'entités commerciales nommées Fournisseur d'Accès Internet (FAI) ou domaines reliant le client au service. Cependant, la nature de l'Internet est le principal facteur réticent à la mise en place de QoS de bout-en-bout. En effet, les FAIs doivent coopérer et se coordonner entre eux afin de fournir cette QoS. Or, chaque entité ne gère que localement son réseau de manière egoïste, sans perspective globale. Cela explique notamment pourquoi la QoS de bout en bout est un problème difficile qui peine à trouver une solution majoritairement acceptée. Même s'il existe des technologies garantissant de la QoS intra-domaine, ces dernières requièrent une parfaite coordination des domaines qui n'est pas envisageable à l'heure actuelle.

La garantie de la QoS de bout-en-bout soulève de nombreux problèmes techniques illustrés par la Figure 1.1. Dans cette thèse, nous nous focalisons spécifiquement sur la partie recherche de chemins.

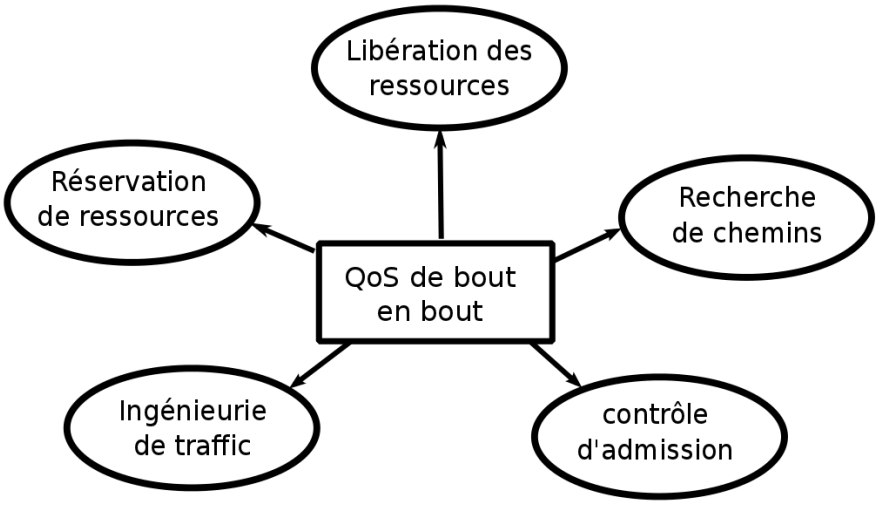


FIGURE 1.1: Principaux domaines impliqués dans la QoS de bout-en-bout.

1.1.1 Plan de la thèse

Cette thèse se compose de deux parties. La première consiste à présenter le contexte technologique ainsi que les connaissances fondamentales pour comprendre les travaux menés lors de cette thèse.

Au chapitre 3, nous commençons par présenter le modèle d'organization de l'Internet. Ceci nous permettra de montrer que la nature de l'Internet, c'est-à-dire ses entités et les relations purement financières qui les relies, constitue le principal obstacle à la mise en place de la QoS de bout-en-bout. Ensuite, après avoir définie la QoS, nous présentons les principales technologie qui fournissent de la QoS intra-domaine et nous montrerons pourquoi il n'est pas possible de les utiliser dans l'Internet. Après quoi, nous décrivons mathématiquement le problème de la QoS de bout-en-bout souvent appelé problème multi-contraint. Ce problème à été prouvé NP-complet signifiant que la recherche d'une solution exacte ne peut se faire qu'au prix de calculs prohibitifs. En même temps, nous montrerons que les algorithmes les plus pertinents sont inapplicables dans l'Internet parce qu'ils ne respectent pas la confidentialité et l'autonomie des domaines qui sont indispensables pour les communications inter-domaines.

Le chapitre 4 est dédié à l'étude des générateurs de topologie de l'Internet. Vouloir générer des topologies ressemblant à celle de l'Internet est un processus crucial afin d'obtenir une modélisation et des simulations réalistes pour pouvoir évaluer au plus juste les performances de nos algorithmes. Pour cela, nous présentons les caractéristiques de

cinq générateurs and nous discutons du choix que l'on a fait pour n'en sélectionner qu'un seul.

Dans la second partie, nous exposerons les contributions de cette thèse.

Dans le chapitre 5, nous présenterons SANP (**S**ub-Graph **A**lgorithm for finding feasible **N**on dominated **P**ath), un algorithme capable de trouver des chemins de bout-en-bout dans l'Internet satisfaisant un ensemble de contraintes de QoS données. D'une part, SANP respecte la confidentialité et l'autonomie des domaines et d'autre part, il ne fait pas l'hypothèse d'une chaîne de domaines pré-calculés. Nous évaluons les performances de SANP en comparant les chemins qu'il a trouvés avec la solution exacte.

Dans le chapitre 6 nous faisons l'hypothèse que les domaines sont prêts a former des alliances. Ainsi, nous présentons un algorithme que nous avons nommé ACQA (**A**lgorithm for **C**omputing end-to-end **Q**oS path within the Internet composed of ASes an **A**lliances) capable de trouver des chemins de bout-en-bout dans l'Internet composé d'alliances et de domaines restés indépendants. Comme nous sommes intéressés par le bénéfice de former des alliances, nous évaluons les performances d'ACQA en le comparant avec SANP. A l'aide de plusieurs scénarios permettant de mettre en valeur un paramètre spécifique, nous comparons la qualité des chemins trouvés par ACQA à la qualité des chemins trouvés par SANP.

Enfin, au chapitre 7, nous résumons les contributions de cette thèse et nous discutons des possibles directions pour de futures recherches.

1.2 Limites des technologies existantes

1.2.1 BGP

L'idée la plus triviale pour offrir de la QoS à travers plusieurs domaines est d'utiliser le protocole de routage inter-domaine Border Gateway Protocol (BGP). BGP est actuellement utilisé par tous les systèmes autonomes (AS) mais il comporte de fortes limitations quant à la prise en compte de la QoS. En effet, il ne sélectionne qu'une seule route pour une destination. Une possibilité est d'adapter BGP, mais il est difficile de le changer. Plusieurs extensions et modifications de BGP afin de fournir de la QoS à travers plusieurs

domaines ont été proposées [1, 2]. Mais, hégémonie de BGP et la confiance qui lui est apportée ne motivent pas les acteurs de l'Internet à son changement [3].

1.2.2 IntServ

l'Internet Engineering Task Force (IETF), l'organisme de régulation de l'Internet, a proposé dans les années 90 plusieurs architectures permettant de fournir de la QoS à l'intérieur d'un domaine. IntServ, proposée en 1994, a été la première architecture et se base sur la réservation de ressources. Elle a été développée pour permettre de garantir de la QoS stricte. IntServ distingue chaque service et pour chacun d'entre eux elle réserve des ressources le long d'un chemin. Ainsi, on dit que IntServ est une architecture basée sur le principe **flux par flux**. Afin de pouvoir gérer les flux de manière individuelle, les routeurs doivent exécuter de nombreuses tâches telles que:

- Contrôle d'admission pour pouvoir déterminer si un nouveau flux peut être accepté.
- Classification de paquets.
- Conformité avec les politiques : supprimer des paquets quand le trafic ne respecte plus les caractéristiques spécifiées.
- Ordonner les paquets selon les requêtes.

Étant donné la complexité élevée due à la gestion flux par flux, IntServ ne passe pas à l'échelle de la taille de l'Internet. De manière générale, on utilise IntServ dans des réseaux de taille réduite.

1.2.3 DiffServ

Contrairement à IntServ qui gère les flux de manière individuelle, DiffServ gère des flux agrégés. Proposée en 1998, DiffServ est une architecture se basant sur un dimensionnement au préalable du réseau. Après avoir défini des **classes de service**, ces dernières seront ensuite utilisées pour les communications. Afin d'agréger les flux, DiffServ se base sur deux principes. Lorsque qu'un flux entre dans un domaine DiffServ [1], il est

[1] un groupe de serveurs appliquant les même règles DiffServ

identifié. Généralement, l'identification se fait grâce à l'adresse source ou destination, le protocole de transport utilisé ou encore les ports dans l'en-tête de transport. Ensuite, selon l'identification, les paquets sont marqués avec le DiffServ code point (DSCP). Ce dernier indique le niveau de QoS auquel les paquets seront soumis via les classes de services. DiffServ défini trois classes de service: celle par défaut *best-effort* (BE), *expedited forwarding* (EF) et *assured forwarding* (AF). Par exemple, EF a été défini afin que le flux ne subisse que très peu de pertes, une faible gigue et un faible délai quand AF assure une large bande passante aux paquets.

Même si DiffServ passe mieux à l'échelle qu'IntServ, la configuration de chaque routeur afin d'avoir une politique globale cohérente est loin d'être triviale. Dans le même temps, à cause le manque d'une politique unique de l'Internet, limite l'utilisation de DiffServ à un domaine.

1.2.4 Traffic Engineering

Dans L'Internet, les liens subissent souvent des pics de trafic, des congestions peuvent apparaitre aux goulots d'étranglement ou encore, des ruptures de liens peuvent même survenir. Afin de surmonter ces problèmes, il est possible d'applique des politique d'ingénierie de trafic aux trafics de données. L'idée principale du paradigme de l'ingénierie de trafic est d'établir des chemins spécifiques aux trafics. Dans [4], les auteurs donnent une définition simple de l'ingénierie de trafic : *elle doit mettre le trafic la où il y a de la bande passante disponible.*

Mulitprotocol Label Switching (MPLS) est l'architecture la plus connue proposée par l'IETF permettant d'établir des chemins spécifiques dans un réseau. MPLS se base sur le principe de commutation d'étiquettes ou "labels" qui a été conçu pour d'accélérer le routage IP en évitant une recherche complexe dans les tables de routage. A l'intérieur d'un réseau MPLS les chemins sont soit mise en place manuellement via l'administrateur soit mise en place de manière automatique à l'aide d'un protocole de signalisation tel que Label Distribution Protocol. Étant donné qu'un chemin est unidirectionnel, on distingue le routeur d'entrée au routeur de sortie. Le routeur d'entrée a pour but d'étiqueter le trafic entrant en se basant sur l'interface sur laquelle il a été reçu. Chaque paquet est ensuite transmis jusqu'au routeur de sortie suivant son étiquette. Pour créer les étiquettes, le routeur d'entrée utilise une classe d'équivalence de transfert qui est une

table de correspondance entre un élément du paquet (ex: adresse MAC, adresse IP) et une étiquette. Le router de sortie à pour rôle de supprimer l'en-tête MPLS.

Dans le soucis de confidentialité, les informations de l'ingénierie de trafic ne doivent pas sortir du domaine. De plus, un routeur doit connaître toute la topologie du réseau pour pouvoir prendre des décisions de routage. Ainsi, il n'est pas possible de calculer des chemins MPLS qui traverse plusieurs domaines.

Afin de satisfaire au problème précédent, l'IETF propose deux méthodes. La méthode dite *par domaine* [5] qui construit un chemin inter-domaine en concaténant un ensemble de segments. Concrètement, le domaine source calcul un segment entre la source et un routeur de sortie puis envoie ce segment au domaine suivant. Le domaine suivant calcule lui un segment entre son routeur d'entrée et un routeur de sortir et concatène les deux segments. Il envoie ensuite la concaténation des segments au domaine suivant qui réalise alors la même opération et ce jusqu'à que la destination soit atteinte.

La deuxième méthode dite *Element De calcul de Chemin* [6] propose de calculer un arbre de plus courts chemins (APCC) entre la source et la destination. Similairement à la précédente méthode, le domaine source calcul un APCC entre la source et ses routeurs de sortie puis l'envoie au domaine suivant. Le nouveau domaine répète le processus et fusionne le résultat au APCC. Ce processus est répété jusqu'à ce que le APCC à atteint le domaine de la destination.

les deux méthodes font l'hypothèse d'une chaîne de systèmes autonomes pré-calculée pour soit construire le chemin dans la première soit l'APCC dans la seconde. Or, cette hypothèse constitue la principale limitation de ces deux méthodes. En effet, ces méthodes ne trouvent une solution que si et seulement une solution existe dans cette chaîne de systèmes autonomes pré-calculée.

1.3 Routage inter-domaine multi-contraint

Une solution envisageable pour offrir de la QoS est que les FAIs publient des offres qui correspondent à des garanties en termes de bande passante, délai, coût etc. entre paires de nœuds dans leur réseau. Nous faisons l'hypothèse que les FAIs sont prêts à publier ces informations. En effet, elles sont souvent utilisées dans les accords de niveaux

de services qui définissent les relations entres les différents fournisseurs et leurs clients. Ainsi, trouver un chemin à QoS garantie entre une source et une destination consiste à trouver un ensemble d'offres cohérent qui part de la source jusqu'à la destination. La Figure 1.2 montre un exemple de chemin de bout-en-bout qui est le résultat d'une concaténation de plusieurs offres.

Le problème de trouver des chemins satisfaisant plusieurs contraintes (bande passante, délai, coût) est souvent appelé problème multi-contraint. Les contraintes peuvent être soient additives soient min-max. Concernant les contraintes min-max, le poids d'un chemin d'unes de ces contraintes est le minimum (ou le maximum) des poids des liens constituants le chemin. Un exemple de contraintes min-max est la bande passante. Pour les contraintes additives, la valeur ou le poids d'une contrainte le long d'un chemin est la somme des poids des liens constituants le chemin. Des exemples de contraintes additives, on peut citer le délai, le coût, le nombre de sauts (la distance). Nous nous concentrons seulement sur les contraintes additives. En effet, pour les contraintes min-max il suffit d'éliminer les liens qui ne satisfont pas les contraintes. Dans [7], Wang et Wrowcroft ont prouvé que chercher des chemins avec deux contraintes additives ou plus est un problème NP-complet. Toutes solutions au problème multi-contraint doit respecter les propriétés des communications inter-domaine : confidentialité, autonomie et scalabilité. La confidentialité est le fait de ne pas révéler la topologie interne d'un FAI et l'autonomie est le fait de laisser l'administrateur d'un FAI le choix de la politique à l'intérieur de son réseau.

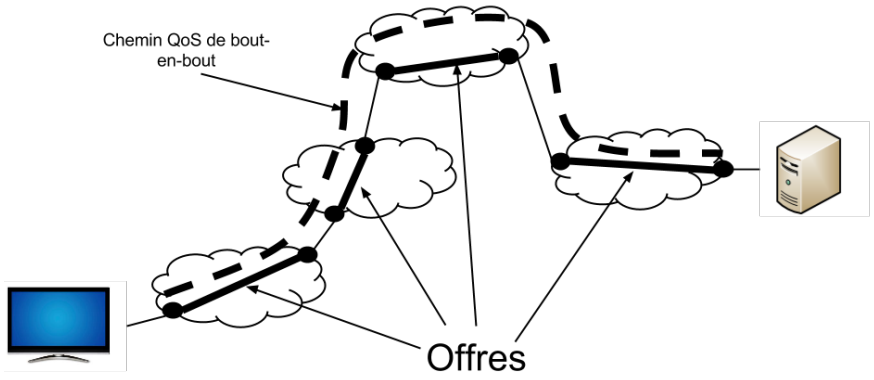


FIGURE 1.2: chemin de bout en bout composé de plusieurs offres

1.3.1 Algorithmes

D'un point de vue commercial, notre but est de rechercher un ensemble d'offres cohérent entre la source et la destination. D'un point de vue mathématique, on recherche un ensemble de chemins satisfaisant plusieurs contraintes entre la source et la destination. Peu importe le point de vue, nous proposons un algorithme nommé SANP [8] qui a pour but de satisfaire les objectifs précédemment mentionnés. Pour cela, SANP ne fait pas l'hypothèse d'une chaîne d'ASes pour calculer des chemins de bout-en-bout. En revanche SANP fait l'hypothèse que les ASes sont prêts à publier des offres entre leurs routeurs de bordure. Ainsi, à la fois la confidentialité et l'autonomie des ASes sont respectées. Afin de trouver un ensemble cohérent d'offres, SANP construit un sous-graphe autour de la route donné par le protocole de routage sous-jacent entre la source et la destination. Ce sous-graphe est obtenu en fusionnant les voisinages de chaque nœud traversé par la requête. Nous définissions un voisinage comme l'ensemble de tous les nœuds et liens en deçà d'une certain distance r_0. Ce sous-graphe est reçu par la source qui l'utilise pour calculer un ensemble de chemins faisable non-dominés. Par soucis d'autonomie, la source peut utiliser l'algorithme de son choix pour déterminer un ensemble de chemins faisable non-dominés. De plus, nous avons implémenté deux heuristiques qui ont pour but de limiter la taille du sous-graphe de tel sorte que SANP est scalable avec r_0. Les deux heuristiques limitent la taille du sous-graphe à M avec M égal à $\alpha \times L$ et L égal à la longueur du plus court chemin entre la source et la destination.

Ensuite nous avons fait l'hypothèse que les ASes sont prêts à former des alliances. Le principal but des alliances est de mieux satisfaire aux requêtes des consommateurs en mutualisant les ressources et ainsi augmenter ses revenues. Nous proposons un algorithme nommé ACQA capable de calculer des chemins de QoS de bout-en-bout dans un graphe composé d'alliances et d'ASes. Pour cela et de la même manière qu'avec SANP, nous faisons l'hypothèse que les ASes et les alliances publient des offres de QoS entre leurs points d'entrées et de sorties. Ainsi ACQA respecte aussi la confidentialité et l'autonomie des ASes. La source utilise ensuite ce sous-graphe pour calculer un ensemble de chemins faisables non-dominés et en sélectionne un selon ses préférences. Nous utilisons une heuristique qui limite la taille du sous-graphe pour être scalable. Nous montrons grâce aux simulations que ACQA satisfait un plus grand nombre de requêtes que

SANP et que les chemins trouvés par ACQA sont de meilleure qualité que les chemins trouvés par SANP. Ce résultat encourage les ASes à former des alliances.

1.4 Conclusion et Perspectives

Dans cette thèse nous avons étudié le problème de recherche de chemins de bout-en-bout satisfaisant plusieurs contraintes. Nous avons montré que c'est un problème difficile parce qu'il implique autant des contraintes techniques que commerciales. En effet, l'Internet est composé d'une interconnexion d'ASes qui limite la quantité d'information échangée et chaque AS applique une politique qui vise à optimiser son réseau sans coopération et vision globale. Ainsi, une solution au problème de recherche de chemins inter-domaines doit respecter à la fois la confidentialité mais aussi l'autonomie des ASes.

Les solutions que nous proposons tiennent en compte ces contraintes. De plus, nous avons introduit un paramètre r_0 qui permet de limiter la taille du sous-graphe construit par SANP et ACQA et que ces derniers soient ainsi scalable. Enfin, nos algorithmes ont la particularité de fonctionner avec n'importe quel mécanisme de routage qui augmente ainsi leur portabilité.

Nos travaux peuvent être prolonger dans plusieurs directions. Premièrement, il serait intéressant d'approfondir les simulations afin de déterminer l'influence des graphes générés par Inet. En effet, de nombreux travaux ont été réalisés afin de mieux connaitre la topologie de l'Internet et de simulations plus réalistes pourraient être mener.

Enfin, dans le soucis de simplicité, nous avons fait l'hypothèse d'un réseau stable avec des acteurs qui satisfaisaient toujours leurs contrats. Or, afin d'être plus réaliste, il serait d'intéressant de faire l'hypothèse de défaillance de liens ou de triche de la part des acteurs. L'introduction d'un mécanisme de réputation pourrait être une solution à ce problème et pourrait être une direction à suivre.

Chapter 2

Introduction

The Internet is now used for accessing a plethora of services such as the web, e-mail, video games or even telephony. In order to deliver these services, the Internet relies on the IP protocol. The best-effort service offered by IP often suffices to satisfy most of the quality of service (QoS) requirements (e.g., e-mail, web). However, for network applications that require strict QoS, such as telephony or video gaming, the best-effort service does not offer any QoS guarantees. For instance, telephony does not tolerate long delays while video gaming needs a large bandwidth and small delay. Therefore, in order to achieve their full potential, these services need the widespread adoption of appropriate QoS technologies.

Several frameworks exist to satisfy QoS network applications within a single network. However providing end-to-end QoS guarantees to traffic traversing several providers is still an unsolved problem. This subject is the main topic of this dissertation.

2.1 Problems Addressed in the Dissertation

2.1.1 End-to-End QoS

Given the nature of the Internet, most of the time, the source and the destination are connected to different providers. As a consequence, the QoS within the Internet must be considered as an end-to-end problem. At the same time, the nature of the Internet is the main obstacle to provide end-to-end QoS. The Internet is composed of thousand of

Autonomous Systems (ASes) that are self-governed entities in competition. Hence, they limit the quantity of information they exchange and are unwilling to strongly cooperate which is essential to offer end-to-end QoS.

On the one hand, works such as [3] or [9] identify that no business plan incites the providers to implement an end-to-end QoS framework. The investment and the maintenance of a QoS framework are prohibitive and no return to investment is guaranteed causing it to loose the interest of the providers. On the other hand, European projects such as TEQUILLA [10], MESCAL [11] or ETICS [12] involving universities and leading operators propose ecosystems allowing end-to-end QoS with the concerns to fairly redistribute the benefits and to demonstrate their profitability.

Deciding whether a QoS framework enabling end-to-end QoS is profitable for the providers or not is not the scope of this thesis. However, many applications need end-to-end QoS guarantees in order to offer an acceptable service. VoIP is a telephony service that requires low delay and low jitter to work correctly. Another important service for which QoS is required is video-on demand (VoD), which takes an ever-increasing share of the total traffic in the Internet, as Figure 2.1 shows. VoD services need a large bandwidth to work correctly. A typical method to provide such QoS guarantees is to find a path from the source to the destination with guaranteed performances and to reserve resources along this path. Figure 2.2 shows an example of paths with guaranteed performances reserved for both VoIP and VoD services to offer end-to-end QoS.

2.1.2 Service Level Agreements

In order to guarantee a certain level of QoS, a customer, or the user of a service, signs a contract called *Service Level Agreement* (SLA) with its provider. An SLA is the formalization of an agreement negotiated between two parties. The technical part of the SLAs is called the *service-level specification* (SLS). In the SLSs, QoS metrics are described, e.g., a transmission delay lower than 50 milliseconds, a packet-lost rate lower than 0.1 percent and a jitter lower than 10 milliseconds and are guaranteed for the traffic described in the contract. As these guarantees usually apply within a single domain, several SLAs must be combined in order to offer end-to-end guarantees. Figure 2.3 shows an example end-to-end QoS provisioning by combining several SLAs.

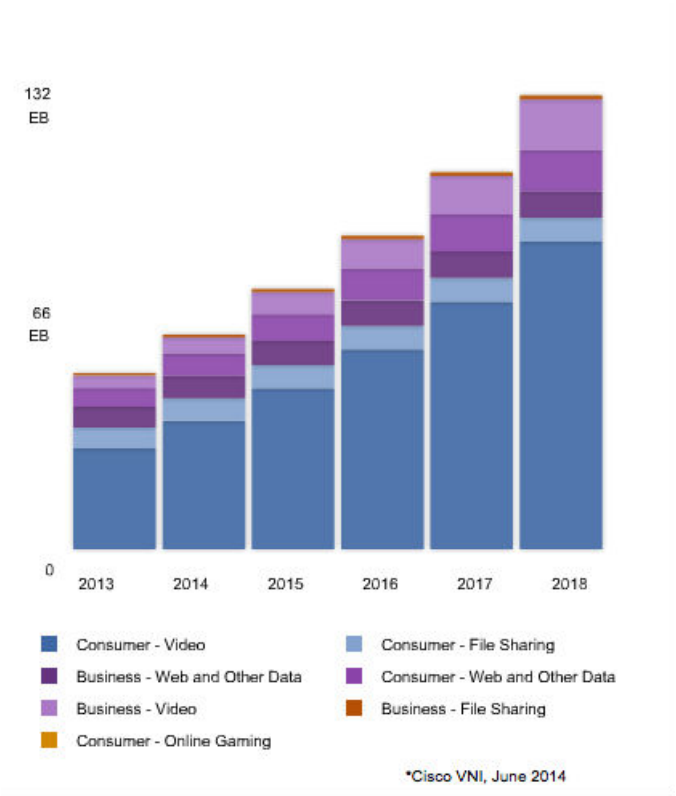


FIGURE 2.1: Internet traffic types

AS the problem of provisioning end-to-end QoS covers various domain, it can be divided into sub-problems as Figure 2.4 shows.

- **SLA implementation**: A provider must decide how to implement the SLAs that it is offering, using its current infrastructure. The way a provider configures its SLAs is based on an internal process such as performance capabilities or business strategies and is outside the scope of this work.
- **Network path computation**: This phase consists on computing a path that answers the customer QoS-requests. It is known that finding paths with a number of additive constraints, e.g., delay and cost, greater than or equal to 2 is NP-complete [7]. The goal of this dissertation is to propose scalable algorithms to find end-to-end paths with guaranteed QoS, hence this dissertation focuses on this part.
- **Network monitoring**: As the name indicates, it consists in monitoring the network in order to ensure that the technical performance expressed in the SLAs is respected.

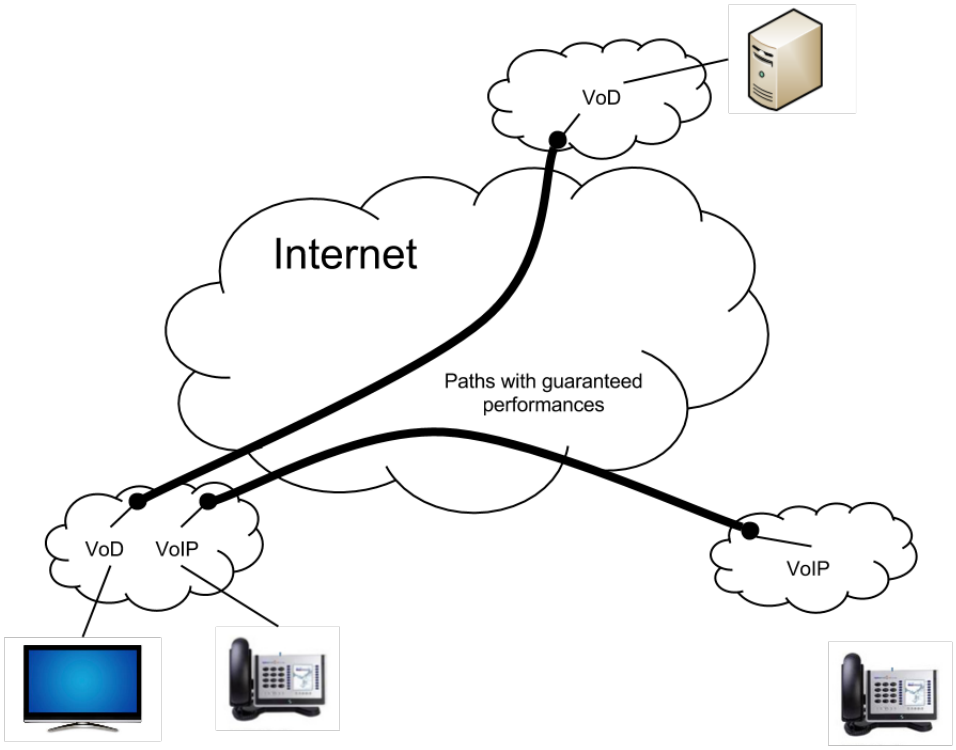


FIGURE 2.2: Use cases of end-to-end QoS provisioning

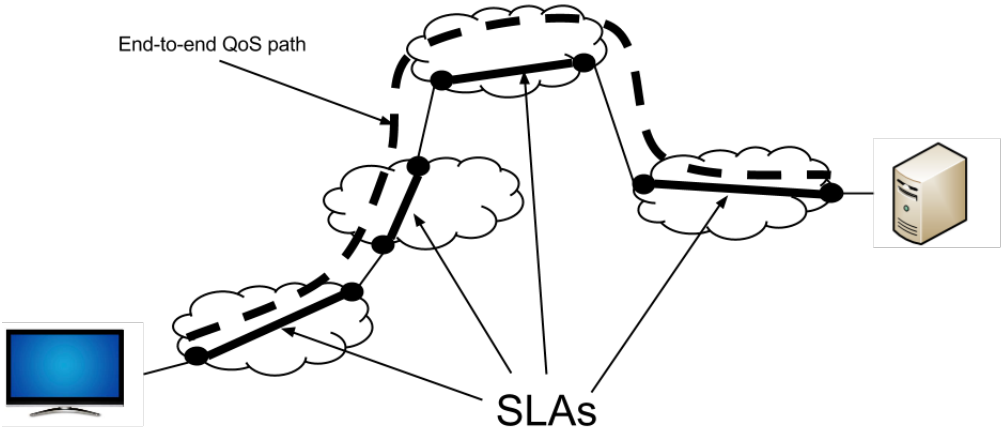


FIGURE 2.3: Providing end-to-end QoS by combining SLAs

- **Admission Control**: This phase consists in validating whether the current resources are sufficient to satisfy the QoS constraints of new connections.
- **SLA termination mechanism**: The termination process frees reserved resources and configuration settings linked to the SLA instance to make them available for subsequent SLA requests. This phase ends the SLA lifecycle therefore also it terminates the provisioning phase.

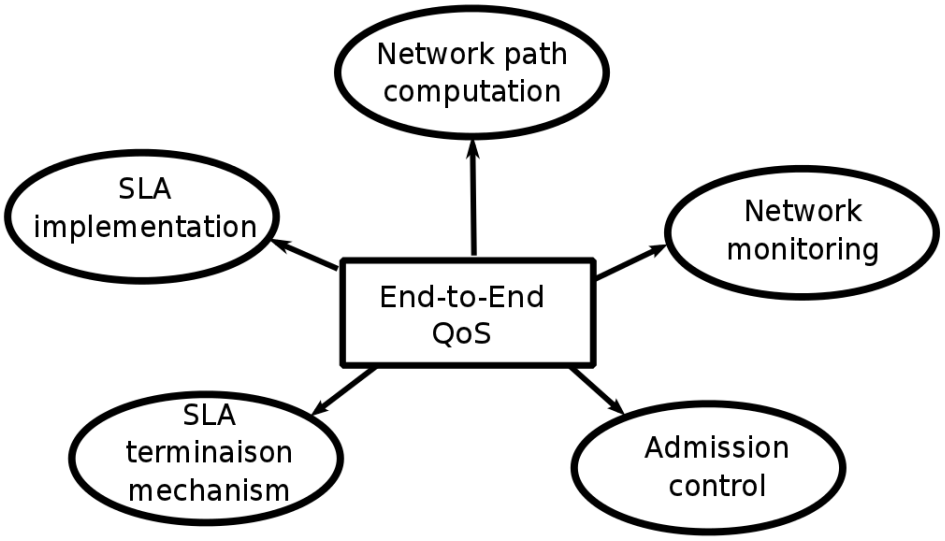


FIGURE 2.4: Sub-problems for providing QoS across multiple providers

2.2 Scope and Objectives of the Dissertation

2.2.1 End-to-end QoS Paths Computation Algorithms

Path Computation Element (PCE) [13] is an architecture proposed by the Internet Engineering Task Force (IETF) to provide end-to-end QoS in the Internet. The PCE architecture installs one or several PCEs in each domain that are able to perform complex paths computations. Moreover, each PCE has a Traffic Engineering Database (TED), i.e., a table containing the internal topology of the domain and Traffic Engineering (TE) information. A Path Computation Client (PCC) is defined as any client application that requests path computations to a PCE thanks to The PCE Communication Protocol (PCEP) [14]. The key features of the PCE architecture are to preserve the confidentiality (and autonomy) of each domain as well to scale to thousands of domains. Both these features are needed by any solution aspiring to be deployed throughout the Internet.

Relying on cooperation of PCEs across different domains, protocols such as [5, 15] and algorithms such as [16, 17] are able to to compute end-to-end QoS paths. In order to optimize the resources management within a domain, the PCEP extensions for Stateful PCE [18] have been recently proposed. They propose to introduce a database, called LSP-DB, in which information about paths previously computed are saved. Although

it allows a fine-grained path computation, it introduces complexity since synchronization between LSP-DB and current state of the network is needed to perform correctly. Moreover, links failures or how to handle restart issues have not yet been addressed.

Most of the existing techniques for computing end-to-end QoS paths assume that a pre-computed sequence of ASes between the source and the destination is given [13]. As an example, in Figure 2.5, the sequence of domains ASes [s, b, d , t] is given in order to compute end-to-end QoS paths between ASes s and t. On the one hand, [19] precises that *no explanation is given of how this sequence is generated.* An *auto discovery* procedure is assumed to provide the sequence of domains but no criteria are specified for the selection. On the other hand, in [20], it is mentioned that the sequence of domains is given by the border gateway protocol (BGP). For both cases, the pre-computed sequence of ASes constitutes one main important limitation of the PCE architecture. Indeed, it is either arbitrarily given or given by BGP and for both, the sequence of domains is not QoS-driven.

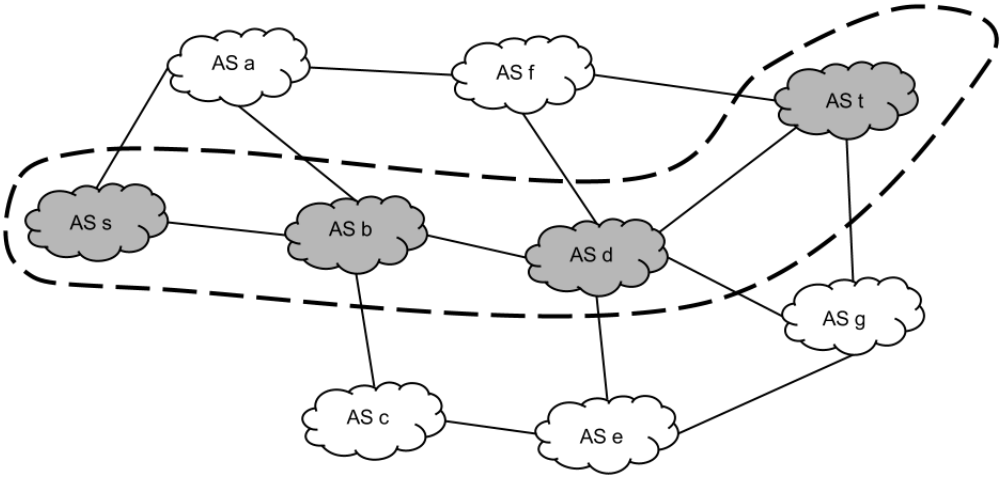


FIGURE 2.5: Example of a pre-computed sequence of domains

The major concern of this dissertation is to compute end-to-end QoS paths without assuming a pre-determined sequence of ASes between the source and the destination. Any mechanism aiming at solving this issue must remain scalable. Furthermore, special requirements such as respecting the confidentiality and the autonomy of the ASes are essential in the inter-AS communications. The confidentiality propriety guarantees that no proprietary information, such as internal topology, is exchanged between different

domains. The autonomy propriety is the fact that each AS is free to decide how it implements its services, without any influence from the other domains of the end-to-end path. .

2.2.2 Contributions

The first contribution of this dissertation is an algorithm called SANP [8]. SANP can be used to find feasible end-to-end paths, while satisfying the confidentiality and independence of each AS. Furthermore, contrary to other existing solutions (e.g., [17, 21]), it does not assume a pre-determined sequence of ASes to compute end-to-end QoS paths.

In the final part of the dissertation, we assume that ASes are ready to form alliances. The main purpose of forming alliances is to facilitate the provisioning of end-to-end QoS paths leading to increased revenues for the participating ASes. Our second contribution is ACQA [22], an algorithm able to compute feasible end-to-end paths in the presence of alliances. It is an extension to the first algorithm and hence respects the inter-domain communication properties.

2.2.3 Organization of the Manuscript

We organize this thesis in two parts. The first part consists in presenting the technological context as well as the required background to understand the work carried out during the Ph.D. program.

We start the first part by introducing the AS-level model of the Internet (Chapter 3). The goal is to show that the nature of the Internet, i.e., its entities and the business-driven relationship between them, is a significant obstacle to offer end-to-end QoS. After giving a definition of QoS, we then introduce the principal existing technologies that can be used satisfy QoS constraints within a single domain. We will show why they can not be extended to the whole Internet. Then, we formally present the problem of finding end-to-end QoS path as a multi-constrained path problem or constrained routing problem. This problem has been proven to be NP-complete meaning that finding an exact solution is computationally unfeasible. One technique often used is to transform the multi-objective optimization problem into a mono-objective optimization problem for which it is computationally faster and easier to find a solution. However, we will

show that the most relevant algorithms are inapplicable for the Internet because they do not respect the confidentiality and the autonomy of the domains which is essential in the case of inter-domain communications.

Generating a realistic Internet-like topology is a vital part for running realistic simulations, which are needed to assess the performances of our algorithms. Hence, Chapter 4 is dedicated to the study of existing generators of Internet-like topologies. We start by presenting different methods that collect data in the Internet and use them in order to infer the Internet topology. As each AS limits the information spread about its interconnections, we will show that each method makes assumptions that affect the accuracy of the estimated Internet topology. In a second part, we first present an overview of the different models that have been used to generate Internet-like graphs. Then, we focus on five generators and compare their performances. Finally, we discuss our choice to use only one (inet).

The second part consists in presenting the contributions of this dissertation.

In Chapter 5, we introduce SANP (**S**ub-Graph **A**lgorithm for finding feasible **N**on dominated **P**ath), an algorithm capable of finding end-to-end paths in the Internet satisfying a set of given QoS constraints. On the one hand, SANP respects the confidentiality and the autonomy of the ASes and, on the other hand, it does not assume a pre-computed sequence of ASes. We assess the performance of SANP, through simulations, by comparing the paths it finds to the exact solution.

In Chapter 6, we assume that the ASes are ready to form alliances. We present an algorithm named ACQA (**A**lgorithm for **C**omputing end-to-end **Q**oS path within the Internet composed of ASes an **A**lliances) capable of finding end-to-end QoS paths in the Internet composed of alliances with some ASes remaining independent, i.e., not belonging to any alliance. As we are interested in estimating the benefit or not of forming alliances, we assess the performance of ACQA by comparing it with SANP. In several simulations scenarios for which we highlight a specific parameter, we compare the quality of the paths found with ACQA to the quality of the paths found with SANP.

Finally in Chapter 7, we synthesize the contributions of this dissertation and discuss the possible directions for future research.

Chapter 3

End-to-End QoS provisioning

3.1 Introduction

In this chapter we give an overview of end-to-end QoS in the Internet. The Internet was initially designed to offer only a best effort service, with no specific QoS guarantees. While this is a viable solution for many applications, some applications need end-to-end QoS guarantees in order to offer an acceptable user experience. Due to the current trend of **everything over IP**, the number of applications and services that are migrated to the Internet and that need QoS guarantees is increasing. Examples of these services are telephony, videoconferencing and video distribution/broadcasting.

We start by presenting the organization of the Internet in Section 3.2.1. The goal of this section is to help the readers to understand the limit of the Internet for fully satisfying the potential of services such as telephony or videoconferencing. Then, after giving a definition of QoS (Section 3.2.3) and present the specific challenges for providing end-to-end QoS (Section 3.2.4), we introduce the most well-known QoS models used in the Internet in Section 3.2.5. We focus especially on their limitations and explain why their utilization are restricted to a single AS and can not be used to the whole Internet. In Section 3.2.6, we focus on the negotiation and the enforcement of inter-domain SLAs to provide end-to-end QoS. As this is a promising solution, our work is inspired from this approach.

Providing end-to-end QoS can be seen as a constrained routing problem, often called the multi-constraint path (MCP) problem in the literature. In Section 3.3, we introduce

the formal definition of the MCP problem. As the MCP problem has been proven to be NP-complete [7], the literature often deals with the worst-case complexity and most of the time violates the inter-domain communications properties. Hence, the existing algorithms are inapplicable at the inter-domain level requiring novel solutions (Section 3.3.3). Finally, we discuss our approach in Section 3.3.4.

3.2 Quality of Service

3.2.1 Internet Model Organization

The Internet is an interconnection of independent networks owned by different companies. Each network is called an Autonomous System (AS). More precisely, an AS is a network or a set of networks supervised by a single administrative entity such as a company, or a university, or a public institution. Each AS applies its own policies independently of the other ASes. Each AS is uniquely identified by an AS number. At the time of writing this thesis, the number of advertised ASes is almost equal to 55000 [1].

It is useful to further characterize the main Internet actors, in three different groups. First of all, the **service providers**, as their name suggests, offer to their services to the second group, the **customers**. In order to achieve this, they use the connectivity offered by the **network providers** to reach the customers. In same cases, the same entity, can play different roles at the same time, for instance a network provider can also provide a certain number of services, making it a service provider as well. It is often the case that the network providers, e.g., Orange, propose their own services such as VOD or video games to their customers. We refer both the service providers and the network providers as ASes.

The relationships between ASes can be classified in two different ways. The first possibility if to make a distinction between public and private relationships called peering in this case. The public peering are performed across a shared network called Internet eXchange Point (IXP) thanks to a Layer 2 technology. At a IXP, multiple carriers interconnect each other using one or more physical connections. Generally, a member connected to an IXP adopts an open peering policy [23, 24]. To do so, it is connected to

[1] http://www.cidr-report.org/as2.0/

a route server in order to receive all the routes in the IXP and thus to peer with all the members. Exceptions such as *Carrier A does not want to peer with Carrier B* can be set up thanks to the BGP community parameter. Public peering is interesting for small networks since it offers an excellent way to be interconnected with many other networks.

A private peering is an interconnection between only two ASes. Generally, this kind of relationship is set up between two ASes that generate the same quantity of traffic. The main advantage of establishing a private peering is to guarantee the capacity between the members. Moreover, it is more reliable and more secure than a public peering. However, adopting a private peering policy requires more time to setup. Nowadays, most the Internet traffic is carried through private peering [25, 26].

While two ASes interconnected via a public peering have agreed to exchange traffic between them without any monetary compensation, private peering often call for payments to be made based on the actual traffic carried.

The second possibility, often used in the literature [27–30], is to classify the relationships between the ASes into three types. The first type is the **customer-provider (c2p)** relationship. It connects a customer that pays a provider for carrying its traffic towards the Internet. It can be assimilated to a private paying peering. The second type is the **peer-to-peer (p2p)** relationship. Two ASes in a p2p relationship freely exchange traffic between them. The traffic can come from the ASes themselves or from their customers.

In a **sibling-to sibling (s2s)** relationship, the traffic is freely exchange between the ASes only. Generally, It is established between two ASes belonging to the same company but geographically distant. This type of relationship usually appears as a result of mergers and acquisitions. For instance, AT&T owns several networks around the globe that are connected with sibling-to-sibling relationship [27].

In a work published in 2007 [30], the authors have inferred the type of relationships within the Internet. They computed that almost 91% of the relationships are customer-provider, 9.21% are peer-to-peer and less than 0.5% are sibling-to-sibling. Because the vast majority of the relationships are of type customer-provider, the Internet is organized in a multi-tier hierarchical structure as illustrated by Figure 3.1.

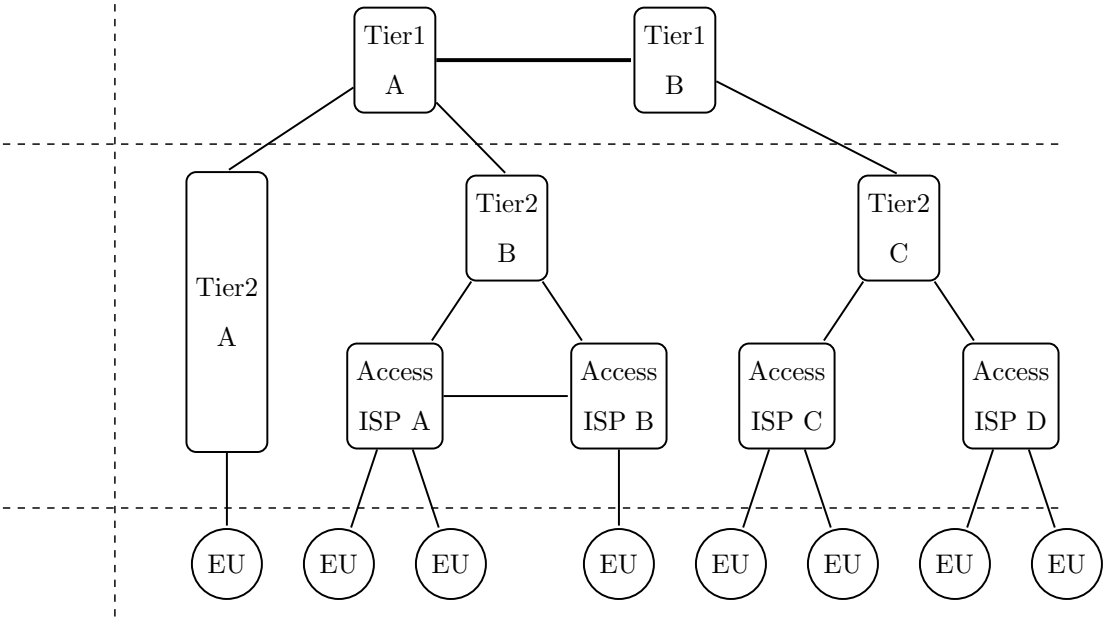


FIGURE 3.1: Multi-tier Internet structure

At the top of Figure 3.1 are presented the **tier-1** providers. Between 10 and 20 [31] are well inter-connected by free peering agreements. The tier-1 providers are often referred as the default-free zone meaning that their routing tables do not have the default route entry. They are also referred as the Internet core or Internet backbone since a tier-1 provider is a provider that can reach any network on the Internet without paying any IP transit.

In the middle of Figure 3.1, we can find the **tier-2** providers and the access providers. By definition, a tier-2 provider can peer with other networks but still needs to pay IP-transit fees to reach some portion of the Internet. As its name indicates, an access provider gives access to the Internet to the end-users (EU).

Recent studies [31–33] show that the multi-tier structure of the Internet is turning into a flat structure as Figure 3.2 shows. The main reason is that the content providers are bringing their networks closer to the users [34]. The flatting Internet structure seems to shorten the distance between two ASes. However, recent studies [35, 36] show that the average path length is stable around 3.8 since 2001.

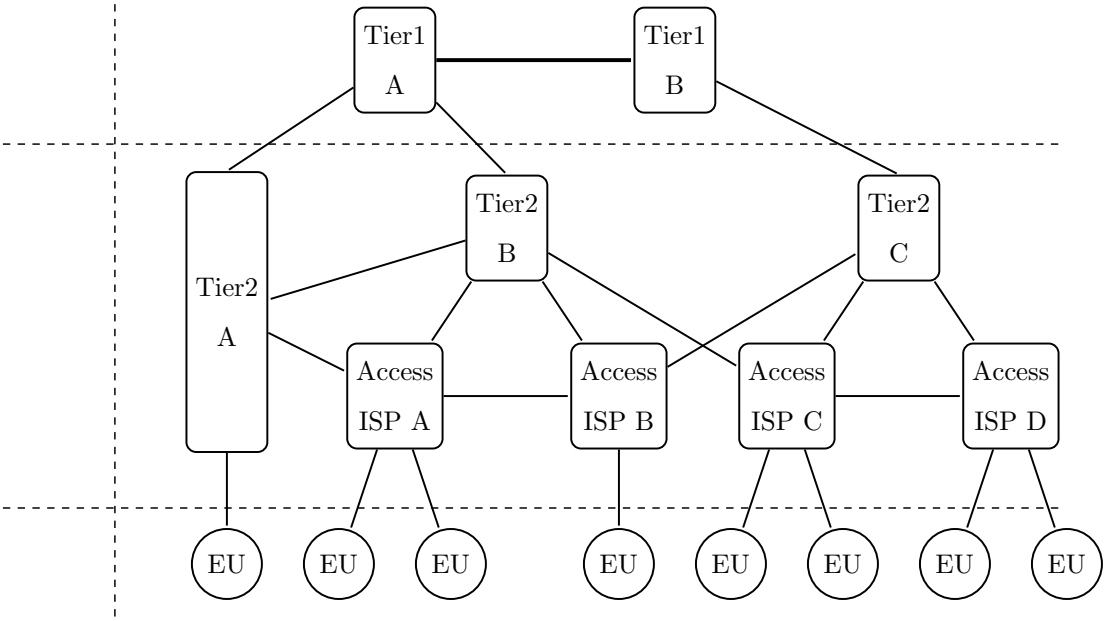


FIGURE 3.2: Flat Internet structure

3.2.2 Why QoS in the Internet ?

The last decade has seen the emergence of new technologies leading to the **everything over IP** philosophy. Recall that it consists in using IP networks, including the Internet, to offer services like telephony or videoconferencing.

However, the Internet is based on the **packet switching** principle and can not give guarantees to these new services. As a consequence, the **everything over IP** philosophy can work only if the Internet is modified in order to accommodate these new services. For instance, telephony relied on a **circuit switching** system. It consists in establishing a circuit through the network between two end-nodes. This circuit is strictly reserved to one communication. Thus, it guarantees the full capacity and remained connected throughout the duration of the call. Hence, the QoS of the communication was strictly guaranteed by the nature of the network. By transferring the telephony into a packet switched network relying on a best effort service, the capacity to guarantee any QoS to an application has been removed. Moreover, it is often the case that multinational companies need to communicate with their clients or with their owns staff, located at remote sites. These communications have to be secured and require end-to-end QoS in the case of telephony and/or videoconferencing. One viable solution is to set up a Virtual Private Network (VPN) with QoS guarantees [37]. However, setting up a VPN

requires human interventions that can take several weeks to be completed. Moreover, human configuration errors can occur. A solution that could automate this process and reduce the setup delay to a few seconds, or even a few minutes, would be a significant improvement.

Network providers have tried to alleviate the lack of QoS guarantees, implicit in a best effort service, by increasing the capacity of their networks. The idea being that an over-provisioned network can offer a best effort service that is good enough even for some applications that need QoS guarantees (e.g., small delays). Even if this strategy works in some cases, it is not a panacea, as it does not guarantee **strict** or **hard** QoS. In other words, it cannot guarantee that given bounds on one or more QoS metric will always be satisfied. Moreover, over-provisioning a network has a non negligible cost, which keeps getting higher due to the ever-increasing network load. Such a solution is therefore not sustainable.

3.2.3 Quantifying the QoS

To qualitatively define the level of QoS, we can distinguish two types of QoS constraints. The first category is the **min-max** constraints. When a path has more than a single link, the value of the corresponding metric for the whole path is the minimum (resp. maximum) of all the values for that metric for each link. As an example, bandwidth is a min-max constraint. The second type of constraint is the **additive** constraints. The value of the corresponding metric for a path is the sum of the values of the links in the path. As an example, the delay and the total path cost are additive metrics. Note that there exist **multiplicative** constraints such as independent packet-loss probabilities on each link. In this case, the probability of successful transmission of the packet is the product of one minus the loss probability of every link of the path. The **multiplicative** constraints can be transformed into additive constraint by taking the logarithm.

3.2.4 Inter-domain Communication Properties

In this section we discuss the requirements of any mechanism whose goal is to enable the automated provisioning of end-to-end QoS guarantees. The key feature of any such system is that it must correctly handle the needs of each AS, considered as an independent

entity.

First, an AS is in charge of defining its own routing policy for its network and has complete autonomy on this, e.g., it can use any routing protocol such as OSPF [38] or RIP [39]. As a consequence, any end-to-end QoS mechanism has to respect the autonomy of each AS. Furthermore, it does not has to be restricted to a specific technology and must take into account the heterogeneity of the Internet.

Second, due to the strong competition between the domains [40], an AS is reluctant to diffuse confidential information such as its internal topology [41]. Thus an end-to-end QoS mechanism must respect the confidentiality of each AS and can rely only on public information, e.g., the entry and the exit points of the domains.

In addition to the autonomy and confidentiality requirements, an end-to-end QoS mechanism must be scalable. It is not conceivable to probe the entire Internet to find a path with guaranteed QoS. All these requirements make the problem of provisioning end-to-end QoS especially complicated.

3.2.5 Technological Context

The goal of this section is to present the limitations of the existing technologies for providing end-to-end QoS. We focus especially on BGP, the de-facto inter-domain protocol. And on IntServ, DiffServ and MPLS, the three most well-known architectures proposed by the IETF to offer QoS guarantees, at least within a single domain.

Recall that ASes can offer SLAs and that it is possible, at least in theory, to combine several SLAs to build end-to-end paths with guaranteed QoS. In this section, we also present several works that focus on finding a chain of SLAs.

BGP Extensions

BGP has been designed to exchange routing and reachability information between ASes on the Internet. However, in its most commonly used form, BGP does not support QoS: for one destination corresponds to only one route [42, 43]. Moreover, the routes are often chosen as the shortest path in term of number of hops, ignoring any QoS aspect. Plenty of BGP extensions have been proposed at the IETF to add QoS capabilities to BGP. For instance, works such that [44] and [1] propose to integrate a new attribute in order

to allow BGP to exchange QoS information between ASes. In [44], the authors propose to chose the route with the maximal available bandwidth while in [1], they propose to advertise QoS-enabled reachibility information so that the ASes can select the routes thanks to several QoS parameters. In [45], the authors propose Multi-path BGP, which, as its name indicates, can use multiple routes for a given destination.

These proposals have not been widely adopted for two main reasons. First, due to scalability and convergence time issues, it is difficult to change BGP [43]. Second, the ISPs are reluctant to change BGP due to its widespread utilization and for no clear source of revenue [46]. For these reasons, changing BGP for supporting QoS is not a promising way to provide end-to-end QoS guarantees in the Internet.

Providing QoS in the Internet is a problem that has been raised more than two decades ago. In 1994 the IETF proposed IntServ, an architecture for assuring strict QoS by treating each (micro) flow individually. Four years later, it proposed DiffServ that suppresses per-flow management and adopts an aggregate flow strategy. Finally, in 2001, it proposed MPLS that enforces explicit paths within a network. In the remainder of this section, we discuss briefly these three architectures.

IntServ

The idea of IntServ is to explicitly reserve the resources needed by each flow. For this purpose, IntServ uses the Resource reSerVation Protocol [47] (RSVP). For a given flow, RSVP starts to find a path between the source and the destination with the help of a PATH message. Then, the resources are reserved via a RESV message. This **flow-by-flow** technique enables IntServ to offer a fine-grained control over QoS. However, to achieve this flow scale, the routers must implement plenty of features such as:

- Admission control to determine whether a new flow can be accepted.
- Packets classification to respect the specified level of QoS.
- Compliance with policies: e.g., dropping the packets when the traffic does not conform to its specified characteristics.
- Scheduling the packets according to the QoS requests.

In a network with many flows, performing each of the four previous operations introduces a non-negligible processing overhead [48]. Hence, IntServ suffers from scalability limitations that limits its utilization to small-scale networks.

DiffServ

Unlike IntServ, which handles the flows individually, DiffServ is based on an a-priori dimensioning that attributes resources to flows identified by a class of service. Because of this, DiffServ is more core-network oriented than IntServ. In order to set up aggregated flows, DiffServ relies on two principles. First, whenever a flow enters in a DiffServ domain [2], it is classified. The classification is performed thanks to the IP header of the packets, namely the source and destination IP address, the value of the DiffServ field and the upper layer protocol [49]. Then, depending on the result of the classification step, the packets are marked with a DiffServ code point (DSCP), which indicates the level of QoS that the packets will be subjected to through Per-Hop Behaviors (PHB). Three fundamental PHBs have been standardized and associated with specific use [50].

- *Default* PHB. Best-effort behavior.
- *Expedited forwarding* (EF) PHB [51]. For low delay, low loss and low jitter traffic.
- *Assured forwarding* (AF) PHB [52]. The AF PHB is divided into four classes of service, namely AF1, AF2, AF3 and AF4, each of them allocates a certain amount of forwarding resources in the DiffServ nodes. Each class is divided into three subclasses, each of them corresponds to a *drop precedence* (high, medium, low) that is applied to the packets. In other words, if the queue is full, it indicates which packets should be dropped first. The higher the drop precedence is, the higher the number of packets dropped.

As mentioned in [50], configuring correctly the routers in order to avoid starvation of low-priority queues while favoring high-QoS traffic in case of congestion is a very hard task. To tackle this issue, network operators have to either over-dimension their networks or to carefully manage and control the resources in their network. In [53], the authors propose to use Bandwidth Broker (BB) in order to manage the bandwidth resource

[2] a group of routers that implement common Diffserv policies

within a network. A BB is an agent that allocates resources by accepting or rejecting requests based on the policies of the network domain.

Several architectures using BBs have been proposed [54–56] and papers such as [57, 58] propose the cooperation of BBs in order to create end-to-end services. However, in the cases of re-rerouting or network failures, the flows already admitted in the network must be re-examined and this can take a long time and block other requests.

Traffic Engineering

Internet links are often congested due to traffic peaks. In order to tackle these issues that affect the QoS, it is possible to apply Traffic Engineering (TE) policies to the traffic. The basic idea of the TE paradigm is to establish specific paths for specific traffic types. A simple explanation of the TE objectives is given in [4] : "*to put the data traffic where the network bandwidth is available*". Generally, the TE is defined as *"the aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks"* [59].

MPLS (MultiProtocol Label Switching) and MPLS-TE (MPLS Traffic Engineering) are the most well-known protocols proposed by the IETF enabling the establishment of specific paths for different traffic types. MPLS is based on *label switching* and has been designed to facilitate the IP routing by avoiding complex lookups in the routing tables. Within an MPLS network or domain, paths are named Label Switch Path (LSP) and are either manually established or automatically established via a signaling protocol, such as the Label Distribution Protocol. The MPLS routers are called Label Switch Router (LSR). As an LSP is unidirectional, we distinguish the ingress LSR (iLSR) from the egress LSR (eLSR). The iLSR aims at labeling the entry traffic based on the interface it has been received on. Each packet is then forwarded through the network thanks to its label. In order to create the labels, the iLSR utilizes the Forwarding Equivalence Class (FEC) as an example, it can map a MAC address, an IP address and a TCP or UDP port to a label. Then, each LSR will use the label contained in the packets to forward it. More precisely, based on an incoming interface and a label, an LSR forwards the packet according to their forwarding table to an outgoing interface with a new label. This process is repeated until the packets have reached the eLSR which removes the MPLS header before they exit the MPLS domain.

Figure 3.3 shows an example of label switching principle in an MPLS domain. In this example the iSLR received a packet that must be sent to the eLSR named *d*. Based on this information, the iLSR marks the packet with the label 16 and forwards it to the interface 2 thanks to its MPLS forwarding table. Then, each LSR along the path performs the label switching operation as Figure 3.3 shows.

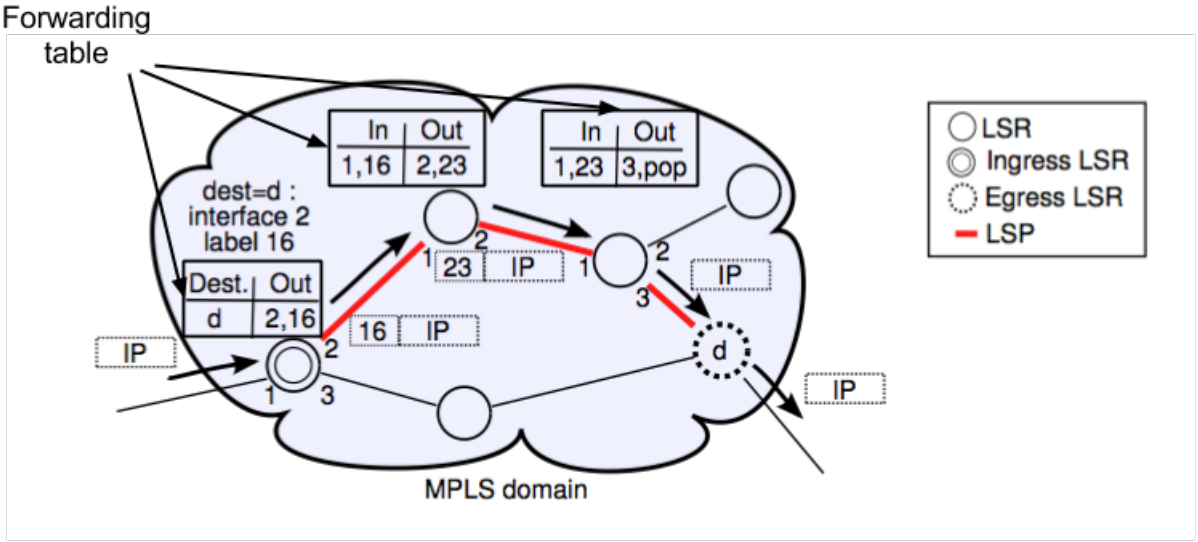


FIGURE 3.3: Label switching in an MPLS domain, source [17]

Thanks to the label switching principle of the MPLS architecture, it is possible to set up LSPs that are generally different from the IP routing. These paths are called TE-LSPs and are often established within the ASes to route traffic along the highest performing paths. RSVP-TE is a protocol enabling to reserve LSP within a domain.

For confidentiality purposes, the TE information is not exchanged outside a single domain. Moreover, a LSR needs to know the entire topology to take routing decisions thus computing an TE LSP that crosses several domains is not possible using the aforementioned method. In order to compute paths that cross domain boundaries, additional mechanisms must be added to RSVP-TE to respect the ASes' confidentiality and to preserve the scalability of routing protocol that exchanges QoS information. In [41], the authors give a summary of different mechanisms to signal inter-domain LSPs.

The first method named *contiguous inter-domain path* [60] consists in having the same MPLS policy for all the domains from the ingress node up to the egress node. More

precisely, the same identifier for the label switched path (LSP) is used. As this method needs to be implemented manually, it requires human intervention that can lead to configuration errors or large setup delays. The second method called *LSP nesting or hierarchy* [61] proposes that portions of several LSPs can overlap in a pre-established H-LSP. The main advantage of this method is that each pre-established H-LSP is fully managed by the domain administrator. The third method, called *LSP stitching* [62], is similar to LSP nesting, however in the case of LSP nesting, an S-LSP can accomodate only one LSP. It is possible to mix the three signaling methods to form hybrid signaling methods [41].

In order to compute the inter-domain TE LSPs, the IETF has proposed two techniques that we present in the next section.

3.2.6 End-to-End Paths Computation Between Several Domains

The per-domain Method

The per-domain method [5] assumes a pre-determined sequence of domains computed thanks to an "auto discovery" procedure. Then, the source domain computes a path segment from the source to an exit border router and sends it to the second domain. The latter computes a second segment to the next exit border router and concatenates the segments for sending it to the next domain. This operation is repeated until the destination domain has been reached. The concatenation of per-domain path segments yields an end-to-end QoS path. In [63], the authors introduce a crankback mechanism in order not to be limited to one sequence of ASes given by the "auto discovery" procedure in the case where the constraints are not respected. As the end-to-end path has been constructed by the concatenation of several segment paths, each of them computed from individual decision, it is often sub-optimal [19].

The Path Computation Element Technique

Bertrand et al. [6] have proposed ID-MCP, a distributed algorithm based on the PCE architecture [13] to compute end-to-end QoS paths. The source starts to send a request to the destination utilizing the PCE architecture (Figure 3.4). Once the domain containing the destination has received the request, the PCE within the domain creates a Virtual Shortest Path Tree (VSPT) of feasible non dominated paths between the destination and

its Autonomous System Border Router (ASBR). It then sends the VSPT to the PCE of the next AS in the path, using the BRPC procedure [15]. The new PCE repeats the process and merges its result to the VSPT. This process repeats itself until the VPST reaches the PCE of the domain containing the source, as Figure 3.5 shows. At the end, the source can choose the path it prefers among all those contained in the VSPT. The criteria used to select the path is out of scope of this work. ID-MCP assumes that the sequence of ASes between the source and the destination is given. This assumption constitutes its main limitation.

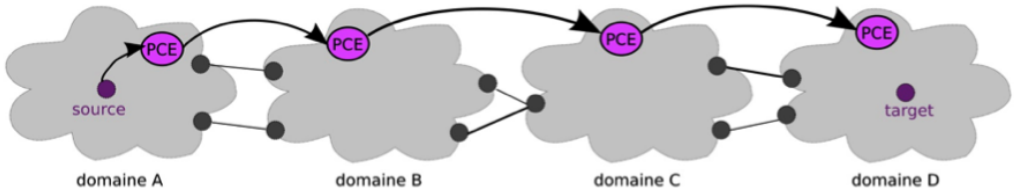


FIGURE 3.4: The source sends the request to the destination, source: [17]

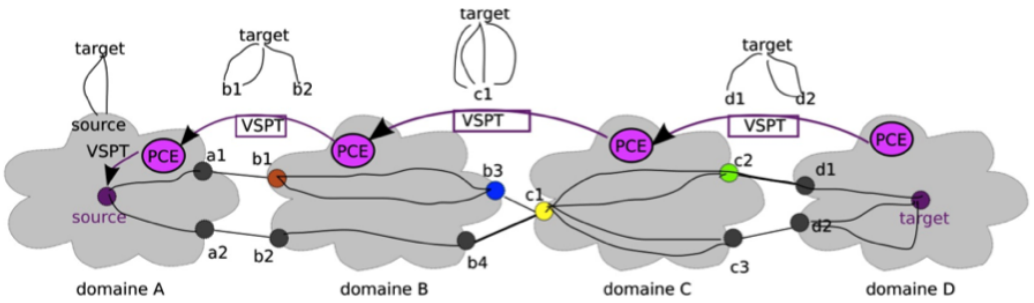


FIGURE 3.5: Construction of the VSPT, source [17]

In [20], not to be restricted to the given sequence of ASes, the authors propose to construct several VSPTs along different paths. An overlay network of PCEs is constructed, by exchanging reachability information. This overlay network can be used to find several paths between the PCEs. However the paths are not guaranteed to be edge-disjoint which can lead to redundancy of computed information. Furthermore, as several paths are explored in parallel, the authors propose a termination mechanism based on node coloration. At the end, information from the VSPTs are merged which requires complex computation and constitutes its main limitation.

3.2.7 Negotiation of Inter-providers SLAs

Apart from technical perspectives presented in the previous section to provide end-to-end QoS, it is possible to address the problem of providing QoS guarantees across several domains from a contractual point of view. Assuming that the ASes are ready to publish SLAs between end-points, the problem of offering e2e QoS guarantees can be addressed by combining several SLAs, one for each AS. The main difficulty of this method lies in finding an appropriate chain of SLAs that can guarantee strict QoS for the path requested. A brute-force search that would list all of the possible combinations of the SLAs between a source a destination is impossible at the scale of the Internet.

In [64], the authors assumes that each domain is ready to publish a set of SLAs that specifies what guarantees it can offer between an entry and an exit point. Then, thanks to a distributed algorithm that creates a VSPT resulting from the concatenation of the SLAs along a given path, it is possible to construct an end-to-end path with QoS guarantees. Note that the path is given by the underlying mechanism, in this case BGP. As the BGP paths are not QoS-aware, the QoS requirements are less likely to be meet. We can note that this work is similar to ID-MCP [17] in the sense that both construct a VSPT between the source and the destination.

On the contrary, in [65], a centralized architecture is proposed in which the ASes publish SLAs in a Quality Broker. Whenever the Quality Broker receives a request from a customer, it is responsible for building an appropriate chain of SLAs named "super SLA" respecting the constraints requested. Then, the Quality Broker proposes the super SLA to the customer and if the latter agrees with it, the service can start.

In [66], the authors propose a cascading model in which the source is responsible of negotiating and building an appropriate chain of SLAs with other domains. The domains with which the source negotiates are found thanks to the IP routing tables. As a negotiation process is involved, the SLAs are a better match for the customer requirements.

As already mentioned in Section 2.1.1, several European projects such as TEQUILLA [10], MESCAL [11] and ETICS [12] have worked on the negotiation of SLAs for providing end-to-end QoS. As well known Internet providers such as Orange or Deutsch

Telekom are involved in these projects, business issues take an important place, especially the issues for respecting the confidentiality and the autonomy of the ASes. Our work is in part inspired by these projects.

3.3 Multi-Constraint Problem

3.3.1 Introduction

Finding paths across several domains with specific QoS guarantees is a constrained routing problem. Generally speaking, it consists in finding paths satisfying multiple constraints. It is a well-known problem, called the Multi-Constraint Path (MCP) problem with applications in many areas. For instance, in the telecommunication area, one is often interested in finding a path with a maximum delay for a maximum cost the goal of this section is to describe the MCP problem. In Section 3.3.2, we start by defining the general MCP problem as well as the Multi-Constrained Optimal Path (MCOP) problem, the problem of finding feasible non-dominated paths. A usual approach to solve a multi-constraint optimization problem such as the MCOP is to transform it into a mono-constraint optimization problem. In Section 3.3.3, we describe how to apply this method and what are the main issues that arise.

Then, we present existing solutions to the MCP problem. These solutions, either exact algorithms, heuristics or approximations deal with the worst-case complexity of the MCP problem, ignoring the inter-domain communications requirements (discussed in Section 3.2.4). Thus, they are often not applicable for solving the inter-domain problem.

3.3.2 Mathematical Formulation

Due to the ever increasing demand for Internet-based multimedia applications, the literature has well investigated the MCP problem. Although the QoS applications are numerous, implying plenty of QoS needs, the literature often restricts the problem to two constraints. In this section, we describe the *general MCP problem* meaning that there is no restrictions on the number of constraints. Wang and Crowcroft have proved that the general MCP problem is NP-complete when the number of additives metrics is greater than one [7].

Notation	Meaning
$\mathbb{N}$	Set of natural integers
$\mathbb{R}$	Set of real numbers
$G(V, E)$	A graph
V	Set of vertices
E	Set of edges
K	Number of additives metrics
$W(e)$	Weights of a edge
$W_i(e)$	The i-th weight of a edge
s	Source
t	Target
p	A path
$W(p)$	Weights of a path
$W_i(p)$	The i-th weight of a path
$\overrightarrow{L}$	Vector of constraints
L_i	The i-th constraint

TABLE 3.1: Summary of notations

The literature often uses a graph to represent the Internet topology in the context of the MCP problem. Let $G(V, E)$ be a graph where V is a set of vertices representing the ASes and E is a set of edges representing the inter-domain links. The number of QoS attributes (e.g., delay or hop count) is denoted by K. Each edge $e \in E$ is specified by a vector $W(e)$ of K positives QoS weights. The source and the destination nodes are denoted by s and t respectively. QoS weights can be classified into *additive* (e.g., delay) or *minimum(maximum)-based* (e.g., bandwidth). For additive QoS weights, the QoS value of a path is equal to the sum of the corresponding weights of the links l along that path:

$$W_k(p) = \sum_{l \in P} W_k(l), k \in K$$

For a minimum(maximum)-based QoS weight, the QoS value of a path is the minimum (maximum) link weight along that path. For minimum-based constraints it suffices to remove the links not satisfying the constraint and proceed with the resulting graph. This method is called **edge pruning** and can be executed in polynomial time [69]. Hence, we only consider additive metrics [70]. Table 3.1 gives a summary of the notations used in this thesis.

Given a **connection request** specifying a source $s \in V$ and a destination (target) $t \in V$ as well as K constraints L_k with $k \in [1, K]$, the MCP problem consists in finding a path p that satisfies the constraints of the request such that $W_k(p) < L_k$ for all $k \in [1, K]$. Any path that fulfills the constraints of the request is said to be **feasible**. For instance, finding a path with a end-to-end delay lower than 100 millisecond with a maximum cost of fifty dollars is an MCP problem.

Definition 3.1 (MCP problem)**.** Given K constraints L_k, where $1 \leq k \leq K$, the problem is to find a path p from the source s to the destination t such that

$$W_k(p) = \sum_{s \to t} W_k \leq L_k \text{ for all } k \in [1, K]$$

Consider a graph composed of eight vertices as shown in Figure 3.6, where s is the source node and t the destination. Each link is associated with two additives QoS metrics: delay and cost. Assume that $L_{delay} = 15$ and $L_{cost} = 15$. p_1 = [s,a,c,t] with $W_{delay}(p_1) = 9$ and $W_{cost}(p_1) = 6$ We denote the weights of a path p, $W(p) = \frac{delay}{cost}$, for example $W(p_1) = \frac{9}{6}$. Hence, the three paths: p_1 = [s,a,c,t] with $W(p_1) = \frac{9}{6}$, p_2 = [s,a,d,t] with $W(p_2) = \frac{11}{9}$ and p_3 = [s,b,d,t] with $W(p_3) = \frac{7}{7}$ satisfy the constraints and thus are **feasible**. However, p_4 = [s,b,e,f,t] with $W(p_4) = \frac{17}{12}$ does not satisfy the constraints and is said to be **non feasible**.

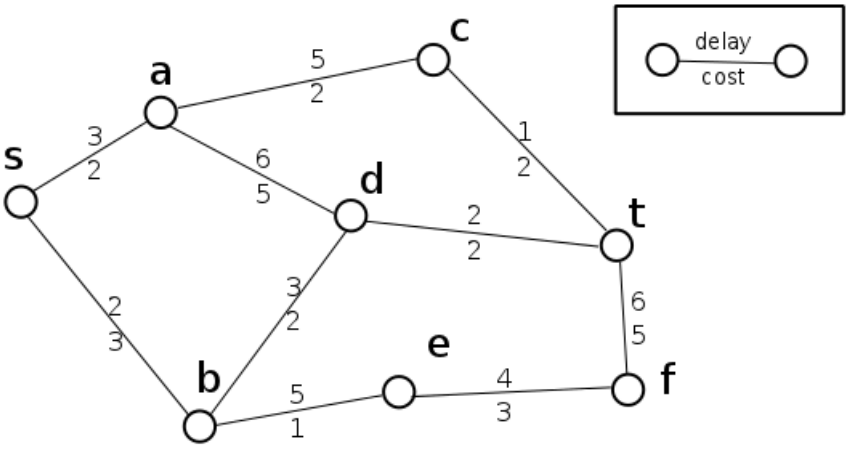


FIGURE 3.6: Exemple of the MCP problem

As we are interested in minimizing each additive constraint (e.g., delay, cost), no single optimal solution exists. For example if a path has a total delay d_1 and a total cost c_1 and a second one has a delay d_2 and a cost c_2, it is possible that $d_1 < d_2$ but that $c_1 > c_2$, in other words the first path has a smaller delay but it is more expensive. In this case it is

not possible to establish which path is the best as neither one **dominates** the other. If, instead, $d_1 \leq d_2$ and $c_1 \leq c_2$, the first path *dominates* the second one and it is, therefore, better. The notion of dominance [70] is commonly used in multi-objective optimization and allows to limit the number of "relevant" solutions: dominated paths are always worse than non-dominated ones, this is why we are interested only in non-dominated ones.

In the example above, p_1 and p_3 dominate p_2. However, we can notice that neither p_1 dominates p_3, nor p_3 dominates p_1. We say that p_1 and p_3 are both non-dominated paths.

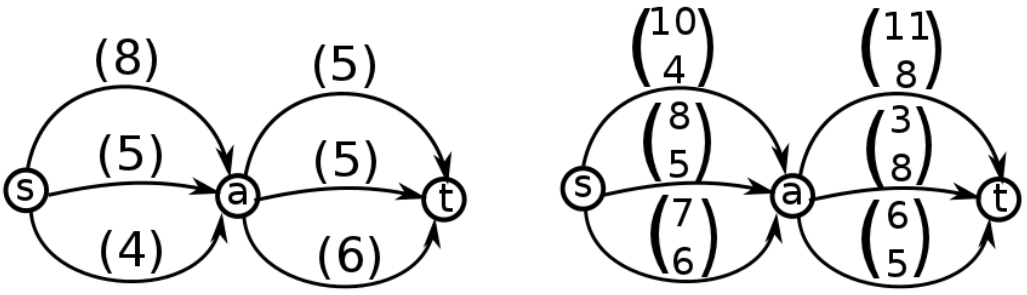
The problem of finding end-to-end feasible non-dominated paths is called MCOP and is defined as follow:

Definition 3.2 (MCOP problem)**.** Given K constraints L_k, where $1 \leq k \leq K$, the problem is to find a path p from the source s to the destination t such that

- $W_k(p) = \sum_{s \to t} W_k \leq L_k$ for all $k \in [1, K]$
- $W_k(p) \leq W_k(p')$ for all $k \in [1, K]$ except for at least one j for which $W_j(p) < W_j(p')$

3.3.2.1 NP-completeness of the MCP problem

As previously mentioned, between two paths, it is impossible to establish which path is the best when they are both non-dominated. This statement is one of the principal causes for the NP-completeness of the MCP problem. Indeed, let's take the example shown by Figure 3.7 in which several paths are possible between the source node s and an intermediate node a as well as between node a and destination node t. In Figure 3.7(A), only one weight is associated to the links. From node s to node a, $W(p_1) = 8 > W(p_2) = 5 > W(p_3) = 4$. Thus, since is is possible to establish an order between them, it is possible to affirm which one is the best path, which is p_1 in this case. Hence, in order to find the best path between node s and node t, it suffices to save path p_1 in node a. However, in Figure 3.7(B) where two weights are associated to each link, it is impossible to directly affirm which path will lead to the best end-to-end path. As a consequence, the three paths must be considered during the MCP computation operations which drastically affects the computational complexity.



(A) One weight associated per link (B) Two weights associated per link

FIGURE 3.7: *NP*-completeness of the MCP problem

3.3.3 The Need for Novel Solutions

The MCP problem has been largely studied in the literature. In [71], the authors give an overview of the most relevant algorithms to solve the MCP problem while in [72] they focus on evaluating the performance of these algorithms. In this section, we present these algorithms that are either exact, heuristics or approximations and explain why they are inapplicable at the inter-domain level.

3.3.3.1 Shortest Path Algorithms

As mentioned in Section 3.3.1, one often tackles the computational complexity of the MCP problem by transforming it from an multi-objective optimization problem into a single-objective problem. In so doing, it is possible to use a polynomial algorithm to find a shortest path, such as Dijkstra or Bellman-Ford to solve the single-objective problem. For instance, a solution proposed in [73] is to compute the shortest path following the first metric W_1 in the hope that the value of the other metrics $W_2, ..., W_k$ satisfy the constraints. If no feasible path is found for the first metric, the search is repeated with the second metric until a feasible path is found or all QoS measures are examined.

In [74], Jaffe uses the following linear path length to assign a composite weight to every link:

$$W(e) = d_1 \times W_1(e) + d_2 \times W_2(e) \text{ with } d_1 \text{ and } d_2 \text{ two positives multipliers.}$$

Then, Jaffe uses the Dijkstra algorithm to find the shortest path by optimizing the path-length function. Other work [75] proposes to extend the Jaffe's algorithm to an arbitrary number of constraints. Using a linear path-length function defined in its general form in 3.3 suffers from a severe drawback: the shortest path is not necessarily feasible.

Definition 3.3 (Linear Path-Length Function)**.** $W(p) = \sum_{i=1}^{K}(\frac{W_i(P)}{L_i})$

As an example, consider the linear path length $W(p) = W_1(p) + W_2(p)$ and $L = [10, 10]$. Now assume two paths p_1 and p_2 with the associated weights $\frac{11}{2}$ and $\frac{8}{8}$ respectively. $W(p_1) = 11 + 2 = 13$ and $W(p_2) = 8 + 8 = 16$. As $W(p_1) < W(p_2)$, p_1 is selected. However, $W_1(p_1) = 11$ is greater than $L_1 = 10$ meaning that the shortest path is not feasible whereas p_2 is.

3.3.3.2 Exact MCP Algorithms

An exact algorithm for the MCP problem finds a set of feasible non-dominated paths if such a set exists. Typically, the optimal solution can only be found by enumerating all the possible paths. This method is called **brute-search** force. It is clear that the larger the graph, the longer it takes to enumerate all the paths. As a consequence, this method is utilized on small graphs and does not scale.

In [76], the authors propose an exact algorithm called SAMCRA to solve the MCP problem. Basically, SAMCRA implements an extension of the Dijkstra algorithm to find a end-to-end QoS paths. SAMCRA performs the same operation as Dijkstra but as it has to deal with several constraints, it has to store all the feasible non-dominated (sub)-paths per vertex to find an exact solution. The feasibility of the sub-paths is computed thanks to following non-linear path-length function:

Definition 3.4 (Non-linear Path-Length Function)**.** $l_\infty(P) = \max_{1 \leq i \leq m} \left[\frac{w_i(P)}{L_i}\right]$

Furthermore, only the non-dominated sub-paths need to be stored, as a dominated sub-path is necessarily worse than a non-dominated one which substantially reduces the calculations. As SAMCRA implements an extension of the Dijkstra algorithm, each node has to know the entire graph. As a consequence, SAMCRA can not be used directly at the inter-domain level since it would break the confidentiality property of inter-domain routing communications. Apart from the confidentiality property, any strategy requiring

to know the entire topology to find end-to-end path does not scale up to the size of the Internet. Therefore, using link-state information for solving the MCP problem is nor promising nor sustainable.

The A*prune algorithm described in [77] focuses on finding not only one but k shortest paths. Starting from the source and by examining all its neighbors, A* prune extends and prunes the paths in a Dijkstra-like fashion until k feasible paths are found. In order to compute the path's length, A* uses linear path-length function defined in Definition3.3. The running time of A* prune can be exponential in the worst-case, i.e., A* prune never does the prune operation.

3.3.3.3 Fast MCP Algorithms

An approach often used to deal with the NP-completeness of the MCP problem is to quantize the metrics. It means that instead of taking real values, the links weights take discrete and bounded values. According to lemma 3.5 [17], if two intermediate paths have the same weights, then, an MCP algorithm can memorize only one of the path without losing the guarantee to find an optimal end-to-end path. As a consequence, the combination of the scaling method and lemma 3.5 can drastically reduce the number of paths saved, i.e., instead of saving many non dominated paths that can take an infinite number of values, only a bounded number of paths has to be considered.

Lemma 3.5 (identical weights)**.** *If two intermediate paths p_1 and p_2 have the same weights, i.e., $W(p_1) = W(p_2)$, then an MCP algorithm can memorize only one of these intermediate paths without losing the guarantee to find an optimal end-to-end path*

Narhstedt and Chen in [78] use the aforementioned method to solve the MCP problem with two constraints such that $W_1(p) \leq c_1$ and $W_2(p) \leq c_2$. They start to scale and bound the second constraint (delay) so that the delay parameter of each edge is approximated by a bounded integer i.e., $W_2(e) = \frac{W_2(e)\times x}{c_2}$ where x is a constant defined by the algorithm. Then, they use an Extended Dijkstra algorithm called EDSP and a Bellman-Ford algorithm called EBF to solve the MCP problem. With EDSP (resp EBF) they find a solution in $\theta(x^2V|^2)$ times (resp $\theta(x|V|Z|)$)

Yuan [79] proposes a heuristic called limited path heuristic (LPH), based on the Bellman Ford algorithm. It uses both non-dominance and stores at most k paths per node,

similarly to what TAMCRA [80] does. The main shortcoming of LPH is that it stores k arbitrary paths and checks the feasibility of the paths only when the destination t is reached.

In [81] and [82] the authors improve the performance of the Yuan's algorithm by minimizing the time complexity of the path search phase in $\theta(T\frac{|V|^2\times B}{\epsilon}^{K-1})$ where B is the maximum value of the cost metric of each link, T the number of bounded metrics and $\theta((|V||E|loglogloglog(|E|)+|E|\frac{|E|}{\epsilon}^{K-1})$ respectively.

In [80], the authors propose a heuristic called TAMCRA, which is an alternative to SAMCRA to bound the worst-case complexity of its brute-force search. TAMCRA implements a k-shortest path algorithm, i.e., it memorizes the k shortest paths per node during the path computation process instead of storing all the feasible non-dominated sub-paths as SAMCRA does. Hence, it reduces the complexity of the MCP computation. TAMCRA uses the non-linear path function defined in Definition3.4 to compute the length of the paths.

Yuan [79] proposes to stores k arbitrary paths. It checks the feasibility of the paths only when the destination t is reached.

In [83], the authors describe a randomized heuristic algorithm to solve the MCP problem. It consists of two parts: (1) pruning all the links that cannot be on any feasible path, and (2) performing a randomized heuristic search to find a feasible path, if one exists. In the first part, each node u computes the shortest path from s to u and from u to t for all QoS weights. Then, in the path search phase, the algorithm uses a modified Breadth-First Search method to find a feasible path. Rather than systematically discovering every node, as it is done with Breadth First Search (BFS), the algorithm randomly chooses a node and tries to discover its neighbors. The worst-case computational complexity of the algorithm is $\theta(|V|^2)$.

Table 3.2 gives a summary of the most prominent solutions of the MCP problem as well as their properties. Column 2 gives the type of the algorithms and their worst-case time complexity is given in column 3. $|V|$ and $|E|$ correspond to the number of nodes and edges respectively in the network. K is the number of constraints. α and α_{max} are parameters for SAMCRA and TAMCRA respectively and ϵ is an approximation parameter that reflects how far the solution is from the optimal one.

Algorithm	Category	Worst-Case Time Complexity	Reference
SAMCRA	Exact	$\theta(\alpha\|V\|log(\alpha\|V\|) + \alpha^2 K\|E\|)$	[70]
A* prune	Exact	$\theta(\|V\|!\,(K + \|V\| + \|V\|log\|V\|))$	[77]
Chen & Nahrstedt	Approximation	$\theta((\|E\| + \|V\|log(\|V\|)\frac{\|V\|}{\epsilon})$	[78]
Yuan	Approximation	$\theta((\|V\|\|E\|(\frac{\|V\|}{\epsilon}^{K-1})$	[79]
TAMCRA	Heuristic	$\theta(\alpha_{max}\|V\|log(\alpha_{max}\|V\|) + \alpha^2_{max} K\|E\|)$	[76]
Jaffe	Heuristic	$\theta(\|V\|log\|V\| + K\|E\|)$	[74]

TABLE 3.2: Summary of algorithms for the MCP problem

The main limitations of existing solutions are that they require either centralized computation or links state information. For instance, Yuan and SAMCRA assumes centralized computations while TAMCRA and Chen & Nahrstedt postulate that each node has to maintain information about the complete network.

3.3.4 Our Approach to the MCP Problem

Any algorithm intended to work at the inter-domain level must respect the inter-domain communications properties. At the time of writing this thesis, it is not conceivable for the operators to reconsider their autonomy. Therefore, an operator that delegates the control over its routing to another operator is not a viable solution for computing end-to-end QoS paths. Thus, any centralized architecture is not appropriated. A distributed solution for which each operator has the control of its network is more suitable.

Due to the competition between operators, ISPs are reluctant to divulge confidential information such as their internal topology. This challenge introduces visibility limitations for every domain. The first consequence is that it is not possible to use link-state routing, as this requires a complete view of the network. Second, for the sake of autonomy, we cannot make any assumption about the policies of each domain and thus about the intra-domain paths. As a consequence, the computation of inter-domain QoS paths cannot use any information about the internal topology of each domain, nor it can force a domain to implement a specific policy.

As the PCE architecture appears to be a solid foundation for end-to-end QoS in the Internet, the goal of this dissertation is to enhance it by addressing one of its main

limitations, namely the fact that the sequence of ASes used to satisfy a request is supposed to be known in advance. We propose two algorithms that can compute end-to-end paths satisfying given QoS constraints. Both algorithms respect the confidentiality and autonomy of each AS and are scalable.

3.4 Conclusion

In this chapter, we presented the organization of the Internet in order to show that its nature is an obstacle to enable end-to-end QoS. Then, we have shown that the existing QoS frameworks can only be applied within a domain in order to preserver the confidentiality and autonomy of each AS but also because of limited scalability.

In the second part of this chapter, we introduced formal models for finding end-to-end QoS paths in the Internet. We presented the most relevant solutions found in the literature. Due to its NP-completeness, most of these solutions focus on minimizing the worst-case complexity of the MCP algorithms without often taking into consideration the inter-ASes communications properties. Thus their utilization are restricted to the inter-domain level.

Chapter 4

Internet-like Graphs

4.1 Introduction

In order to use simulations to evaluate the performance of the solutions proposed in this dissertation, we need to use realistic Internet-like graphs of different sizes. In this chapter we give an overview of the state of the art on this subject.

The Internet topology can be modeled at different levels depending on what we are interested in simulating. The router-level is, for instance, preferable for studying the impact of router and link failures whereas the AS-level is more appropriate to the study the inter-domain routing issues. As the latter is the main topic of this dissertation, we focus on a generator that produces Internet-like synthetic topologies at the AS-level.

In order to describe and understand what an Internet-like graph is, one needs to know how the Internet topology is estimated and which metrics are used to characterize it. Therefore, we start by describing the three main methods used to infer the Internet topology in Section 4.2. Then, in Section 4.3, we present a comparison between five synthetic Internet-like topology generators and discuss the one that we have chosen for our simulations.

4.2 Inferring the Internet Topology

In this section, we present three sources of data commonly used to infer the Internet topology: CAIDA, Routviews and WHOIS. For each one, we describe the data collected and the method used to infer the Internet topology.

4.2.1 CAIDA

The Center for Applied Internet Data Analysis (CAIDA)[1] is *"a collaboration involving commercial organizations, government and research sectors in order to investigate practical and theoretical aspects of the Internet"*. One of its main topic of research is *"measuring, analyzing, modeling and visualizing the Internet topology"*. Figure 4.1 shows the different steps performed by CAIDA in order to infer an AS-level topology of the Internet. We now detail these different steps.

In 1998, CAIDA has developed a tool named Skitter to actively probe the Internet in order to collect data to infer its topology. Several monitors are scattered all over the Internet, each one periodically measuring the forward IP Paths to a list of destinations (/24 prefixes).

The skitter infrastructure has been replaced in 2008 by a more powerful one: Archipelago (Ark). Nowadays, the Ark infrastructure is composed of 60 monitors deployed in 30 countries on the six continents. In 48 hours the set of Arks monitors is able to probe the entire IPv4 address space thanks to the scamper tool. Scamper is an active measurement tool relying on Traceroute and Ping. After 48 hours of probing the Internet, CAIDA has a list of IP paths.

[1] http://www.caida.org/home/

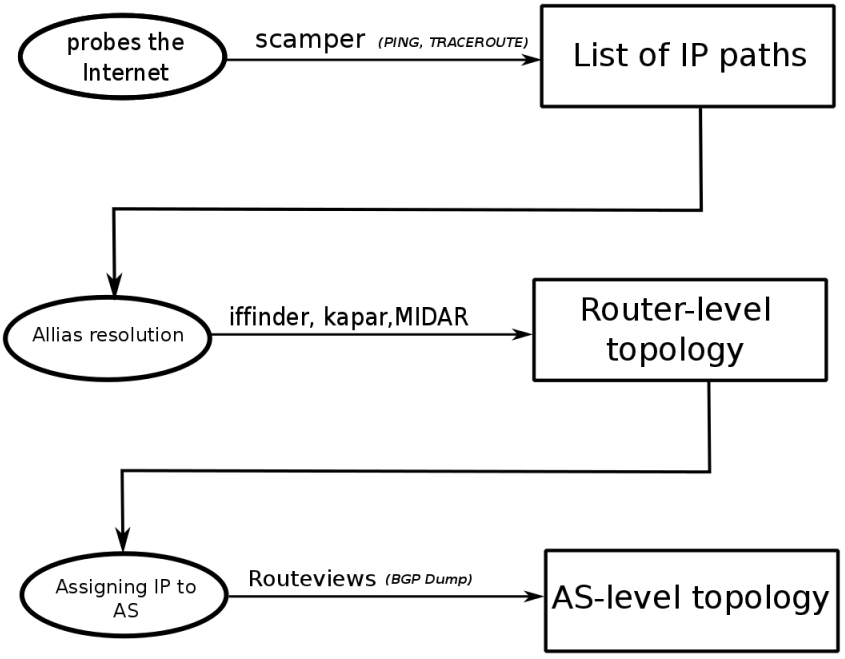


FIGURE 4.1: CAIDA work-flow to generate the AS-level topology of the Internet

From the Traceroute outputs, CAIDA infers a router-level topology thanks to an alias resolution technique. An alias resolution consists in identifying in the Traceroute outputs IP addresses belonging to the same router and the set of two or more routers on the same IP link. Various alias resolution techniques exist [84–90], but as the responses obtain via Traceroute can contain irregularities such as non-responsive hops, loops, private or bogon addresses [2], each of them presents some limitations [90].

A good survey of the accuracy and performances of theses techniques is given by [89]. CAIDA uses three tools in conjunction, iffinder [3], MIDAR [4] and kapar [5] in order to perform the alias resolution. While iffinder uses active UDP probing to discover whether two interfaces belong to the same router [84, 85], kapar identifies IP Aliases using subnets properties [88] and MIDAR uses IP ID values used in the routers [90].

The next, and last, step consists in assigning IP addresses to an AS to infer the AS-level topology from the router-level topology. To do so, CAIDA uses the BGP table dumps from the RouteViews project [6] to map IP addresses space into the AS space.

[2] http://www.team-cymru.org/Services/Bogons/bogon-bn-nonagg.txt
[3] http://www.caida.org/tools/measurement/iffinder/
[4] http://www.caida.org/tools/measurement/kapar/
[5] http://www.caida.org/tools/measurement/midar/
[6] http://www.routeviews.org/

Irregularities such as IP prefixes originating from multiple ASes can occur at this step. CAIDA handles it by selecting the AS the most frequently seen in the BGP tables as the origin AS. Once the IP addresses have been mapped to an the AS space, CAIDA uses a technique described in [91] to infer the Internet AS-level topology.

4.2.2 RouteViews

The RouteViews project was originally *"motivated by interest on the part of operators in determining how the global routing system viewed their prefixes and/or AS space"* [7]. RouteViews uses 7 collectors, 5 of which are located in the USA, 1 in the UK and 1 in Japan to collect BGP routing tables. Each collector has a number of globally placed peers (or vantage points) that collect BGP messages exchanged between ASes. Then, by using the collected BGP messages e.g., BGP update and withdraw messages or the static snapshots of BGP tables, it is possible to infer an AS-level graph of the Internet. In [91], the authors give two methods to derive an AS-level graph using both types of data.

Since the BGP messages exchanged between the ASes and the BGP tables reflect only a small part of the real map of the Internet, using the BGP data to infer its topology leads to an incomplete map [91].

4.2.3 WHOIS

WHOIS [92] is a protocol used for querying a collection of databases that store a wide range of information such as AS number, IP address block or domain name. Internet Routing Registries (IRR) is one of these databases in order *"to help debug, configure, and engineer Internet routing and addressing"* [8].

Figure 4.2 shows an example of information contained in the IRR database. From this IRR record, it is possible to extract three ASes: AS 3303, AS 701 and AS 1239 and two edges: AS 3303 - AS 701 and AS 3303 - AS 1239. Hence, it is possible to infer an AS-level topology of the Internet from theses records.

[7] http://www.routeviews.org/
[8] http://www.irr.net/

aut-num:	AS 3303
import:	from AS 701
export:	to AS 1239

FIGURE 4.2: Example of IRR

IRR information are manually maintained by the ASes themselves. Therefore, AS links derived from IRR could be outdated or incomplete [91]. However, the up-to-date IRR entries provide significant and unique information that could not be obtained from any other source. According to [93], with careful processing of the data, it is possible to extract a non-trivial amount of correct and useful information. As the European WHOIS database contains the most reliable topological information [91, 94], only the European Internet topology is inferred thanks to the IRR records [93, 95].

4.2.4 Concluding Remarks on Estimating the Internet Topology

In this part, we focused on the three main source of collected data used to infer the AS-level map of the Internet. In [91], the authors discuss the advantages and disadvantages of each of data. To summarize, the CAIDA dataset, based on the Traceroute tool, suffers from the non-responsive hops as well as irregularities in the Traceroute outputs. It also fails to receive the replies from small ASes [91]. The AS-level graph inferred with the Routview information depends on the BGP routing tables exchanged. As not all peering ASes advertise all their peering relationships, the inferred topology tends to miss these unadvertised links. Moreover, misconfigurations, e.g., announcement of prefixes not owned by an AS, are some of the other causes of errors. The WHOIS database is manually maintained by the ASes and it is most likely to contain inaccurate and incomplete data [93]. In [91], it is noted that *"some ISPs entering inaccurate information in the WHOIS database to increase their "importance" in the Internet hierarchy"*. They conclude that deciding which dataset most closely match actual Internet topology remains an open question.

4.3 Generating Internet-like Graphs

As shown in the previous section, obtaining the exact topology of the Internet is a complicated and unfinished task. Moreover, the complete Internet map is extremely large (tens of thousands), leading to potentially time-consuming and complex simulations, if used in its entirety. A smaller but Internet-like graph is often preferable for running simulations. Hence, a synthetic Internet-like topology generator is an essential tool for running realistic simulations in order to assess the performances of new algorithms for end-to-end QoS guarantees. In this section, we present the most widely used techniques to synthetically generate these graphs. We then focus on a study [96] in which the performance of five generators are compared for generating the Internet map. We rely on this study in order to chose the generator that we use for our simulations.

Figure 4.3 shows the relationship between the observed AS-level topology inferred with CAIDA, Routeviews or WHOIS and a synthetic topology. Relying on a set of topological metrics such as the number of nodes or the number of links, it is possible to extract properties from the observed AS-level topology. Based on these properties, a generator aims at producing synthetic topologies with equivalent properties. Obviously, one of the limitations of such an approach is that the generated graphs are representative of the nature of the Internet only as long as the estimated topology is accurate.

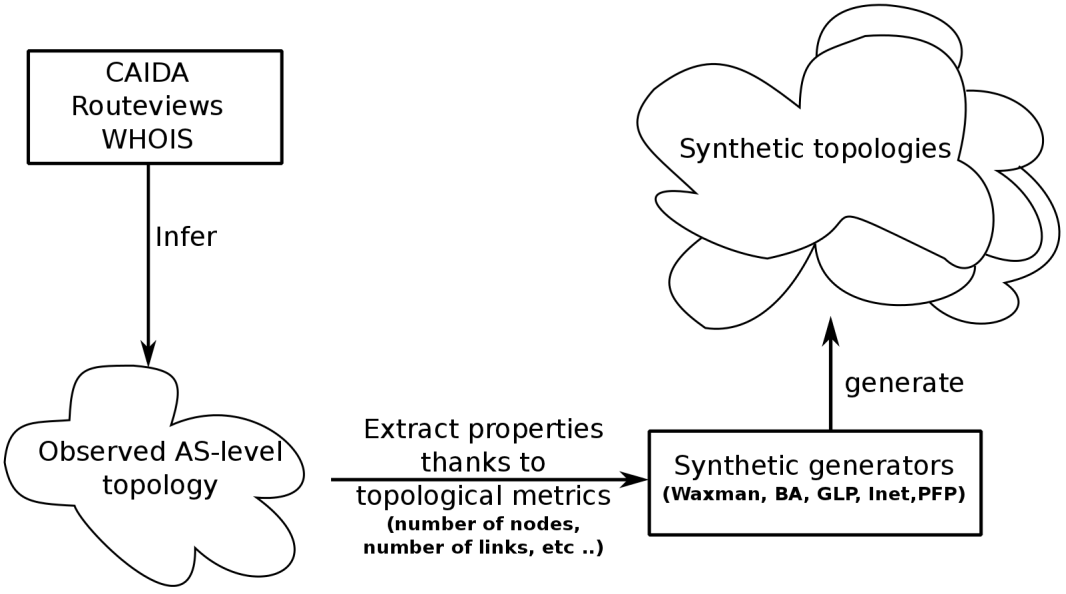


FIGURE 4.3: Relationship between observed and synthetic topology.

As previously mentioned, the study in [96] compares the performances five generators to generate Internet-like topologies. We now give the details of this study. We start by describing the five generators. Then, we present the Internet topologies inferred thanks to the methods described in Section 4.2 as well as the set of metrics used for the comparison.

4.3.1 Graph Generators

In this section, we describe the five generators compared in [96]. Note that for the rest of this chapter, we use the name of the generator to refer to the corresponding model as well.

Waxman

The first generator is based on the well-known Waxman model [97]. The Waxman model has been widely used to generate topologies [98] in computer science [99, 100] as well as in medicine [101, 102] or physics [103]. It is a random geographical model in which the nodes are uniformly distributed on the plane and edges are added according to probabilities that depend on the distances between the nodes. The probability to have an edge between node i and node j is given by the following formula: $P(i,j) = a \exp^{\frac{-d}{bL}}$ where $a > 0$, $b \leq 1$, d is the Euclidean distance between node i and j and L is the maximum Euclidean distance between any two nodes. This model does not match the Internet map because the Internet is known not to be a random network [104] and, as we will see with the next generators, nodes do not connect with each other according to their geographical distance.

Barabasi-Albert

The second generator is based on the Barabasi-Albert (BA) model [105]. The basic assumption of this model is that nodes are not node connected each other according to their geographical distance, the BA model is the first model that introduces the "preferential attachment" principle. This principle assumes that when a node enters in the network, it is more likely to connect with high-degree nodes than low-degree nodes. Mathematically, when a node i joins the graph, the probability that it connects to a node j already in the network is given by: $P(i,j) = \frac{d_j}{\sum_{k \in V}(d_k)}$ where d_j is the degree of node j, V is the set of nodes already in the graph and $\sum_{k \in V}(d_k)$ is the sum of degrees

of all nodes already in the graph. Note that, we use the same notation for the three other models.

Also called "the rich-get-richer" principle, the preferential attachment principle leads to a power law distribution of the node's degree. In mathematical terms, the logarithm of the degree distribution p_k is a linear function of degree k thus:

Definition 4.1 (Power law distribution)**.**

$$\ln p_k = -\alpha \ln k + c, \text{ where } \alpha \text{ and } c \text{ are constants.}$$

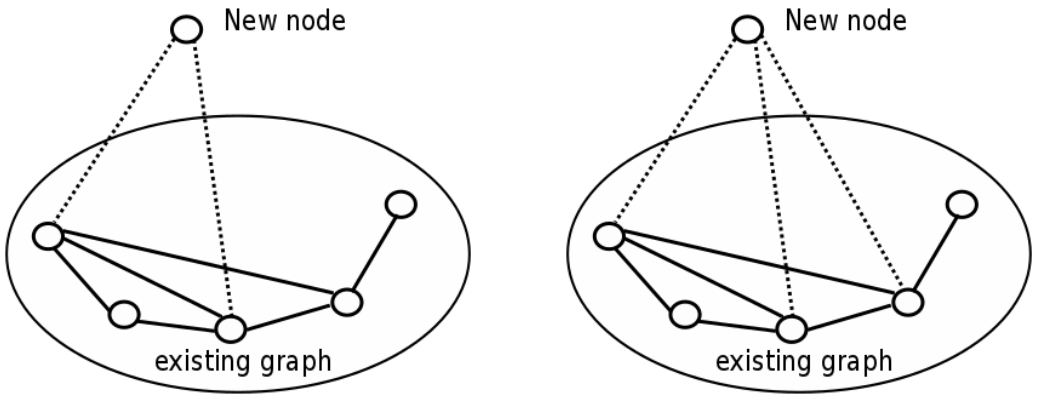
and by taking the exponential of both sides, we can write

Definition 4.2 (Power law distribution)**.**

$$p_k = Ck^{-\alpha}, \text{ where } C = e^c \text{ is another constant.}$$

α is known as the *exponent* of the power law. Often, it is enough to define α [106–108] to characterize the Internet graph.

The BA model takes as input parameter the number of nodes l to which a node joining the graph has to be connected to and the number of total nodes in the graph. Figure 4.4(A) (resp. Figure 4.4(B)) shows an example of the BA growth with $l = 2$ (resp $l = 3$). As the BA model does not capture the rich-club phenomenon [109], it is not widely used these days. The Positive Preference Feedback model is the first one to include the rich-club phenomenon.



(A) BA growth with $l = 2$ (B) BA growth with $l = 3$

FIGURE 4.4: Example of BA growth

Generalized Linear Preference

The third generator is based on the Generalized Linear Preference (GLP) model [106]. GLP uses a slightly different preferential attachment than the one used in the BA model. When a node i joins the graph, the probability that it connects to a node j already in the network is given by: $P(i,j) = \frac{d_j - \beta}{\sum_{k \in V}(d_k - \beta)}$ where $\beta \in (-\infty, 1)$. The parameter β indicates the preference for a new node to be connected to the highest degree nodes. The smaller the value of β is, the lower the preference given to high degree nodes. GLP matches better than BA the degree distribution, the clustering coefficient and the paths length observed in Internet [110].

Inet

The fourth generator is based on the Inet model [98]. Inet relies on measurements of the observed Internet topology. The authors have extracted properties from 51 topologies between November 1997 and February 2002. They focus especially on respecting the power law distribution of the node degrees. When a node i joins the graph, the probability that it connects to a node j already in the network is given by:

$$P(i,j) = \frac{w_i^j}{\sum_{k \in V}(w_i^k)}$$

where $w_i^j = MAX\left(1, \sqrt{\left(\log(\frac{d_i}{d_j})\right)^2 + \left(\log(\frac{f(d_i)}{f(d_j)})\right)^2}\right) d_j$, with $f(d_i)$ the frequency of the degree d_i.

As noted in [111], *"Inet it is capable of creating networks with accurate details of degree distributions"*. However, Inet produces 25% fewer links than the observed Internet topology [111]. Inet takes as input the number of nodes with a minimum of 3037 nodes corresponding to the number of ASes in the Internet in November 1997. Moreover, one third of the nodes have a degree equal to 1 based on the measurements from Routeviews in 2002. The authors believe that Inet topologies do not represent the Internet well in terms of maximum clique size and clustering coefficient. At the same time, according to [112]: *"Inet matches the Internet AS graph very well"* and in [113] the authors affirm that *"Inet is the most used AS-level topology generator"*.

Positive Feedback Preference

The fifth generator is based on the Positive Feedback Preference (PFP) model [109]. PFP uses a non-linear preferential attachment principle: when a node i joins the graph, the probability that it connects to a node j already in the network is given by:

$$P(i,j) = \frac{d_j^{1+\lambda \log(d_j)}}{\sum_{k \in V}(d_k^{1+\lambda \log(d_k)})}$$

where λ is a parameter of the model. Moreover, for each new node to be added to the graph, new links are added between nodes already in the graph. Figure 4.5 shows the three cases implemented by the PGP model to add a new node in the graph. The probability to choose the a (resp. b, c) case is equal to p (resp. $1-p$, $1-p-q$). The authors claim that with $p = 0.3$ and $q = 0.1$, the graphs they obtain have the same ratio of nodes and links observed on the Internet.

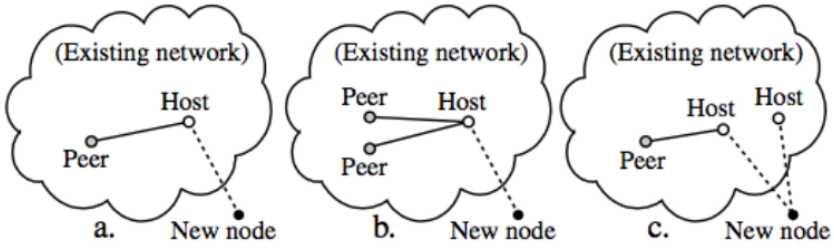


FIGURE 4.5: Different ways to add a new node used by the PFP model, source: [109]

Moreover, PFP implements the rich-club phenomenon introduced in [114]. The rich-club phenomenon translates the fact the high-degree nodes are very well connected to each other. In order to measure how well nodes are connected to each other, the authors

define a rich-club coefficient. The rich-club coefficient $\gamma(r)$ is defined as the ratio of the actual number of links over the maximum possible number of links between nodes with node rank less than r. The node rank r is the rank of a node on a list sorted in a decreasing order of node degree, and r is normalized by the total number of nodes.

The authors claim that many topological properties of the Internet such as degree distribution, rich-club connectivity, shortest path lengths, and betweenness centrality are well represented in the PFP topologies.

4.3.2 List of Metrics

In the section we describe the metrics used in [96] to characterize the properties of Internet-like graphs. Note that we assume that metrics such as number of links or the average degree do not require further explanations. Note as well that these metrics are commonly used [110, 115, 116].

- **Betweeness**: betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes [9]. Hence, the betweeness of a node is equal or superior to 0. Figure 4.6 shows an example of graph in which the red nodes have a betweeness equal to 0 and the blue nodes have the maximal betweeness.

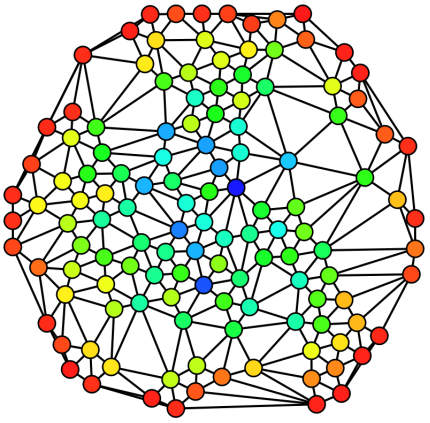


FIGURE 4.6: Exemple of betweeness, source: [10]

- **Max coreness**: the l-core of a graph is a connected sub-graph in which each node has at a least a degree equal to l. A node has coreness equal to l if it belongs to

[9] https://en.wikipedia.org/wiki/Centrality

the l-core but not to a $(l+1)$-core. In the l-core composition of a graph, we start to put all the node with a degree equal to 1 in the 1-core. Then, by recursively removing from the graph the nodes in the 1-core, we add in the 1-core the nodes having a degree equal to one. Similarly, we put in the 2-core the nodes having a degree equal to 2 after removal of the 1-core as well as, recursively those having a degree equal to 2 after removal of the former and so on. The core of a network is the l-core such that the $(l+1)$-core is empty. Figure 4.7 shows a graph with the corresponding coreness for each node. Note that the l-core of the graph is equal to 3.

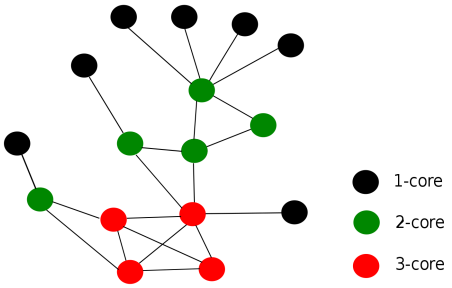


FIGURE 4.7: Exemple of coreness

- **Assortivity coefficient**: it measures the likelihood of connection of nodes of similar degrees. The assortivity coefficient is comprise between -1 and 1. -1 means that the graph is completely disassortative, 0 means that the graph is non-assortative and 1 means that the graph have perfect assortative mixing pattern. For instance, the left (resp. right) graph in Figure 4.8 is said to be disassortative (resp. assortative).

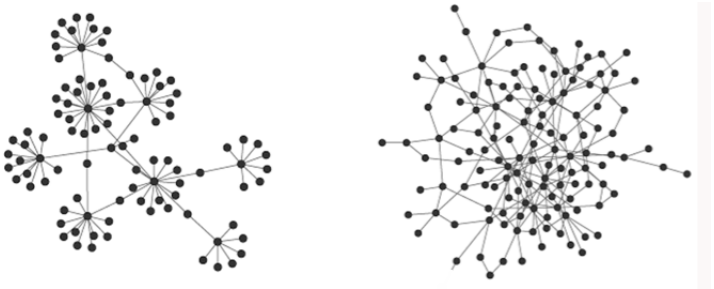


FIGURE 4.8: Exemple of assortativity, source: [117]

- **Clustering coefficient**: given a node i with k_i links, these links could be involved in at most $k_i\frac{(k_i-1)}{2}$ triangles (e.g. nodes $a \to b \to c \to a$ form a triangle). The

greater the number of triangles the greater the clustering of this node. The clustering coefficient, G, is defined as the average number of 3-cycles (i.e., triangles) divided by the total number of possible 3-cycles: $G = \frac{1}{N}\sum_{i=1}(\frac{T_i}{k_i\frac{(k_i-1)}{2}}), k_i \geq 2$ where T_i is the number of 3-cycles for node i, k_i is the degree of node i.

Figure 4.9 shows the clustering coefficient of the blue node. The red dotted links represent potential links and the thick black node are effective links.

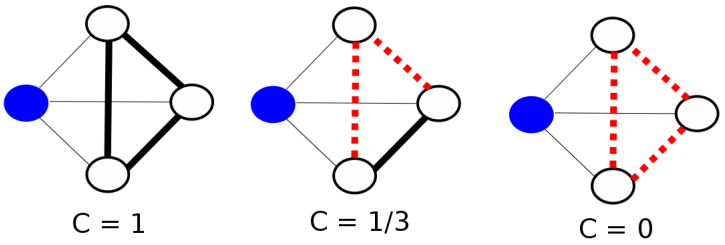


FIGURE 4.9: Example of clustering coefficient with 3 neighbors

- **closeness**: The farness of a node i is defined as the sum of its distances, defined as the length of the shortest path, to all other nodes. Its closeness is defined as the inverse of the farness [11].

Table 4.1 summarizes the comparison between the five generators. The first column corresponds to the name of Internet observed topology with which the topologies of the five generators are compared. In line 1, the Internet observed topology corresponds to the Chinese network and has 84 ASes. Note that Inet is not compared with the Chinese network since Inet does not generate topology with a number of nodes inferior to 3037. In line 2, the observed Internet topology is inferred thanks to CAIDA and has 9.204 ASes. In line 3, the observed Internet topology is inferred thanks to RouteViews and has 17,446 ASes. In line 4, the UCLA Internet topology has 28 899 ASes and is obtained from the BGP routing tables performed by the work in [118]. The other columns correspond to the metrics previously described.

In each cell of Table 4.1, the bold number corresponds to the generator that best matches the observed Internet topology. For instance, for the number of links (first column) with the Chinese topology (first line). The Chinese topology has 211 links. In this case, PFP is the best as it generates a topology with a a number of links equal to 250 which is the closest to 211.

[11] https://en.wikipedia.org/wiki/Centrality

Based on the results given by Table 4.1, we can conclude that Inet outperforms the other generators expect for the Skitter topology. Combining this with the remarks in [113] claiming that Inet is the most used AS-level topology generator and in [112] claiming that Inet matches best the Internet AS graph, we decide to chose Inet as synthetic Internet-like generator.

Finally, even if the generators like described in [119] or [120] use more recent and accurate data, they produce directed graphs that do not fit our assumptions.

Topology	Links	Avg. deg.	Max. degree	Top clique size	Max. betweenness	Max. coreness	Assort. coef.	Clust. coef.	Max. closeness
Chinese	*211*	*5.02*	*38*	*2*	*1,324*	*5*	*-0.32*	*0.188*	*<0.01*
Waxman	252	6	18	**2**	404	4	0.039	0.117	**0.506**
BA	165	3.93	19	3	**1,096**	2	-0.096	0.073	0.515
GLP	151	3.6	44	3	2,391	**5**	-0.257	**0.119**	0.643
PFP	**250**	**5.95**	**37**	10	849	9	**-0.38**	0.309	0.638
Skitter	*28,959*	*6.3*	*2,070*	*16*	*10,210,533*	*28*	*-0.23*	*0.026*	*<0.01*
Waxman	**27,612**	**6**	33	0	474,673	4	0.205	0.002	**0.264**
BA	18,405	4	190	0	5,918,226	2	-0.05	0.001	0.315
GLP	16,744	3.64	**2,411**	2	34,853,544	5	-0.089	0.003	0.496
INET	18,504	4.02	1,683	3	15,037,631	7	-0.195	0.004	0.514
PFP	27,611	**6**	3,000	**16**	**13,355,194**	**24**	**-0.244**	**0.012**	0.588
RouteViews	*40,805*	*4.7*	*2,498*	*9*	*30,171,051*	*28*	*-0.19*	*0.02*	*<0.01*
Waxman	52,336	6	35	0	1,185,687	4	0.205	0.001	**0.25**
BA	34,889	4	392	3	33,178,669	2	-0.04	0.001	0.33
GLP	31,391	3.6	4,226	4	127,547,256	6	-0.08	0.002	0.48
INET	**43,343**	**4.97**	**2,828**	**6**	**31,267,607**	14	-0.258	0.006	0.522
PFP	52,338	6	4,593	23	39,037,735	**30**	**-0.252**	**0.009**	0.564
UCLA	*116,275*	*8.05*	*4,393*	*10*	*76,882,795*	*73*	*-0.165*	*0.05*	*0.32*
Waxman	86,697	6	40	0	3,384,114	4	0.213	<0.001	0.246
BA	57,795	4	347	0	52,023,288	2	-003	<0.001	**0.3**
GLP	52,456	3.63	7391	2	371,651,147	6	-0.08	<0.001	0.486
INET	**91,052**	**6.3**	**6,537**	**12**	**88,052,316**	38	-0.3	**0.01**	0.55
PFP	86,696	6	8076	26	123,490,676	**40**	**-0.218**	**0.01**	0.57

TABLE 4.1: Comparison of AS-level dataset with synthetic topologies, source: [96]

4.4 Conclusion

The goal of this chapter is to select the generator that we are going to use for our simulations. As a generator aims to generate topologies with properties similar to the observed Internet topology, we first presented three methods used to estimate the Internet topology. Then, in order to chose a generator, we relied on a study [96] that gives a complete comparison between five generators. We describe the five generators, the observed Internet topologies as well as the metrics used to compare them. The study clearly shows that Inet outperforms the other models at generating Internet-like graphs. As a consequence, we use Inet for generating topologies in our simulations.

Chapter 5

The SANP Algorithm

5.1 Motivation

The NP-completeness of the MCP problem has led the literature to focus on bounding the worst-case complexity yielding plenty of solutions, each of them claiming appealing features compared to the others. As an example, one bounds the worst-case complexity by transforming the multi-constraint problem into a mono-constraint problem, others by considering a subset of paths, as SAMCRA does, by implementing a k-shortest path algorithm. Moreover, most of them do not respect the inter-domain communications properties such as confidentiality and autonomy of the ASes and scale poorly.

In this chapter, we introduce SANP [1], a scalable algorithm that can solve the general MCP problem respecting the confidentiality and the autonomy of the ASes by considering a subset of nodes corresponding to a sub-graph. Contrary to other existing solutions, SANP does not assume that the sequence of ASes is known, instead it builds a sub-graph around the route used by existing routing mechanisms between the source and destination.

This sub-graph is used to compute a set of feasible non-dominated paths between the source and the destination. In order to be scalable, we have implemented two heuristics to limit the size of the sub-graph. To assess SANP, we compare the paths it finds with the all the feasible non-dominated paths that exist in the graph. Our simulations show that SANP finds a reasonable number of paths that are close to the optimal solution.

[1] first presented in [8]

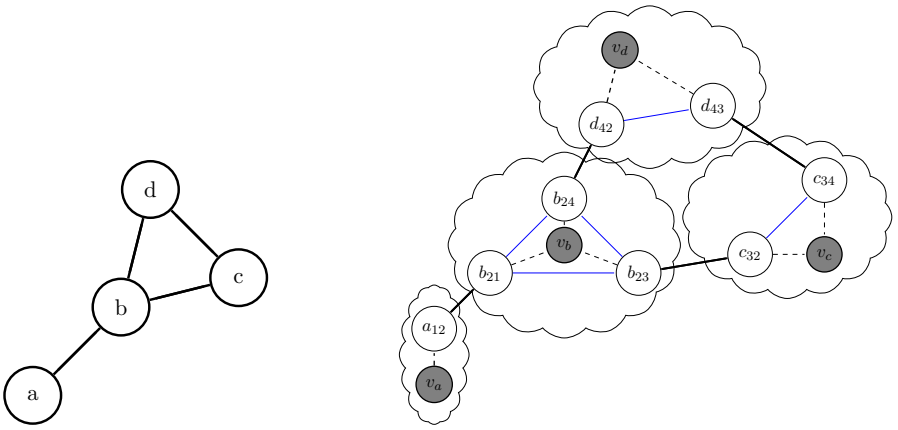
The remainder of this chapter is structured as follows. We first present our model in Section 5.2. Then, in Section 5.3, we show how SANP constructs the sub-graph in which it solves the MCP problem and the two heuristics that limit the size of the sub-graph. Finally, in Section 5.4 we present simulation results showing that the solutions found by SANP are close to the solutions found in the whole graph.

5.2 Model

The goal of the SANP algorithm is to find a set of feasible non-dominated paths between a source and a destination. Generally, they are located into two different ASes. Thus, our algorithm must respect the three properties of the inter-AS communications: confidentiality, autonomy and scalability.

We assume that the ASes are ready to offer QoS guarantees between their entry and exit points. Similarly to what network providers do today when offering SLAs with QoS guarantees for certain customers, we envisage that ASes could sell *QoS-offers* or *offers* to any other ASes. An offer would specify the QoS guarantees in terms of bandwidth, delay (and possible others metrics) and the corresponding cost between one of its entries and one of its exit points. Moreover, for the sake of autonomy, an AS would not need to announce how it implements its offers.

In order to represent the ASes and the relationships between them, we are going to use a graph $G(V, E)$ where each node $v \in V$ is an AS and E is a set of edges that represents the inter-domain links. Figure 5.1(A) gives an example of a graph with four ASes. In this graph, we need also to specify what are the QoS guarantees that each provider is willing to offer between two given nodes. In order to reduce the number of nodes in the graph and, above all, in order to limit the amount of information disclosed by each AS, we represent each AS as a complete graph connecting all the Interfaces of the AS Border Routers (ASBRI).



(A) AS-level topology.

(B) Corresponding ASBRI topology (the v_i nodes are shown in gray and the links connected them to each ASBRI are dashed lines).

FIGURE 5.1: Different representations of the same topology.

Based on the graph G, it is possible to construct a second graph where each node represents an ASBRI. A node belongs to only one AS and it is connected to all others nodes belonging to the same AS as Figure 5.1(B) shows. First, it is possible that two ASBRIs belong to the same ASBR. Second, it is important to stress that the intra-domain links are an abstract representation of the services offered by each AS and do not represent at all their internal topology. Therefore, the model respects the confidentiality property since it considers only public information such as inter-domain links and AS service offers. Note that an AS is not obliged to offer services between all its ASBRIs. By setting the QoS guarantees to 0 or ∞ on certain links, an AS indicates that no offer is implemented between the two corresponding nodes. Moreover, since two ASBRIs can belong to the same ASBR, multiple offers can be implemented between two ASBRs with different QoS guarantees. Furthermore, each node is connected to exactly one node belonging to a different AS. These edges represent the inter-domain links. On an AS-level topology generated by Inet, only one link connect two ASes. Therefore, an AS is connected only one time with another AS at the ASBRI-level.

Each edge, between two nodes belonging to the same AS, is characterized by a vector w with K weights representing the K additives QoS metrics. For the sake of simplicity, we assume that all the links are bidirectional and that the QoS metrics are the same in both directions, so that we can use an undirected graph. It is, nonetheless, trivial to extend the model to take into account asymmetric links, which can be represented using

a directed graph instead. Without loss of generality we are also assuming that all the inter-domain links have infinite capacity and all the additive metrics (e.g., delay, cost) are equal to 0. If this is not the case, these values can be added to the intra-AS links.

As we are interested in finding end-to-end paths between end-nodes, we introduce a "special" node in each AS. This node, called v_i represents a "random" end-node in AS i. It is connected to all the ASBRIs of its AS with certain QoS metrics, whose values are selected as described in section 5.4. Figure 5.1(B) shows the graph in terms of ASBRI corresponding to the AS level topology shown in Figure 5.1(A): for example, AS b in Figure 5.1(A) is connected to three other ASes, therefore it corresponds to three ASBRIs, as shown in Figure 5.1(B).

5.3 Algorithm

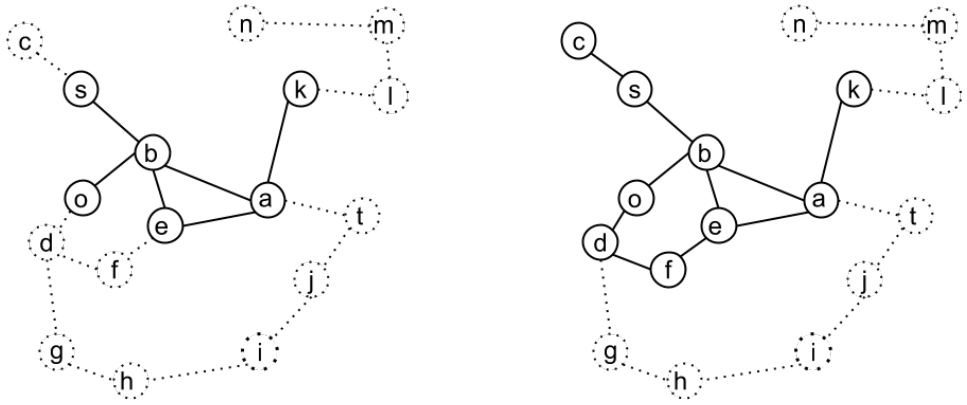
In this section we give a detailed description of the SANP (**S**ub-Graph **A**lgorithm for finding feasible **N**on dominated **P**ath) algorithm, whose goal is to compute a set of feasible non-dominated paths spanning several ASes and satisfying given QoS constraints.

In the context of inter-domain communications on the Internet, BGP is the de-facto routing protocol designed to exchange reachability information between the ASes. However, BGP stores only one path toward a destination ignoring all QoS parameters. Not to be restricted to the path given by BGP, SANP aims at exploring a region between the source and the destination. This region corresponds to a sub-graph that, once known by the source, can be used to compute a set of feasible non-dominated paths. This sub-graph that we call H is built by combining a set of neighborhoods: each node i in G knows the topology of its neighborhood N_i, that is the set of nodes V_i' and links E_i' whose distance is smaller than r_0. The larger r_0 is, the larger each neighborhood is and the larger the resulting sub-graph is.

Definition 5.1 (Neighborhood)**.** Formally, let $G = \{V, E\}$ be the AS-graph, the neighborhood of node i is the set $N_i = \{V_i' \cup i, E_i'\}$ where $V_i' = \{x | d(i,x) \leq r_0, x \in V\}$ and $E_i' = \{e | e(j,k) \in E, j \in V_i', k \in V_i'\}$ where $d(i,j)$ is the length (number of hops) of the shortest path between node i and node j, and $e(j,k)$ is the edge between nodes j and k.

These neighborhoods can be easily computed by each node using several techniques, one of them being a limited flooding of link state information. The SANP algorithm assumes that all the nodes know, and regularly update, their respective neighborhoods.

As an example, Figure 5.2(A) gives an example of a graph in which the nodes and edges in bold are in N_b with $r_0 = 1$. We can observe that the edge between node e and a is in N_b. Recall that *all* the edges between the nodes in the neighborhood are included. Both nodes e and a are in the neighborhood of node b, $d(b, e) \leq 1$ and $d(b, a) \leq 1$ then $e(e, a)$ is in N_b. Figure 5.2(B) shows the neighborhood of node b with $r_0 = 2$.



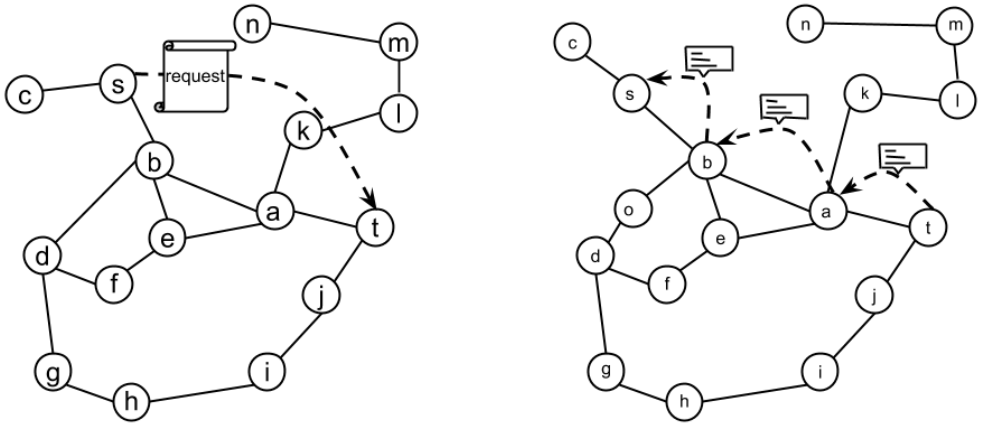
(A) N_b with $r_0 = 1$ (B) N_b with $r_0 = 2$

FIGURE 5.2

Whenever a source node s would like to establish a path with QoS guarantees with a destination node t, it sends a request addressed to the destination. The underlying network delivers this request to the destination as Figure 5.3(A) shows. Then, the destination creates a message that contains its neighborhood and sends it to the source. This message follows the path dictated by the routing scheme used in the network as Figure 5.3(B) shows. In practice, it can follow the path given by BGP. For the sake of simplicity, we consider only the *AS-Path* attribute in our simulations.

Each node along the path, nodes a, b and s in Figure 5.3(B), merges its own neighborhood with the sub-graph H, under construction, contained in the message, so that H grows at each step, until it reaches the source. A node i merges its neighborhood by adding to H all the nodes and edges in N_i that are not in H. Concretely, it starts by identifying each node that belongs to both H and its neighborhood. We can guarantee that for $r_0 \geq 1$ at least node i belongs to both N_i and H. For each of these nodes, it adds its neighbors

that do not belong to H but are in N_i and the edges between them. This operation is repeated until all the nodes in N_i are in H.



(A) request sent to the destination (B) the message follows the shortest path

FIGURE 5.3

At this point the source knows a sub-graph H composed of a set of ASes containing both itself and the destination as well as a set of edges. First, we ensure that with $r_0 \geq 1$, H is a connected graph: each neighborhood merged with H is a connected graph. Furthermore, as previously mentioned , when N_i is merged with H, at least one node belongs to both N_i and H. This guarantees that each merge operation is possible and that H is a connected graph. Second, the edges are either inter-domain links or intra-domain links with the latter corresponding to the QoS-offers. Concerning the intra-domain links, as we are considering only the extremities, namely the ASBRIs, the confidentiality of the ASes is respected. At the end, the source can use whatever algorithm it sees fit to find a set of feasible non-dominated paths in H. One possible solution, and the one we used in our simulations, is for the source to compute all the simple paths in the sub-graph between the source and the destination. As the size of the sub-graph is relatively small, such a brute-force approach can indeed be feasible.

Algorithm 1 gives the pseudo code of SANP. Once the request has arrived at the destination (line 1), this node creates a message and initializes a sub-graph with its own neighborhood (line 2). Note that only the nodes with a degree larger than 1 are merged with the sub-graph. Given that such nodes do not offer any additional diversity. In

the loop starting in line 3, each node along the path, given by the underlying routing mechanism, merges its neighborhood with the sub-graph under construction and sends it to the next hop. In line 4, SANP finds the next hop toward the source and sends the message containing H to the next hop at line 5. The merge operation between the neighborhood and the sub-graph under construction is performed in line 6. The loop is repeated until the source has merged its neighborhood with the sub-graph. Finally, the source computes a set of feasible non-dominated paths in line 8.

Pseudo Code

Algorithm 1 Computation of an end-to-end path between s and t for request q.

SANP(H,s):

Require: node t has received the request q

1: $n \leftarrow t$ {phase 1}

2: $H(V,E) \leftarrow H(V \cup V'_t, E \cup E'_t), \forall v \in V'_t | deg(v) > 1$ {H is initialized with the neighborhood of t. $deg(v)$ returns the degree of node v}

3: **while** $n \neq s$ **do**

4: $n \leftarrow n$.nextAS(s)

5: n.send(H, q) {the request and the sub-graph are forwarded to the next node on the path toward s}

6: $H(V,E) \leftarrow H(V \cup V'_n, E \cup E'_n), \forall v \in V'_n | deg(v) > 1$ {merges the neighborhood of node n with the sub-graph}

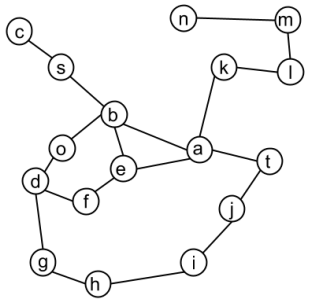
7: **end while**

8: **return** s.selectPath(H) {phase 2}

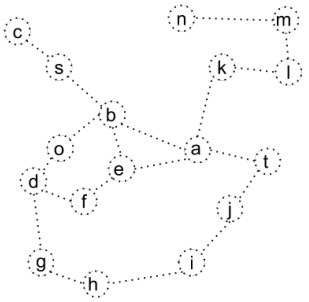
Figure 5.4 shows an example of how H is constructed. In this example, we set $r_0 = 2$ meaning that each node knows a neighborhood composed of all of its neighbors within a radius equal to 2. The destination is node t, the source is node s and the path followed by SANP is [t,a,b,s]. The path [t,a,b,s] corresponds to the shortest path between the nodes t and s. Figure 5.4(A) shows the graph G and Figure 5.4(B) shows the sub-graph H which is at this step equal to $H(V,E)$ with $V \leftarrow \emptyset$, $E \leftarrow \emptyset$. We now assume that destination node has received the request. Then, the condition for Algorithm 1 to start is fulfilled. t is the current node (line 1) and SANP initializes the sub-graph H with N_t, the neighborhood of node t. Figure 5.4(C) shows N_t and the resulting sub-graph at node t is shown by Figure 5.4(D). At this step $H = N_t$. Then, t sends the message containing

H to the next hop which is node a. Once node a receives the message, it merges its neighborhood, represented in Figure 5.4(E). The resulting sub-graph at node a is shown in Figure 5.4(F). Each node in N_a but not in H: nodes s, o, f and l are added to H. They are represented in bold in Figure 5.4(F). Again, node a sends the message containing H to the next hop which is node b. It merges its neighborhood as shown in Figure 5.4(G) with the sub-graph. Figure 5.4(H) shows the sub-graph at node b. This time, only node d represented in bold is added to the sub-graph. As specified in line 6, only the nodes with a degree larger than 1 are added to the sub-graph. Therefore, node c with $\deg(c) = 1$ is not added to the sub-graph. Finally, node b sends the message containing H to node s. Node s merges its neighborhood as shown in Figure 5.4(I) with the sub-graph. Except node c which has a degree equal to 1, all the nodes in the neighborhood of s are already in the sub-graph. Then, the resulting sub-graph is shown in Figure 5.4(J). At this step, the source has merged its neighborhood with H, then phase 2 of Algorithm 1 starts: the source computes a set of feasible non-dominated paths using the sub-graph.

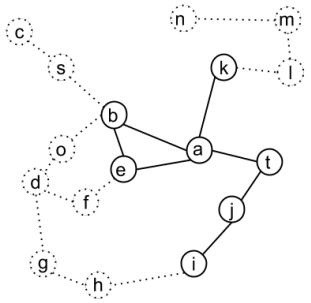
The sub-graph contains both the source and the destination and we ensure that H is a connected graph. As previously discussed, the source can use any algorithm to perform phase 2. As simple example is to consider the path followed by SANP and to compute its weights. If its weights respect the constraints, then the source can use it. We decided to use a brute-force search to list all the paths between the source and the destination. Then, by calculating the weights of each path, we can compute a set of feasible non-dominated paths between the source and the destination. For the sake of scalability, we present two heuristics, in the next section, that aim at limiting the size of the sub-graph H.



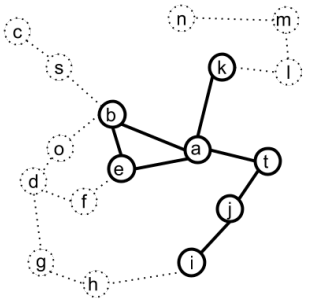
(A) Example of a graph



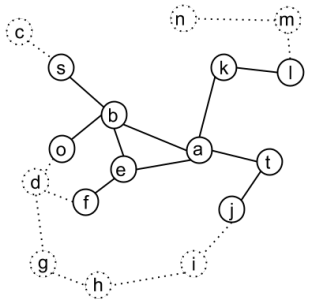
(B) Sub-graph initiliazed to null



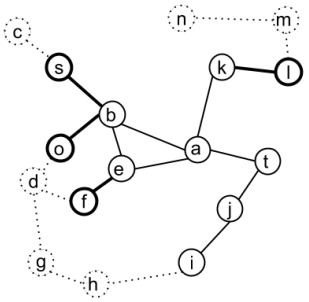
(C) Neighborhood of node t



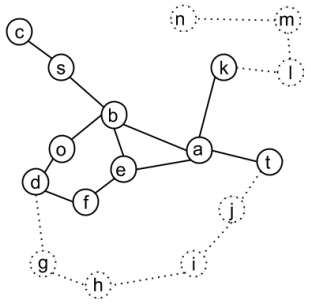
(D) Sub-graph at node t



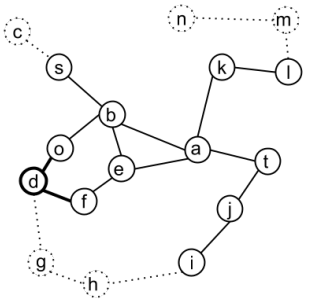
(E) Neighborhood of node a



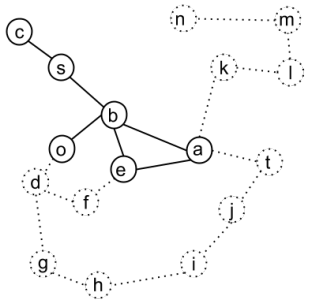
(F) Sub-graph at node a



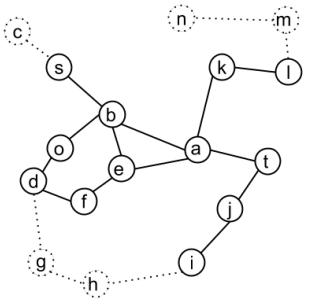
(G) Neighborhood of node b



(H) Sub-graph at node b



(I) Neighborhood of node s



(J) Resulting sub-graph at node s

FIGURE 5.4: Example of the construction of the sub-graph

5.3.1 r-SANP

Recall that, we use a brute-force search algorithm to find a set of feasible non-dominated paths in H. In order for the path search phase to be scalable, the size of H must be limited. Hence, we present two heuristics to limit the size of the sub-graph H. Both heuristics produce, based on H, a reduced sub-graph H' with a size equal to $M = \alpha L$ with L equal to the length of shortest path between source and destination. α is a constant that allows us to tune the size of H'. Furthermore, we designed the two heuristics in such a way that both of them produce H' as a connected graph.

Number of Edges (NoE) Heuristic

The heuristic named *NoE* aims at reducing the number of nodes of the sub-graph H to M at a node n. Intuitively, after merging its neighborhood with the sub-graph H, if the number of nodes in H is superior than the limit M, the node n creates an empty sub-graph H'. By first adding the nodes in the path followed by SANP, n adds recursively to H' the node with the highest number of edges connecting the nodes in H'.

More precisely, let S be to the list of nodes within the path already visited by SANP. After merging its neighborhood with the sub-graph H, if the number of nodes in H is larger than M, n creates a second empty graph $H'(V_{H'}, E_{H'})$ i.e. $V_{H'} = \emptyset$ and $E_{H'} = \emptyset$. n starts to add to $V_{H'}$ all the nodes in S and the edges between theses nodes. Hence, $V_{H'} = \{n | n \in S\}$ and $E_{H'} = \{e(i,j) | i \in V_{H'}, j \in V_{H'}, e(i,j) \in E_H\}$. Then, for the last $M - |S|$ node(s), *NoE* adds recursively in H' the node in H that has the maximum number of edges connecting the nodes in H'. This process is repeated until the number of nodes in H' has reached M. In so doing, we ensure that H' is a connected graph. At the end, $H \leftarrow H'$ and n sends H to the next hop.

Moreover, by first adding the path followed by SANP in H', we ensure that the sub-graph H received by the source contains both itself and the destination. Indeed, both the source and the destination belong to this path. It also ensures that at least one path exists between the source and the destination: the path followed by SANP.

Algorithm 2 describes the heuristic *NoE*. It takes as input a sub-graph H, a maximum number of nodes M and S the list of nodes in the path already visited by SANP. It starts by setting H' to the portion of the path explored so far (line 1). For the last $M - |S|$ nodes to add, Algorithm 2 builds for each node j in H but not already in H' the

set T_j containing all the *edges* between j and a node in H' (line 3), $e(j,k) = \emptyset$, if there is no edge between j and k. In line 4, l is the node that is not already in H' and that has the largest number of edges connecting it to nodes already in H'. If more than one node satisfy this condition, one is randomly chosen. This node is added to H' in line 5 with all the edges connecting it to the nodes in H', so that H' is always connected. The algorithm adds one node to H' at each iteration of the while loop starting in line 2, until there are M nodes in H'.

Pseudo-Code

Algorithm 2 Heuristic *NoE* to limit the size of H to M nodes

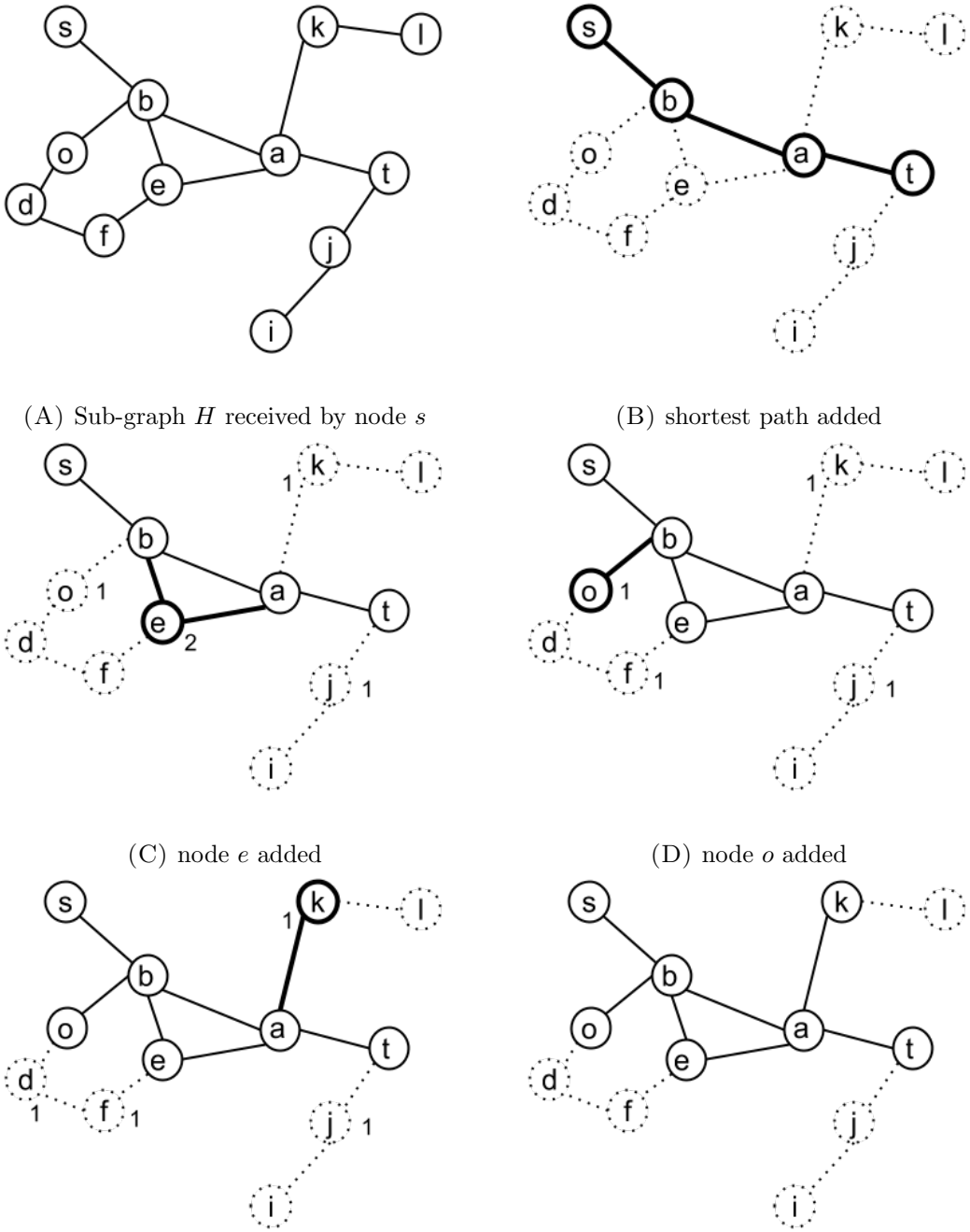
NoE(H, M, S)

1: $H' \leftarrow (V_{H'} \leftarrow S, E_{H'} \leftarrow \{e(j,k) | j \in S, k \in T \ | e(j,k) \in H\})$
2: **while** $|V_{H'}| \leq M$ **do**
3: $\quad T_j \leftarrow \{e(j,k) | k \in H'\} \, \forall j \in \{H \setminus H'\}$
4: $\quad l \leftarrow \arg\max_j |T_j|$
5: $\quad H' \leftarrow (V_{H'} \cup \{l\}, E_{H'} \cup T_l)$
6: **end while**
7: **return** H'

Example

Figure 5.5 shows an example of the execution of the heuristic *NoE* at node s with M equal to 7. The source node is s, the destination node is t and $S = [t,b,a,s]$. Figure 5.5(A) shows the sub-graph H received by node s. As the number of nodes in H is higher than 7, node s uses the heuristic *NoE* to limit to 7 the number of nodes in H. *NoE* starts by creating H' and adding to H' the nodes in S and the edges between them. Figure 5.5(B) shows the nodes in S added to H' with the edges between them corresponding to line 1 of Algorithm 2. For the last $M - |S| = 7 - 4 = 3$ nodes, *NoE* adds recursively the node in H that has the maximum number of edges connecting the nodes in H'. In Figure 5.5(C), the number next to the nodes corresponds to $|T_j|$, the number of edges connecting the nodes in H'. For the sake of clarity, the lack of number next to a node j means that $|T_j| = 0$. As a consequence, this node can not be added to H' yet. For the nodes o, e, k, and j, *NoE* computes a value of $|T_j|$ equal to 1, 2, 1 and 1 respectively. node e is the node with the maximum value of $|T_j|$ ($|T_e| = 2$). Therefore e (bold in Figure 5.5(C)) is

added to H' with the edges between e and the nodes already in H' which corresponds to nodes b and a in this case. At this step, the nodes in H' are s, a, b, t and e and $|V_{H'}| = 5$. Then, *NoE* starts a new iteration. It computes $|T_j|$ for the nodes o, f, k, and j and the values are equal to 1 for each node as Figure 5.5(D) shows. In this case, a node is randomly chosen: node o is added to H' with the edge(s) connecting the nodes in H', $e(o, b)$ in this case. In Figure 5.5(E), the process is repeated and node k is added to H' with the edge $e(k, a)$. At this step, the number of nodes in H' is equal to M. The resulting sub-graph H' is shown in Figure 5.5(F). At the end, node s assigns H' to H.



(A) Sub-graph H received by node s

(B) shortest path added

(C) node e added

(D) node o added

(E) node k added

(F) Resulting sub-graph H'

FIGURE 5.5: Example of the execution of *NoE* at source node s with $M = 7$

Highest Degree (HD) Heuristic

Similarly to the *NoE* heuristic, *HD* aims at limiting the size of H at M nodes in a node n. However, unlike *NoE*, which adds recursively the node that has the maximum number of edges connecting the nodes in H', *HD* adds the node in H that has the highest degree and it is connected with at least one node in H'.

More precisely, let S be to the list of nodes on the path already visited by SANP. As *NoE*, *HD* starts by creating an empty graph $H'(N_{H'}, E_{H'})$ and by adding to H' the nodes in S and the edges between them. For the last $M - |S|$ nodes, *HD* adds recursively the node in H that has the highest degree and it is connected with at least one node in H'. If more than one node satisfy this condition, one is randomly chosen. By doing this, we ensure that H' is a connected graph. By adding the node with the highest degree, *HD* aims at adding in H' the most connected node that corresponds to the node that can offer the greatest path diversity. Thus, *HD* increases the probability to find paths respecting the constraints.

Algorithm 3 shows the pseudo-code of the heuristic *HD*. It takes as input a sub-graph H, the maximum number of nodes M and S, the nodes in the path already visited by SANP. In line 1, *HD* adds in H' the nodes in S and the edges between them. In line 3, for each node j in H but not already in H' and it is connected with at least one node in H', it builds the set T_j corresponding to the degree of node j. In line 4, l is assigned to the node with the highest value of T_j (or maximum degree). In line 5, l is added to H' with all the edges connecting l and the nodes in H'. This operation is repeated until the number of nodes in H' is equal to M.

Pseudo-code

Algorithm 3 Heuristic *HD* to limit the size of H to at most M nodes

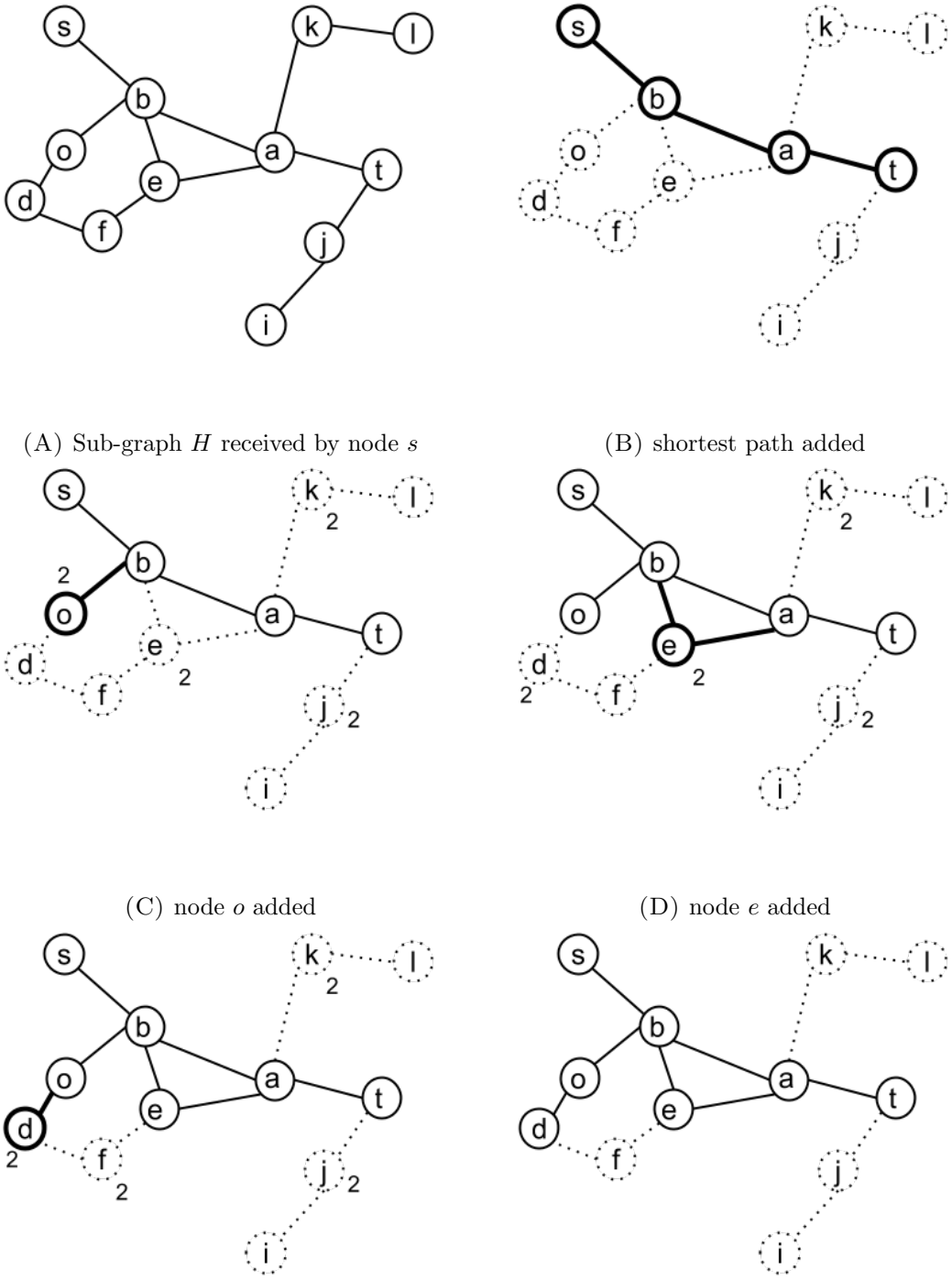
HD(H, M, S)

1: $H' \leftarrow (V_{H'} \leftarrow S, E_{H'} \leftarrow \{e(j,k)|j \in S, k \in T\}|e(j,k) \in H)$
2: **while** $|V_{H'}| \leq M$ **do**
3: $T_j \leftarrow deg(j) \exists e(j,k), k \in H', \forall j \in \{H \setminus H'\}$ {deg(j) returns the degree of node j}
4: $l \leftarrow \arg\max_j |T_j|$
5: $H' \leftarrow (V_{H'} \cup \{l\}, E_{H'} \leftarrow \{e(l,k)|l \in H, k \in H'\})$
6: **end while**
7: **return** H'

Example

Figure 5.6 shows an example of the execution of the heuristic *HD* at node s with $M = 7$. The source node is s, the destination node is t and S = [t,b,a,s]. The sub-graph H

received by node s is shown in Figure 5.6(A). *HD* starts to add to H' the nodes in S and the edges between them as Figure 5.6(B) shows. Then, *HD* computes T_j for each node in H that is connected with at least one node in H'. The number next to the nodes corresponds to its T_j in Figure 5.6(C). The value of T_j for the nodes k, j, e, o is 2. As T_j is the same for each node, a node is randomly chosen. In this case, it is node o. Then, node o, represented in bold, is added to H' with all the edges connecting the nodes in H'. In this example, only one edge is added: $e(o, b)$. In Figure 5.6(D) the nodes in H' are s, a, b, t and o, then $|V_{H'}| = 5$. Therefore, *HD* starts a new iteration. It computes T_j for each node in H and is connected with at least one node in H'. The nodes d, e, j, and k have all the same degree equal to two. Then a node is chosen randomly: node e. Node e is added to H' with all the edges connecting the nodes in H'. In this case the edges are $e(e, b)$ and $e(e, a)$. In Figure 5.6(E), this operation is repeated and node d is added to H' with $e(d, o)$. At this step, the number of nodes in H' is equal to M. The resulting sub-graph H' is shown in Figure 5.6(F). At the end, node s assigns H' to H.



(A) Sub-graph H received by node s

(B) shortest path added

(C) node o added

(D) node e added

(E) node d added

(F) Resulting sub-graph H'

FIGURE 5.6: Example of the execution of *HD* at source node s with $M = 7$

Algorithm 4 shows the pseudo-code of the SANP algorithm extended with the reduction sub-graph' size heuristic. It shows the implementation of SANP with the *HD* heuristic. However, in order to use *NoE* rather than *HD*, it suffices to change line 5 and line 13.

Compared with Algorithm 1, after the merge operations (line 3 and line 11), Algorithm 4 checks whether the number of nodes in H is greater than a constant M. If this condition is true, then it uses *HD* or *NoE* to reduce the number of nodes to M in H.

Pseudo Code

Algorithm 4 Computation of an end-to-end path between s and t for request q.

SANP(H, s):

Require: node t has received the request q

1: $n \leftarrow t$ {phase 1}
2: $S \leftarrow t$
3: $H(V,E) \leftarrow H(V \cup V'_t, E \cup E'_t), \forall v \in V'_t | deg(v) > 1$ {H is initialized with the neighborhood of t. $deg(v)$ returns the degree of node v}
4: **if** $|V| > M$ **then**
5: $H \leftarrow HD(H,M,S)$ {heuristic *HD* to reduce the size of H}
6: **end if**
7: **while** $n \neq s$ **do**
8: $n \leftarrow n$.nextAS(s)
9: n.send(H, q) {the request and the sub-graph are forwarded to the next node on the path toward s}
10: $S \leftarrow S \cup \{n\}$
11: $H(V,E) \leftarrow H(V \cup V'_n, E \cup E'_n), \forall v \in V'_n | deg(v) > 1$ {merges the neighborhood of node n with the sub-graph}
12: **if** $|V| > M$ **then**
13: $H \leftarrow HD(H,M,S)$ {heuristic *HD* to reduce the size of H}
14: **end if**
15: **end while**
16: **return** s.selectPath(H) {phase 2}

5.4 Performance Evaluation

5.4.1 Simulator Overview

In order to evaluate the performance of the SANP algorithm, we have written a simulator in Python. The simulator is a discrete event simulator capable of simulating how each connection is handled. It takes as input the following parameters:

- A graph $G(V,E)$, where V is the set of ASes and E represents the inter-domain links.

- A list QoS offers, one for each intra-domain link. A QoS-offer Q is defined as $Q_i = (b_i, d_i, c_i)$ with b_i equal to the available bandwidth, d_i and c_i equal to the delay and cost respectively.

- A list of connection requests. A request R is defined as $R = (s, t, b_m, d_m, c_m, t_s, t_d)$ where s is the source AS, t is the destination AS, b_m is the minimum bandwidth requested, d_m and c_m are the maximum delay and cost respectively. t_s is the starting time of the request while t_d corresponds to the duration of the request.

5.4.2 Simulation Parameters

Each intra-domain link is characterized by two additive QoS metrics: delay and cost, both of them are uniformly distributed on $[1, 50]$. Since it is difficult to forecast the QoS metric values of the offers, a uniform distribution is able to represent the diversity of the possible offers. The requests arrive according to a Poisson process, with an average of one connection each 20 seconds. The source and destination of each connection are uniformly distributed among all the v_i. In each AS, a "special" node (v_i) represents an end-node belonging to that AS and is the source or the destination of the connections. Each v_i is connected with all the ASBRIs of its AS, the QoS parameters of these links are randomly selected for each connection. Like the other intra-AS links, the delay and the cost are chosen among the same distribution. We run three simulations with different distribution of delay and cost constraints. Since each request represents a certain budget for a maximum delay, we consider the cost as a constraint and not a metric to minimize. In the first simulation, we use d_m and c_m large enough so that the connection can always be satisfied. In the second (resp. third), we set the delay and the cost constraints equal to 150 (resp. 100). Given that, we are mainly interested in the additive constrains and in order to reduce the space of parameters influencing the simulations, we assume that the capacity of all the links is infinite so that bandwidth is never a limiting factor. This is consistent with the notion of offers proposed by ASes: we assume that, faced with a sufficient demand, operators will have the appropriate incentives to increase the prices of each offer and/or increase the capacity they offer, in order to minimize the number of rejected requests.

We run the simulator on an Inet graph composed of 2000 nodes. A simulation consists of 1000 requests and we set $r_0 = 1$. As previously mentioned, the path followed by SANP is the shortest path in term of number of ASes between the source and the destination. The results obtained with the simulations described above can be used to assess both the quantity and the quality of the paths found by SANP. In the sub-graph found by SANP, it is possible to extract a set of feasible non dominated paths. We want to compare this set of paths with the set of feasible non dominated paths in the entire graph. We also use a brute-search force approach to list all the simple paths in the entire graph and then check whether they are feasible and non-dominated. Obviously such a solution is not feasible in the Internet both because the topology of the Internet is not known and, even if it were known, any exhaustive search for such a large graph is out of the question. For the sake of scalability, we set the maximum path length equal to 8 for the simple considered during the brut-search phase.

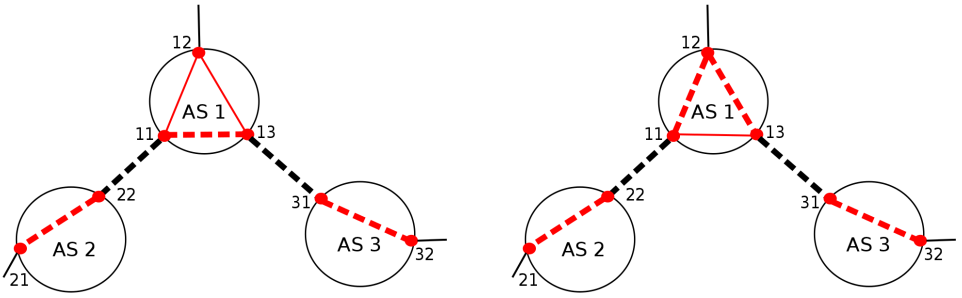
Before presenting the results of the simulations, we present the details of the brute-force search algorithm and we compare the two heuristics (*NoE* and *HD*), presented above, to reduce the size of the sub-graph.

5.4.3 Finding Feasible Non-Dominated Paths in the Sub-Graph

We assume that the ASes propose offers in such a way that an offer between two given ASBRIs has a lower delay and is cheaper than buying any combination of other offers between the ASBRIs of the same AS. That is that for an offer between two ASBRIs i and j is such that its delay is less than the sum of the delays of all the other offers from i to k and from k to j, for all other ASBRIs k of the same AS. The same for the cost. Note that this is a reasonable assumption: if an offer were not to satisfy this constraint, it would never be used.

We exploit this constraint to simplify the feasible path search phase. We forbid the brute-search force algorithm to use two intra-domain links one after the other. From an entry ASBRI a to an exit ASBRI b, the brute-search force algorithm must choose the offer connecting the two ASBRIs a and b. In other words, we forbid the brute-search force to start from the ASBRI a, reach first the ASBRI c before reaching the ASBRI b.

As an example, Figure 5.7(A) shows an admissible path search phase. From the ASBRI 11 of the AS 1 to the ASBRI 13 of the same AS 1, the appropriate offer is taken corresponding to the dashed path. However, in Figure 5.7(B), the path is forbidden because we reach first the ASBRI 12 of AS 1 and after ASBRI 13 from the ASBRI 11 of AS 1.



(A) Path allowed (B) path forbidden

FIGURE 5.7: Example of paths allowed and forbidden

5.4.4 Selecting Heuristic for Reducing the Sub-Graph Size

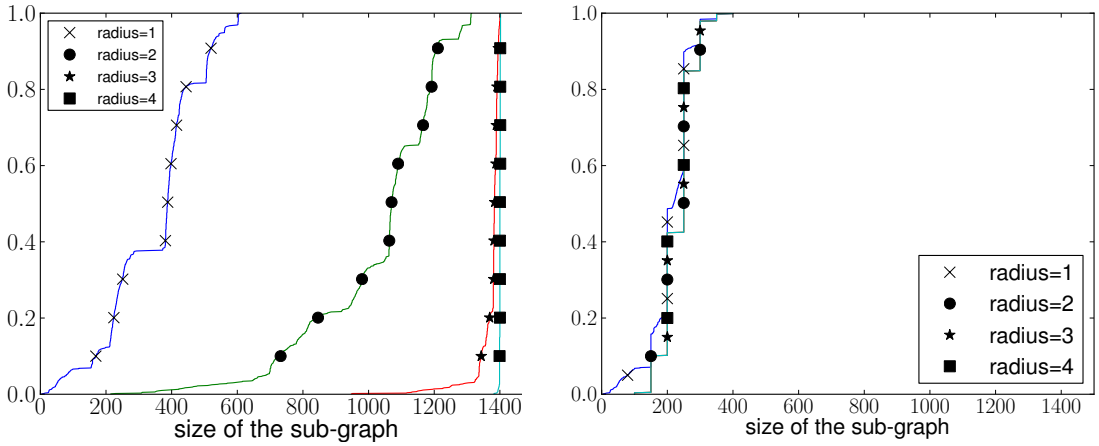
The present section is dedicated to discussing the differences between the two sub-graph size reduction heuristics. To achieve this, we are going to assess whether the heuristics construct different sub-graphs by examining the common and the different nodes and study whether the latter have an influence on the path's dominance i.e., whether the paths found with one heuristic dominate the paths found with the other heuristic or not.

Before doing this, we want to compare the size of the sub-graphs constructed with SANP described by Algorithm 1 and SANP extended with the sub-graph reduction heuristic described by Algorithm 4 (r-SANP). For that, we use the simulator described in Section 5.4.1 and simulate 1000 requests on an Inet graph with 2000 nodes with $r_0 = 1, 2, 3, 4$. As both heuristics reduce the size of the sub-graph to the same value $M = L \times \alpha$, we show the results for only one heuristic.

Figure 5.8(A) shows the CDF of the sub-graph' size for the values of $r_0 = 1, 2, 3, 4$ with SANP (Algorithm 1). The graph generated with Inet has 600 nodes with a degree equal to 1. As specified in Line 6 of Algorithm 1, only the nodes with a degree larger than 1 are added in the sub-graph. Hence, with $r_0 = 3, 4$, Figure 5.8(A) shows that for the

vast majority of the requests, SANP constructs a sub-graph with a size equal to 1400 that corresponds to the maximum possible value.

Figure 5.8(B) shows the CDF of the sub-graph using r-SANP. We set $\alpha = 50$. Thanks to the heuristic, the size of the sub-graph is no more than 15% of the graph' size.



(A) CDF of the sub-graph size using Algorithm 1

(B) CDF of the sub-graph size using Algorithm 4

FIGURE 5.8

In order to determine whether the nodes in the sub-graph H received by the source node s are different using *HD* or *NoE*, we have simulated 1000 requests on an Inet graph composed of 2000 nodes using both the heuristics *NoE* and *HD*. We set $\alpha = 50$ and $r_0 = 1$. Figure 5.9 shows the CDF of the size of the sub-graphs using both *HD* or *NoE* as well as the CDF of the number of different nodes in the sub-graph H received by the source.

By comparing the CDF of the total number of nodes in the sub-graph and the CDF of the number of different nodes in the sub-graph using *HD* or *NoE*, we can conclude that both heuristics construct sub-graphs in which the nodes are almost the same.

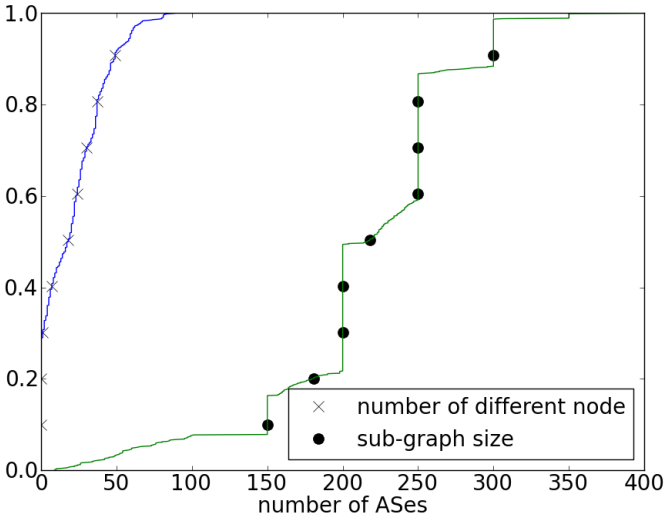


FIGURE 5.9: CDF of the total number of ASes and the CDF of the number of different ASes

We now want to determine whether the nodes included in a sub-graph using one heuristic but excluded using the other heuristic have an impact on path's dominance. In the context of multi-criteria optimization problem, there is no straightforward way to distinguish what is the best solution. Indeed, a solution is generally a set of non dominated points and without additional subjective information such as a criteria preference, it is difficult to affirm that a non dominated set is better than another. In this dissertation, we say that a request is better using *HD* (resp *NoE*) if at least one path found with *HD* (resp *NoE*) dominates all the paths found with *NoE* (resp *HD*). Of the 1000 requests, 1 request is better with the heuristic *HD* while 2 requests are better using the heuristic *NoE*. For the 997 remaining requests we use the following metrics to characterize these solutions. Each metric reveals a certain aspect of the "quality" of the solutions found. A metric is either relative, i.e., it produces an integer that indicates a certain aspect of the difference between two non dominated sets or it is absolute, i.e., it produces an integer for each non dominated set. The comparison between these two values allows us to affirm which one is the best non dominated set according to this metric.

5.4.4.1 Comparing Solutions

Zitzler [121] introduced a metric that counts the average number of solutions in a set that are dominated by at least one solution in another set. Formally, for two sets of

solutions A and B

$$C(A,B) \triangleq \frac{|\{b \in B; \exists a \in A : a \succeq b\}|}{|B|}$$

where $a \succeq b$ means that a dominates b and $|A|$ is the order (cardinality) of set A. $C(A,B) = 1$ means that each solution in A taken individually is dominated by at least one solution in B while $C(A,B) = 0$ means that none of the solutions in A are dominated by a solution in B. As an example, in Figure 5.10: $C(A,B) = \frac{0}{3} = 0$ and $C(B,A) = \frac{1}{4}$. Note that the situation corresponding to the 21 requests mentioned above *does not* correspond to the case the $C(A,B) = 1$; for example in the case of Fig. 5.10 those 21 requests correspond to the case where there is *at least one* solution in A that dominates *all* the solutions in B.

Even though one could say that if path a dominates b ($a \succeq b$) then path a is better than path b, it is still worthwhile to compute the distance between these two paths. The generational distance G [122] computes how far a solution in a set is from its closest solution in the other set. We use a slightly modified version of G that computes the mean of the the euclidean distances between each solution in A and its closest solution in B. Formally,

$$G(A,B) \triangleq \frac{1}{n}\sum_{i=1}^{n} g_i.$$

where $n = |A|$. If a_i is dominated by its closest solution in B we set $g_i = -d_i$ where d_i is the euclidean distance between a_i and its closest solution in B and $g_i = d_i$ otherwise. As an example in Figure 5.10, $G(A,B) = \frac{g_1+g_2+g_3}{3}$ where g_1, g_2 and g_3 are all positives. If we compute $G(B,A)$, the distance between b_2 and its nearest solution in A (i.e., a_1) would be negative.

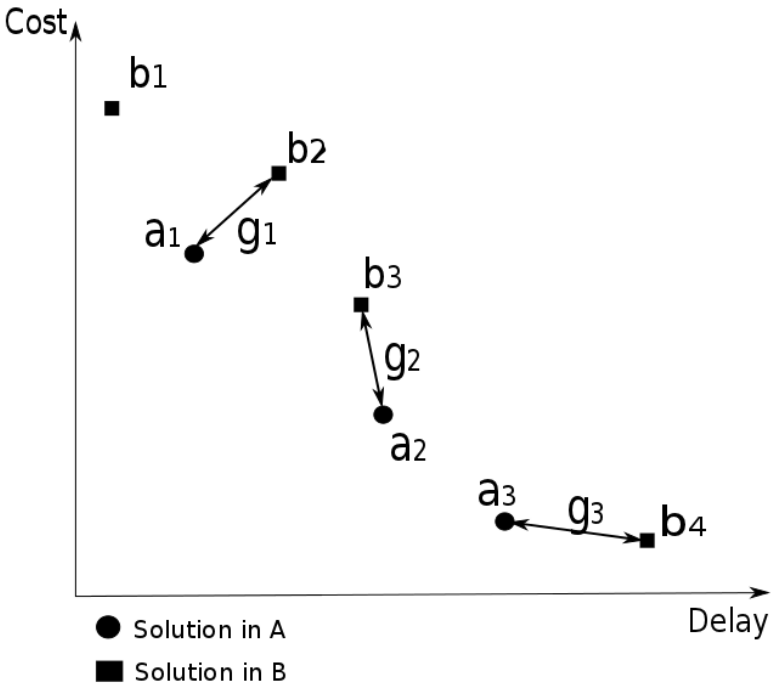


FIGURE 5.10: Summary of the metrics

Using these metrics, we now compare the 997 requests mentioned above. Figure 5.11 shows the CDF of $C(HD, NoE)$ and $C(NoE, HD)$. $C(HD, NoE)$ (resp $C(NoE, HD)$) $= 1$ means that all the paths taken individually found with the heuristic HD (resp. NoE) are dominated by at least one path found with NoE (resp HD). These cases never occur. Hence, for none of the 997 requests, we can claim that a heuristic is better than the other according to C. Furthermore, for only negligible cases, one heuristic finds paths that dominate some other paths found with the second heuristic. This result is confirmed by Figure 5.12 that shows the CDF of $G(NoE, HD)$ and $G(NoE, HD)$. Both $G(NoE, HD)$ and $G(NoE, HD)$ are almost always equal to 0 meaning that both heuristics find the same set of paths.

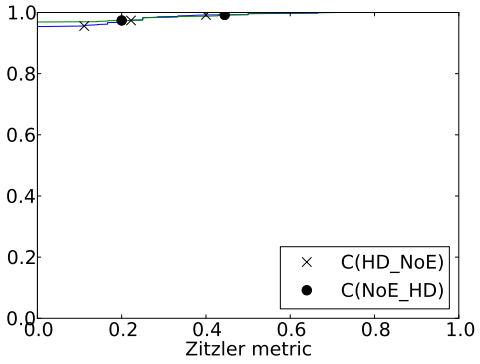


FIGURE 5.11: Zitzler metric CDF

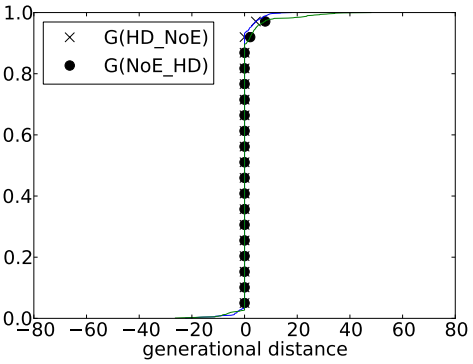


FIGURE 5.12: Generational Distance CDF

As we have shown using simulations, both heuristics yield very similar results. However, because the heuristic *HD* executes the reduction of sub-graph more rapidly, we opt for this heuristic. Indeed, *HD* just has to access to the degree of the nodes which is a constant while the *NoE* heuristic has to compute how many edges a node has connecting it to the sub-graph for each new node added in the sub-graph. For the rest of the manuscript, SANP uses the *HD* heuristics to reduce the size of the sub-graph.

5.4.5 Simulation Results

Using the parameters described in Section 5.4.2, we find that for 21 requests over 1000, all the paths found by SANP are dominated by *at least one* path found in the entire graph. In other words, by reusing the definition of "best request" defined in Section 5.4.4, we can say that 21 requests are better using the entire graph than SANP. For the 979 remaining requests, it is impossible to directly affirm what is the best non dominated set, therefore we use the metric C and G described above.

Figure 5.13 shows the CDF of $C(G, H)$ and $C(H, D)$. Obviously $C(G, H) = 0$, because the solutions in the whole graph is the Pareto frontier. We can observe that $C(H, G) = 1$, meaning that all the solutions found by SANP taken individually are dominated by at least one path found in the whole graph occurs for very few cases. In other words, the cases where SANP finds none of the solutions in the Pareto frontier is negligible. Moreover, for 60% of the cases, $C(H, G) = 0$ meaning that SANP finds some or all solutions in the Pareto frontier.

Figure 5.14 shows "how far" the solutions found by SANP are from the Pareto frontier. We can remark that the maximum negative value of $G(H, G)$ is equal to -50 and $G(H, G)$ is negative for 20% of the cases. Considering the link weights, we can conclude that SANP finds paths that are reasonably close to the paths found in the whole graph.

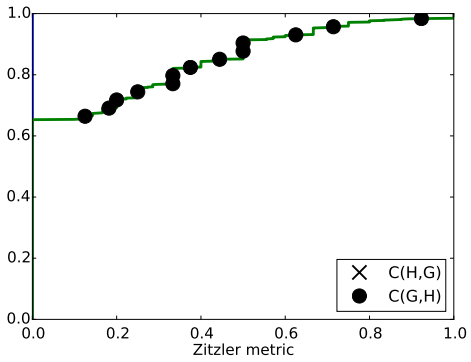


FIGURE 5.13: Zitzler metric CDF

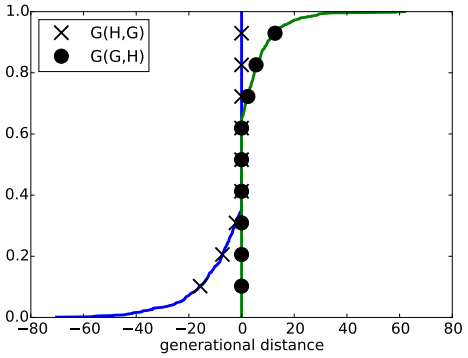


FIGURE 5.14: Generational Distance CDF

5.4.5.1 Assessing the Influence of the Parameters

Influence of the capacity constraint

In order to assess the influence of the capacity, we simulated 1000 requests on an Inet graph composed of 2000 nodes with values of delay and cost constraints large enough not to be the limiting factor. We set the capacity of each link equal to 1000. We run five simulations with the bandwidth constraint uniformly distributed according to column 1 of Table 5.1. For each simulation, we set the duration of a connection large enough such that the path found for a specific request, if such a path exist, is reserved throughout the simulation. Hence, in the first simulation, the network is lightly loaded, i.e., a link can support several connections. In the second and third simulations we increase the network load by decreasing the probability that a link can support several connections. Finally, the fourth simulation corresponds to an extreme case, i.e., each link can support only one connection.

Column 4 corresponds to the case a solution exists in the whole graph but SANP does not find it. Column 3 corresponds to the case when SANP finds a solution but at least one path in the whole graph dominates all the paths found with SANP. Column 2

corresponds to the case a solution exists in the whole graph and SANP finds a solution, but no solution in a set dominates all the solutions in the other set.

First, as expected, the higher the load in the network is, the fewer requests satisfied by SANP. Second, the number of requests for which SANP does not find a solution but a solution exists considering the whole graph increases with the network load. This result can be explained by the fact that the higher the network load, the harder it is to find a suitable path. In other words, it is possible for SANP to construct a sub-graph in which no path has enough capacity to satisfy the request. However, this happens only for a few requests. As a conclusion, in a case of bandwidth being the limiting factor, for the vast majority of the requests SANP is able to find a solution if one exists.

Bandwidth Constraint	Neutral	Whole Graph	only Whole Graph	
$[200, 400]$	923	26	2	
$[200, 600]$	826	24	15	
$[400, 600]$	711	26	27	
999	554	384	25	38

TABLE 5.1: Summary of the simulations assessing the influence of the bandwidth constraint

Influence of the Delay and Cost Constraints

In this part, we want to assess the ability of SANP to find paths when the delay or the cost are the limiting factor. If we utilize a uniforme distribution for the delay and the cost between $[1, c]$ with $c > 1$, the shortest path would have the highest probability to have the lowest delay and cost. Moreover, as SANP ensures that the shortest path is in the sub-graph, SANP has the highest probability to find a solution if one exists. In order to modify this side effect, we set the delay and cost distributions as follow: for 10% of the cases, the delay (cost) is uniformly distributed on $[1, 10]$ and the for the remaining 90% of the cases, the delay (cost) is uniformly distributed on $[1000, 1500]$. At the end, only few links have a low delay or low cost meaning that the shortest path is unlikely to have the lowest delay and/or cost. Furthermore, we ensure that the bandwidth is not the limiting factor by setting the bandwidth constraint to 0.

We run five simulations of 1000 requests on a graph composed of 2000 nodes with different values of delay and cost constraints as shows the first column in Table 5.2. For each simulation, we compute the number of requests for which the solutions found with SANP are dominated by at least one solution found in the whole graph. It corresponds to the second column. The third column corresponds to the number of requests for which SANP finds a solution. The fourth column corresponds to the number of requests for which SANP does not find a solution but one exists in the whole graph.

Delay and cost constraints	Number of requests at least one solution is better in the whole graph	Number of requests SANP finds a solution	Number of requests SANP does not find a solution but one exists
4500	0	133	6
5000	3	351	23
5500	17	579	12
6000	18	687	21
Not a limiting factor	15	1000	0

TABLE 5.2: Summary of the simulations assessing the influence of the delay and the cost constraints

First, whenever the delay and the costs are not the limiting factor, the number of requests for which the solutions found with SANP are dominated by the solutions in the whole graph is very low. This results corroborates the one presented at the beginning of the section: on a different Inet graph composed of 2000 nodes, we run 1000 requests and found 21 requests for which the solutions found by SANP are dominated at least by one solutions in the whole graph. Then, starting from loose constraint we run trough different values of delay and cost constraints to reach very tight ones. As expected, the tighter the delay and cost constraints are, the fewer requests are satisfied (column 3 and 4). Among the requests for which SANP finds a solution, very few have a better solution in the whole graph (column 2). Therefore, the solutions found by SANP are very close to the solutions found in the whole graph. Moreover, column 4 corresponds to the number of requests for which SANP does not find a solution but a solution exists in the whole graph. This case does not occur frequently. To conclude, the solutions found

with SANP are very close to the one found in the whole graph considering very strict delay and cost constraints.

Influence of α

Recall that α multiplied by the shortest path length between the source and the destination is the limit of the sub-graph size. Hence, the larger the value of α, the larger the size of the sub-graph. We want to determine the number of requests for which a solution considering the whole graph is better than the solutions found with SANP as a function of α. We run on an Inet graph composed of 2000 nodes 1000 requests and we set different values of $\alpha = 2, 5, 10, 25, 50, 75$. The bandwidth, the delay and the cost constraints are large enough not to be the limiting factors. The delay and cost of the intra-domain links are uniformly distributed on $[1, 50]$ and the bandwidth is equal to 1000.

Figure 5.15 shows the sub-graph size for different values of α. We can remark that the parameter α enables us to change the sub-graph size in fine-grained manner.

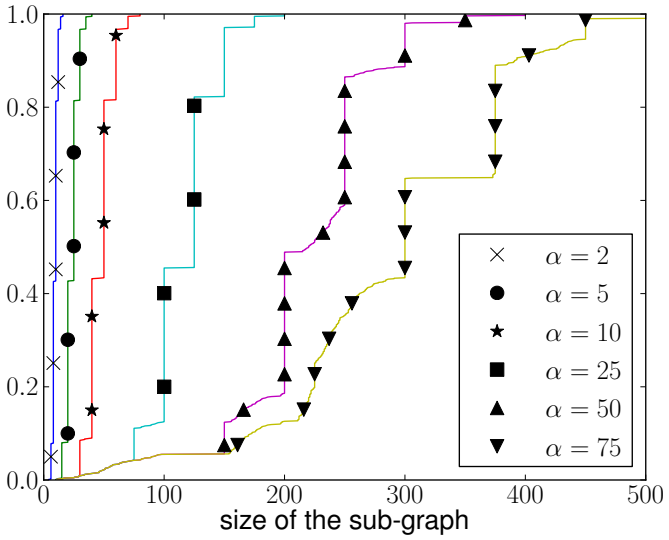


FIGURE 5.15: Sub-graph size for different value of α

Table 5.3 summarizes the number of requests for which the solutions found by SANP are dominated by at least one solution in the whole graph. As expected, the higher the value of α, the better the solutions found with SANP. We can also remark that, even for very small values of α (5,10) that lead to very small size of the sub-graph, the paths found by SANP are relatively good. In these cases, the sub-graph corresponds to the

nodes on the shortest path and few other nodes connected to them. It means that the shortest path and some other paths related to it lead to good solutions. As a conclusion, for small values of α (5,10,25), the quality of the paths found with SANP are very close to the ones in the whole graph.

α	Number of requests for which the solutions found with SANP are dominated
2	301
5	156
10	83
25	29
50	21
75	18

TABLE 5.3: Number of requests for which the paths found with SANP are dominated considering the paths in the whole graph function of α

5.5 Conclusion

We have presented SANP, a distributed algorithm capable of finding feasible non-dominated end-to-end paths satisfying QoS constraints. To do so, it constructs a sub-graph around the path used by existing routing mechanisms between the source and destination. The sub-graph is obtained by merging the neighborhoods of all the nodes traversed by the request. A neighborhood is defined as all the edges and nodes within a certain distance r_0. The source has complete knowledge of this sub-graph and it can use any algorithm capable of finding feasible non-dominated paths.

In order to preserve the confidentiality of each AS, SANP uses an abstract representation of each AS using only the ASBRIs. All the ASBRIs belonging to the same AS are connected by a complete graph, with each link representing a service offering between two routers. These services are characterized in terms of QoS metrics (e.g., bandwidth, delay, cost) between two ASBRIs. For the sake of autonomy, each AS is free to implement these services as it wishes, without the need to reveal any information about the topology of its network. Furthermore, as the size of the sub-graph increases with r_0, we implemented two heuristics aiming at limiting the size of the sub-graph such that SANP is scalable

with r_0. Both heuristics limit the sub-graph size to M to αL, where L is the shortest path length between the source and the destination.

Using simulations, we compared the paths found using SANP to the paths considering the entire graph with constraints large enough so that they are never a limiting factor. With $\alpha = 50$, we showed that SANP finds paths close to the paths in the entire graph. Moreover, α enables us to tune the quality of the paths found with SANP. Even for reasonably small values of α (5,10,25) the results are fairly good. Moreover, by setting the bandwidth (or the delay or the cost) as the limiting factor, SANP almost always finds a solution if one exists. In this context, again, the solutions found by SANP are very close to the solutions in the entire graph.

Finally, in our simulations SANP follows the shortest path between the source and the destination thanks to its simplicity. We plan to run extensive simulations in which SANP follows a path different from the shortest one.

Recent research highlight that the creation of an alliance of ASes can improve the number of satisfied requests. In the next chapter, we address the issue, by studying whether the presence of several alliances in the Internet can improve the number of requests satisfied.

Chapter 6

The ACQA Algorithm

6.1 Introduction

An alliance is defined as a group of ASes that trust each other and agree to share business and/or technical policies. The literature has investigated the notion of alliances to deal with different issues such as improving BGP security [123, 124] or facilitating the search of end-to-end QoS paths within a single alliance [12, 43, 125, 126]. In this chapter, we address the problem of finding end-to-end QoS paths in the presence of alliances. To the best of our knowledge, this problem has not been investigated yet. Thus, we present an algorithm, called ACQA, capable of finding end-to-end QoS paths in the Internet composed of alliances with some ASes remaining independent, i.e., not belonging to any alliance.

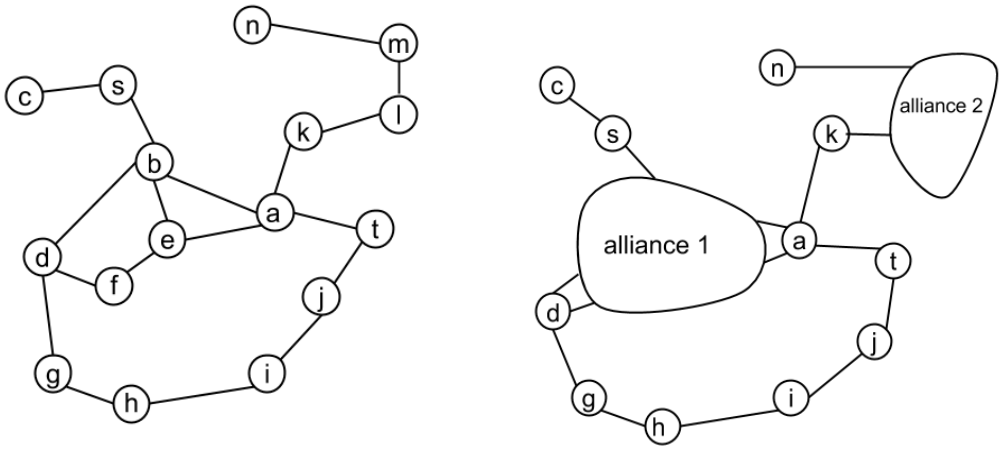
We start to introduce our model in Section 6.2 and the related works in Section 6.3. In Section 6.4 we present ACQA and evaluate its performance in Section 6.5 by comparing it with SANP.

6.2 Model

The model we use in this chapter is based on the model presented in the previous chapter in which we integrate the notion of alliance. The difference being that, rather than being composed of ASes only, the model is composed of both ASes and alliances. As we are interested in searching paths between end nodes located either in an AS or in an alliance,

our model must respect the three properties of Internet communications: confidentiality, autonomy of the ASes/alliances and scalability.

By starting from a graph $G(V, E)$ as defined in Section 5.2, it is possible to construct an Alliance-level multi-graph $G_a(V_a, E_a)$ in which the nodes represent either ASes or alliances and E_a is a multiset of edges between the ASes and the alliances. For instance, let's take a graph composed only of ASes as Figure 6.1(A) shows. Now, let assume that ASes b, f and e want to form an alliance that we call alliance 1 and ASes m and l want to group together to form another alliance that we call alliance 2. Figure 6.1(B) shows the corresponding Alliance-level multi-graph. Note that AS a has two connections with alliance 1 corresponding to its connections with ASes e and b at the AS-level. Similarly, AS d is connected with ASes b and f at the AS-level. Generally speaking, an AS is connected as many time to an alliance as it is connected to the ASes belonging to the alliance at the AS-level.



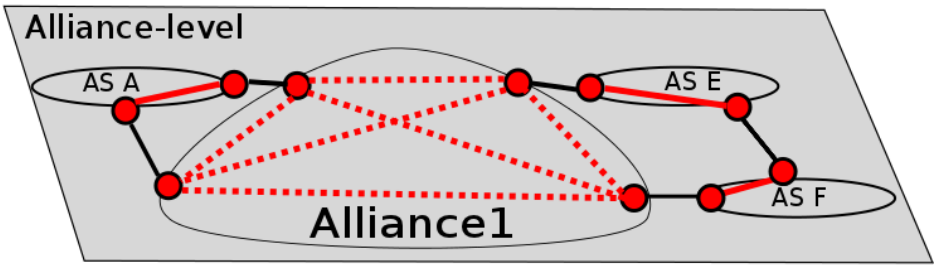
(A) graph composed of ASes only (B) corresponding Alliance-level topology

FIGURE 6.1

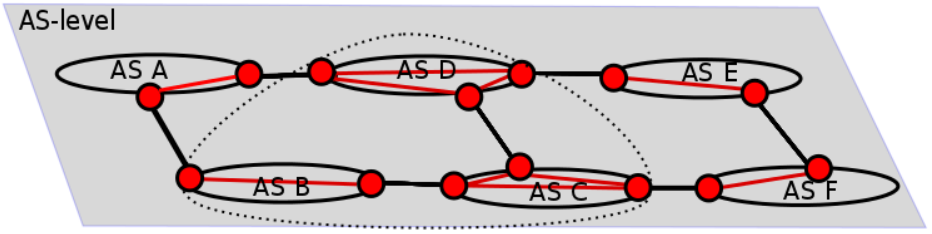
Like an AS that willing to offer QoS guarantees between their entry end exit points, we consider an alliance as a "macro-AS" that is also ready to offer QoS guarantees between its entry and exit points. Hence, as ASes publish offers between their ASBRIs, we assume that alliances publish offers between their edge-ASBRIs. We define an edge-ASBRI as an ASBRI that is connected with at least one another ASBRI belonging to an AS outside the alliance. We name *AS-offer* an offer published by an AS and an

alliance-offer an offer published by an alliance. In order to represent the AS-offers of an AS, we connect all its ASBRIs by a complete graph. In the same way, in order to represent the alliance-offers of an alliance, we connect all its edge-ASBRIs by a complete graph. Thus, these intra-alliance links represent the alliance-offers, hiding the internal topology for confidentiality purposes. Hence, each node in G_a is willing to guarantee QoS between any of its entry/exit points. We characterize each intra-AS or intra-alliance link by a vector w with K weights representing the K additives QoS metrics. Just like each AS is free to decide the offers it publishes, it is up to each alliance to define a list of available alliance-offers. To preserve their autonomy and to respect their confidentiality, alliances do not need to specify how these alliance-offers are implemented inside the alliance. Each alliance needs only to publish a list of QoS metric, such as bandwidth, cost and delay, representing the QoS guarantees between two given end-points. In this dissertation, we assume that an alliance-offer is a chain of AS-offers. (see Section 6.4.1.1 for more details).

As an example, consider Figure 6.2(A) that represents a graph composed of six ASes at the AS-level as described in Section 5.2. Now, let assume that ASes b, c and d decide to form an alliance named alliance 1. The corresponding Alliance-level topology is shown by Figure 6.2(B). The alliance has 4 edge-ASBRIs that are connected by a complete graph to represent the alliance-offers.



(A) Representation of the offers at the Alliance-level



(B) Representation of the offers at the AS-level

●	ASBR	Autonomous System Border Router
———	AS-offer	W = {w_1, ..., w_K}
———	Inter-domain link	
.....	Alliance-offer	Chain of AS-offers

FIGURE 6.2: Representation of the offers within the graphs

6.3 Related Works

Several works have already considered the notion of alliance between different ASes. Most of them focus on finding QoS paths within a single alliance.

In [43], *Service Level Specifications* (SLS) must be published to a neutral centralized third party, which is responsible for negotiating, on behalf of the customer, a chain of SLSs with the rest of the members such that the customer request is satisfied. In [126],

the authors use a Q-learning algorithm to negotiate SLAs between NSPs that belong to the same federation. In [127], the authors present the notion of *federation* as a short-lived association between different stakeholders within the Internet in order to offer end-to-end services.

The FP7 project ETICS [12] introduced different notions of alliance, depending on the trust between the members, the business and/or technical policies that are shared to a lesser degree. The *open association*, which corresponds to the lower level of trust, is a very open and dynamic community with no membership constraints. The information exchange between the members is similar to today's Internet, namely, the offers and their end-points. In an open association, only bilateral agreements between neighbors (SLAs exchanges) are allowed. It is possible to build end-to-end paths by combining the SLAs of the ASes along a path. As it does not require to exchange critical information or to set up complex mechanism such as end-to-end QoS monitoring, an open association is recommended to early markets.

In the second level of trust, called *federation*, the members are ready to better cooperate and fully share technical information, i.e., their offers. Therefore, the end-to-end QoS paths are computed with the complete knowledge of the offers within the community. Moreover, a monitoring service must be deployed in each member such that the community is able to retrieve specific data.

At the highest level of trust, called *alliance*, the members are ready to share both technical and business information to a centralized facilitator entity. The investment are joint and the revenues but also the penalties are shared. An alliance community is recommended for mature market.

In [125], the authors propose to compute QoS paths within an alliance via the Routing Control Platform (RCP) [128], an architecture whose goal is to address the scalability issue of the internal Border Gateway Protocol in a large full mesh network. RCP has been designed to correctly distribute the routes in a fast and reliable way among the routers. In the context of computing QoS paths, the authors utilize the RCP architecture to install an RCP entity within each domain of an alliance. The RCP entities are responsible for establishing and maintaining connections between each other and to compute and to handle connection requests with QoS requirements. To do so, they

exchange reachability information through TCP. This solution is similar to the PCE architecture.

To the best of our knowledge, [129] is the only work that considers several alliances when searching for end-to-end QoS paths. They do not consider the notion of "alliance-offer" but rather, an alliance is defined as a group of ASes that share some information about network services availability and reachability. User requests specify only the source node, the requested service and QoS constraints but not the destination node. In our work we consider communications between *two* given nodes and not between a node and any other node capable of offering a certain service.

Unlike the existing works that find end-to-end QoS paths within a single alliance, we aim at finding end-to-end QoS paths between ASes and/or alliances. To this end, we have introduced the notion of *alliance-offers* that corresponds to a contract published by the alliance in which it specifies which guarantees it can offer between an entry and an exit point. We assume that the level of trust between the members of an alliance is enough to allow them to exchange the information needed to establish the list of available alliance-offers. Compared to ETICS, the level of trust between the members is similar to a federation. However, to preserve the autonomy of each alliance, we assume that each alliance will choose independently the technology used to implement the alliances-offers. One possible solution is to implement one of the solution presented above, e.g., a centralized third entity as defined for an alliance in ETICS.

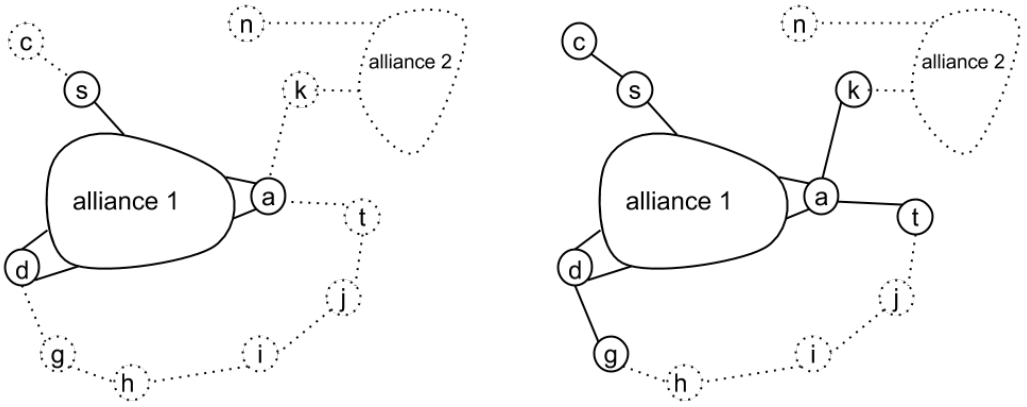
6.4 The ACQA Algorithm

In this section, we present ACQA (**A**lgorithm for **C**omputing end-to-end **Q**oS path within the Internet composed of ASes an **A**lliances) [22], an algorithm capable of finding end-to-end QoS paths within the Internet composed of ASes and alliances. Similarly to SANP, ACQA explores a region between the source and the destination in order to identify feasible non-dominated paths that can satisfy the request. ACQA constructs a sub-graph H composed of several neighborhoods. The source receives this sub-graph and it uses it to compute a set of feasible non-dominated paths between itself and the destination. However, by introducing the notion of alliance, we expect ACQA to explore

a different region than SANP leading to a different and a larger sub-graph in terms of ASes.

Within an alliance-level multi-graph $G_a(V_a, E_a)$, we define a neighborhood N_i of a node $i \in V_a$ as the set of nodes V_i' composed of ASes and/or alliances and the set of edge E_i' within a certain distance r_0. Formally, let $G_a = \{V_a, E_a\}$ be an alliance-level multi-graph (Figure 6.2(B)), the neighborhood of node i, is the set $N_i = \{V_i', E_i'\}$ where $V_i' = \{x | d(i,x) \leq r_0, x \in V\}$ and $E_i' = \{e | e(j,k) \in E, j \in V_i', k \in V_i'\}$ here $d(i,j)$ is the length of the shortest path in term of number of hops between node i and node j and $e(j,k)$ is the edge between nodes j and k.

As an example, consider the alliance-level topology shown in Figure 6.1(B). The neighborhood of alliance 1 with $r_0 = 1$ (resp $r_0 = 2$) are shown as non-dashed nodes and edges in Figure 6.3(A) (resp Figure 6.3(B))

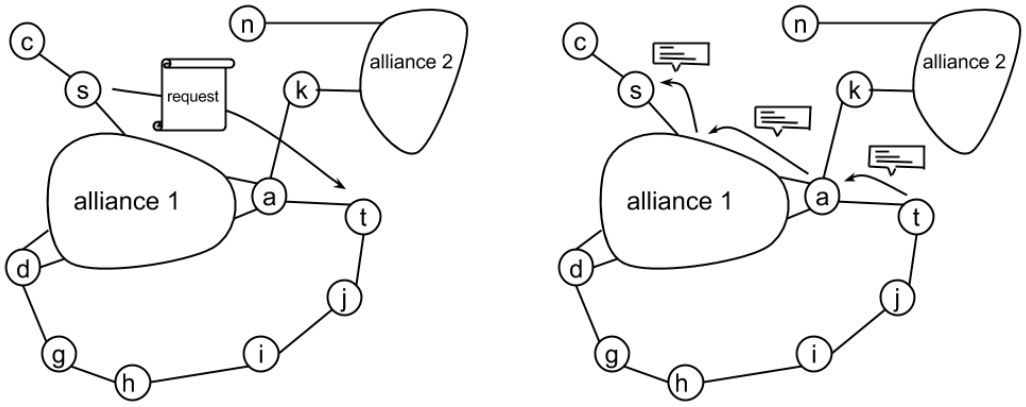


(A) Neighborhood of alliance 1 with $r_0 = 1$ (B) Neighborhood of alliance 1 with $r_0 = 2$

FIGURE 6.3

Whenever a source node s wishes to establish an end-to-end QoS guaranteed path with a destination node t, it sends a request q, specifying the QoS requested on the path toward node t. The request is forwarded to t using the existing routing mechanism as shown in Figure 6.4(A). Upon the reception of this demand, t creates a message that contains its neighborhood and sends it to the source. The message follows the path dictated by the routing mechanism as Figure 6.4(B) shows. As we did with SANP, we use the shortest path in term of number of ASes between the source and the destination in our simulations. Each node along the path, either an AS or an alliance, merges

its neighborhood with the sub-graph H until the message reaches the source. In Figure 6.4(B), AS a, alliance 1 and AS s merge in this order their neighborhood to H. Like SANP, whenever the size of the sub-graph is greater than a limit M, it uses the *HD* heuristic to limit the size of H to M. At the end, the source node s has a sub-graph that contains ASes and/or alliances. It uses this sub-graph to compute a set of feasible non-dominated paths. However, unlike SANP that uses a sub-graph composed only of ASes, the sub-graph constructed by ACQA is composed of ASes and/or alliances. As a consequence, the path search phase performed by ACQA is different from the one performed by SANP. We describe this part in Section 6.4.1.



(A) request sent to the destination (B) the message follows the shortest path

FIGURE 6.4

Algorithm 5 gives the pseudo-code of ACQA. The execution of the algorithm starts when the destination node t receives the request, then, it initializes the variables S and H (lines 2,3) that will contain, respectively, the nodes in the shortest path *from* the destination *to* the source and the sub-graph. Like SANP, if the size of the sub-graph is superior to a limit M (line 4), ACQA uses the heuristic *HD* to reduce the size of the sub-graph (line 5). The loop starting at line 7 is executed at each node along the shortest path: given by the underlying routing mechanism (line 8), the current node n sends the message to the next node at line 9. Once the next node has received the message, it adds itself to the set S (line 10) and then merges its neighborhood with the sub-graph (line 11). Recall that (V'_n, E'_n) is the neighborhood of node n and that each node knows its own neighborhood. If the size of the sub-graph is above a given threshold

(line 12), we use the *HD* heuristic to reduce its size. In the second phase (line 16), the source has received the sub-graph and computes a set of feasible non-dominated paths.

Pseudo Code

Algorithm 5 Computation of an end-to-end path between t and s for request q.

ACQA(H, s):

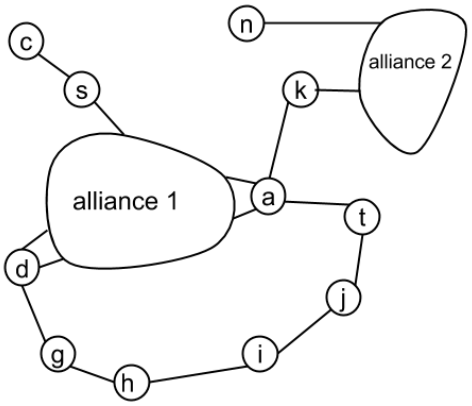
Require: node t has received the request q

1: $n \leftarrow t$ {phase 1}
2: $S \leftarrow \{t\}$ {set of the nodes already visited, it will contain the shortest path between s and t}
3: $H(V,E) \leftarrow H(V \cup V'_t, E \cup E'_t), \forall v \in V'_t | deg(v) > 1$ {H is initialized with the neighborhood of t. $deg(v)$ returns the degree of node v}
4: **if** $|V| > M$ **then**
5: $H \leftarrow HD(H,M,S)$ {heuristic *HD* to reduce the size of H}
6: **end if**
7: **while** $n \neq s$ **do**
8: $n \leftarrow n$.nextAS(s)
9: n.send(H, q) {the request and the sub-graph are forwarded to the next node on the path toward s}
10: $S \leftarrow S \cup \{n\}$
11: $H(V,E) \leftarrow H(V \cup V'_n, E \cup E'_n), \forall v \in V'_n | deg(v) > 1$ {merges the neighborhood of node n with the sub-graph}
12: **if** $|V| > M$ **then**
13: $H \leftarrow HD(H,M,S)$ {heuristic *HD* to reduce the size of H}
14: **end if**
15: **end while**
16: **return** s.selectPath(H) {phase 2}

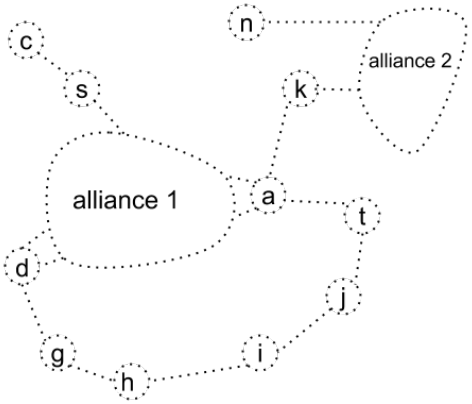
Figure 6.5 gives an example of the sub-graph construction performed by ACQA. Figure 6.5(A) shows of graph composed of ASes and alliances and Figure 6.5(B) shows the sub-graph initialized to *null*. In this example we set $M = 7$ and $r_0 = 1$ meaning that the maximum number of nodes in the sub-graph is equal to 7 and each node, either an AS or an alliance, knows all its neighbors within a distance equal to 1 respectively. We assume that destination node, AS t, has received the request q meaning that starting

condition of ACQA is fulfilled. The path followed by ACQA is [AS t, AS a, alliance 1, AS s].

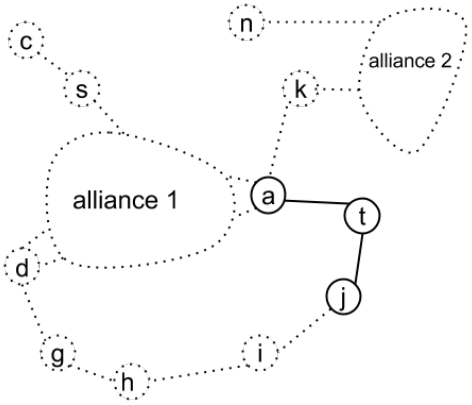
Once the destination has received the request, it merges its neighborhood shown by Figure 6.5(C) to the sub-graph which is initialized to *null*. The sub-graph at AS t is shown by Figure 6.5(D). Then, AS t sends the message containing the sub-graph to the next hop, i.e., AS a. Once AS a has received the message, it merges its neighborhood (Figure 6.5(E)) to the sub-graph under construction. The resulting sub-graph at AS a is shown by Figure 6.5(F). Then, AS a sends the message to alliance 1 that merges its neighborhood (Figure 6.5(G)) with the sub-graph under construction and sends the resulting sub-graph (Figure 6.5(H)) to AS s. When the source receives the message, it merges its neighborhood shown by Figure 6.5(I) to the sub-graph under construction and the resulting sub-graph is shown by Figure 6.5(J). We can remark that AS c is not added to the sub-graph. As it is specified in Line 11 in Algorithm 5, only the node with a degree greater than 1 are added to the sub-graph. At this step the source node owns a sub-graph composed of ASes and alliances and can use it to compute a set of feasible non-dominated paths. We develop the path search phase performed by ACQA in the next section.



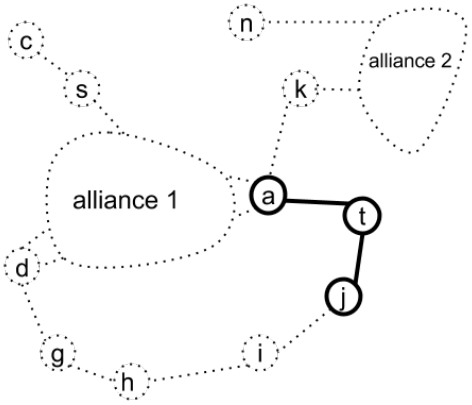
(A) Example of a graph



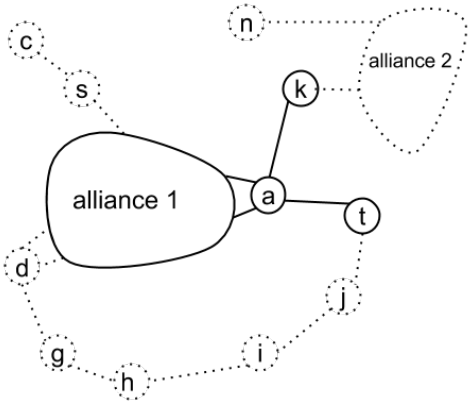
(B) Sub-graph initiliazed to null



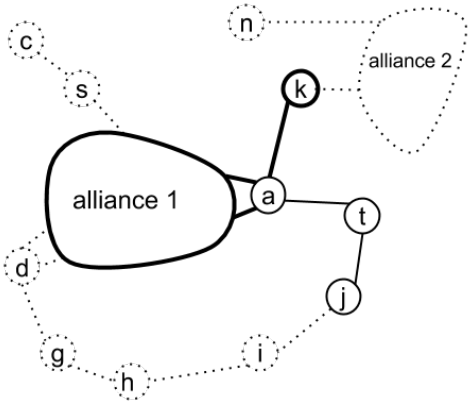
(C) Neighborhood of AS t



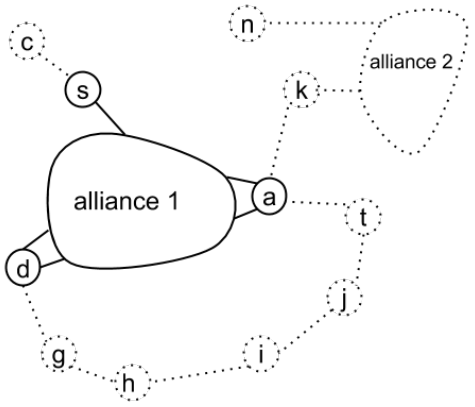
(D) Sub-graph at AS t



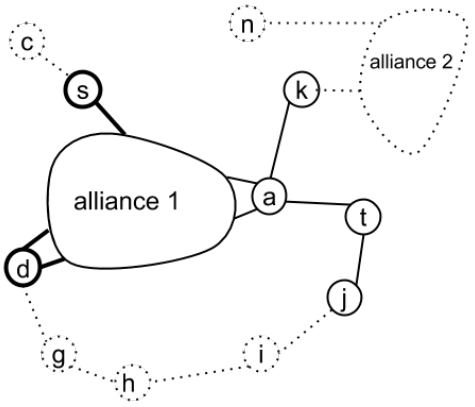
(E) Neighborhood of AS a



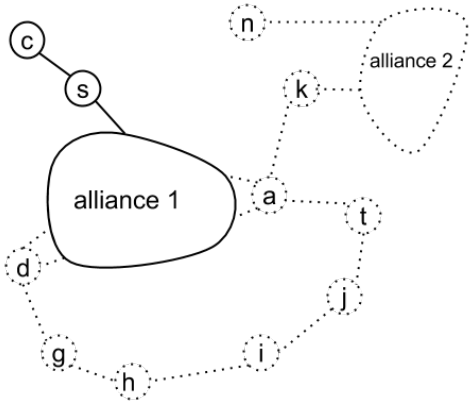
(F) Sub-graph at AS a



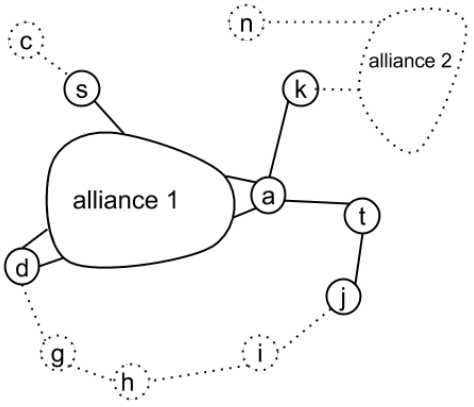
(G) Neighborhood of alliance 1



(H) Sub-graph at alliance 1



(I) Neighborhood of AS s



(J) Sub-graph at AS s

FIGURE 6.5

6.4.1 Finding Feasible Non-Dominated Paths in the Sub-Graph

At the end of phase 1 of ACQA, the source has a sub-graph containing both itself and the destination. The second phase of ACQA consists in searching a set of feasible non-dominated paths in the sub-graph. In this section, we describe the path search phase performed by ACQA. Recall that the sub-graph constructed by ACQA is composed of ASes and alliances. Hence, the path search phase performed by ACQA is different than the one performed by SANP. At the same time, we want to compare the performances of ACQA and SANP. More precisely, we want to compare the set of feasible non-dominated paths found with ACQA to the set of feasible non-dominated paths found with SANP. Hence, for the purpose of our simulations, we must determine how the alliances construct

their offers. In this dissertation, we assume that the offers published by an alliance are an example of an "open association" as defined by the ETICS project [12]: an alliance-offer is the composition (juxtaposition) of several AS-offers within the alliance and can be constructed by any of the solutions presented in Section 6.3. For the sake of simplicity, we propose two straightforwards methods to choose a chain of AS-offers between an entry and an exit points of an alliance. The selected chain of AS-offers corresponds to the alliance-offer. As these two methods do not require the exchange of critical information such as the internal topology of the ASes, they constitute a first step to build the alliance-offers. However, in the case of a high level a trust and thus a strong cooperation between the members of an alliance, more elaborated strategies can be set up.

6.4.1.1 Methods to Construct an Alliance-offer

***Shortest path* Method**

The *Shortest path* (*sp*) method selects the chain of AS-offers that minimizes the path length in term of number of ASes. This method reproduces the *hot potato* policy often implemented in the Internet. Let's take a simple example given by Figure 6.6 that represents an alliance composed of 3 ASes, AS a, b and c as well as two entry and exit points a_1 and b_1. In order to select a chain of AS-offers between a_1 and b_1, the *sp* method chooses the shortest path, in term of number of ASes, that is [AS a, AS b]. The chain of AS-offers starting from a_1 in AS a to b_1 in AS b is shown as dashed lines in Figure 6.6.

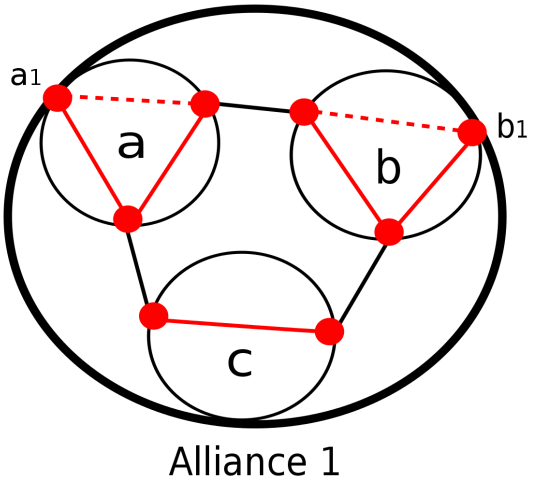


FIGURE 6.6: Chain of AS-offers selected with the *sp* method

ℓ^2 Method

The ℓ^2 method selects the chain of AS-offers that minimizes the path length following the ℓ^2 norm. If each AS-offer is characterized by two additives metrics such as the delay and the cost, we can use a plane to describe an AS-offer in which the x-axis (resp. y-axis) represents the delay (resp. cost), then the ℓ^2 method finds the chain of AS-offers that minimizes the distance with the origin. As an example, let consider Figure 6.7 that represents an alliance composed of three ASes: AS a, b and c as well as two entry and exit points a_1 and b_1. The chain of AS-offers that minimizes the path length between a_1 and b_1 following the ℓ^2 method is shown as a dashed line in Figure 6.7.

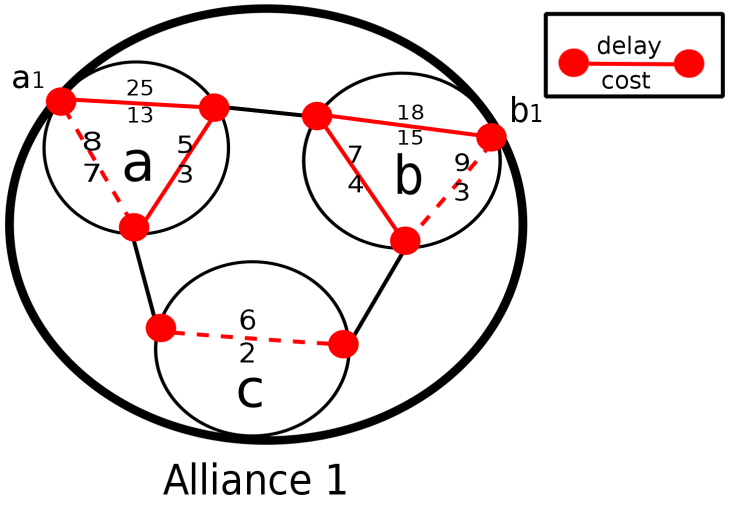


FIGURE 6.7: Chain of AS-offers selected with the ℓ^2 method

6.4.1.2 Finding Feasible Non-Dominated Paths Cases

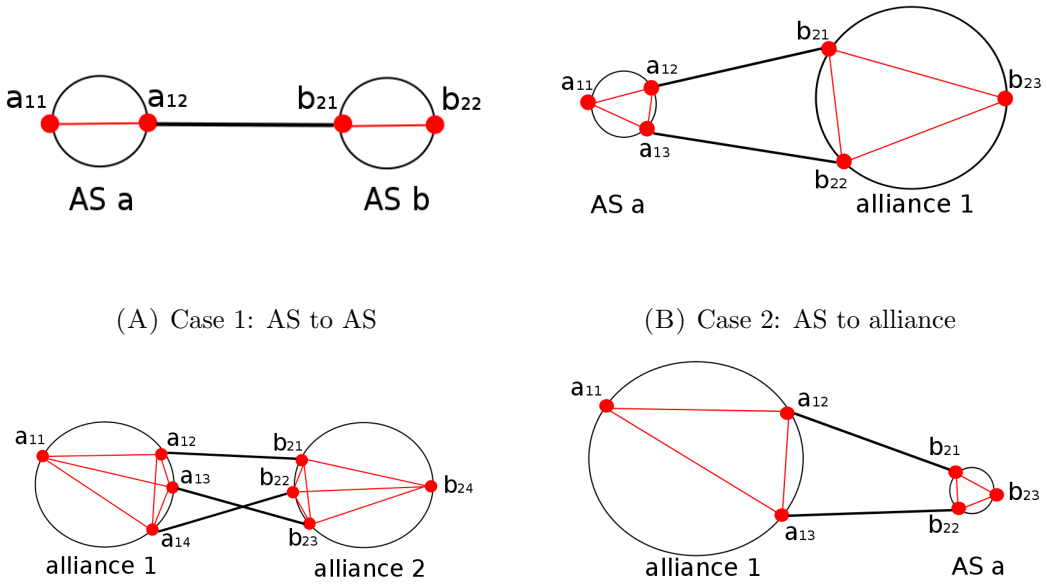
Now that we have shown how an alliance-offer is constructed, it is possible to compute a set of feasible non-dominated paths in the sub-graph. Like SANP, ACQA uses a brute-search force algorithm to list all the simple paths between the source and the destination and then computes the corresponding path weight. However, unlike SANP, which deals with a graph, ACQA deals with a multi-graph to find a set of non-dominated paths between the source and the destination. Furthermore, the multi-graph is composed of ASes and alliances leading to four cases for computing the cost of traversing a node. We now describe these cases.

Figure 6.8(A) shows the case in which ACQA computes the cost of traversing an AS and the next hop is also an AS. In this example, we want to compute the cost of traversing AS a from the entry point a_{11} to reach AS b. This case is handled in the same way as SANP: as explained in Section 5.4.3, only one offer is available to traverse an AS for reaching another AS.

Figure 6.8(B) shows the case in which ACQA computes the cost of traversing an AS and the next hop is an alliance. In this example, we want to compute the cost of traversing AS a form the entry point a_{11} to reach alliance 1. In this case, ACQA considers all the possible exit points of AS a to reach alliance 1. Here, ACQA considers both the exit point a_{12} and a_{13} and creates two paths, the first taking into account the AS-offers between a_{11} and a_{12} and the second between a_{11} and a_{13}. Recall that in our model, an ASBRI belonging to an AS or an edge-ASBRI belonging to an alliance is connected with only one another ASBRI or edge-ASBRI. Hence, a_{12} can not be connected at the same time with b_{21} and b_{22}.

Figure 6.8(C) shows the case in which ACQA computes the cost of traversing an alliance and the next hop is also an alliance. The difference with the previous example is that the first node is an alliance and not an AS. In this example, we want to compute the cost of traversing alliance 1 form the entry point a_{11} to reach alliance 2. In the previous case, ACQA considers all the possible exit points of Alliance 1 to reach Alliance 2. ACQA creates three paths, taking into account the alliance-offer between a_{11} and a_{12} first; then the alliance-offer between a_{11} and a_{13} and finally the alliance-offer between a_{11} and a_{14}.

Figure 6.8(D) shows the case in which ACQA computes the cost of traversing an alliance and the next hop is an AS. In this example, we want to compute the cost of traversing alliance 1 form the entry point a_{11} to reach AS 1. In the same way as the previous case, ACQA considers all the possible exit points of alliance 1 to reach AS a. Here, ACQA creates two paths, the first taking into account the Alliance-offer between a_{11} and a_{12} and the second taking into account the alliance-offer between a_{11} and a_{13}.



(A) Case 1: AS to AS

(B) Case 2: AS to alliance

(C) Case 3: alliance to alliance

(D) Case 4: alliance to AS

FIGURE 6.8: Cases occurring during the path search phase performed by ACQA

6.5 Simulations

In order to evaluate the performances of ACQA, we are going to compare it with SANP. To do so, we will describe three different scenarios, present the results and discuss the performances of the algorithms. The first scenario aims at assessing the performances of ACQA and SANP when the delay and/or the cost are the limiting factors. In the second scenario, we assess the performances of the algorithms when the bandwidth is the limiting factor by increasing the network load. Finally, in the third scenario we modify the delay and cost distribution in order to avoid that the shortest path has the greatest probability to have the lowest delay and/or cost.

For this, we are going to use the Python simulator presented in Section 5.4.1. Figure 6.9 gives a description of the different steps performed by the Python simulator to initialize the simulations for both ACQA and SANP.

The first step consists in creating a graph $G(V, E)$ where V is a set of nodes representing the ASes and E represents the inter-domain links. For that, we use the Inet

generator [98]. Recall that we use a complete graph to connect all the ASBRIs of an AS to represent the AS-offers. In the second step, the simulator associates one AS-offer to each intra-domain link. Each AS-offer is characterized by two additives metrics such as the delay and the cost and one min-max metric: the bandwidth. At this step, it is possible to run SANP on $G(V, E)$ to satisfy connection requests. The third step is needed only by ACQA. In this step, the simulator starts to transform $G(V, E)$ into a multi-graph $G_a(V_a, E_a)$ where V_a is a set of ASes and alliances and E_a is a multi-set of edges as described in Section 6.4. To do so, it uses one of the three algorithms described in the next section (Section 6.5.1) to create the alliances. Then, after choosing an alliance-offer construction metric as described in Section 6.4.1.1, it is possible to use ACQA to simulate connection requests. Recall that each request specifies the source and destination AS. However, we do not handle the connection requests for which the source and destination AS belong to the same alliance. It is possible to use whatever solution proposed in Section 6.3 to deal with this problem. We are now going to present the three alliances-construction algorithms.

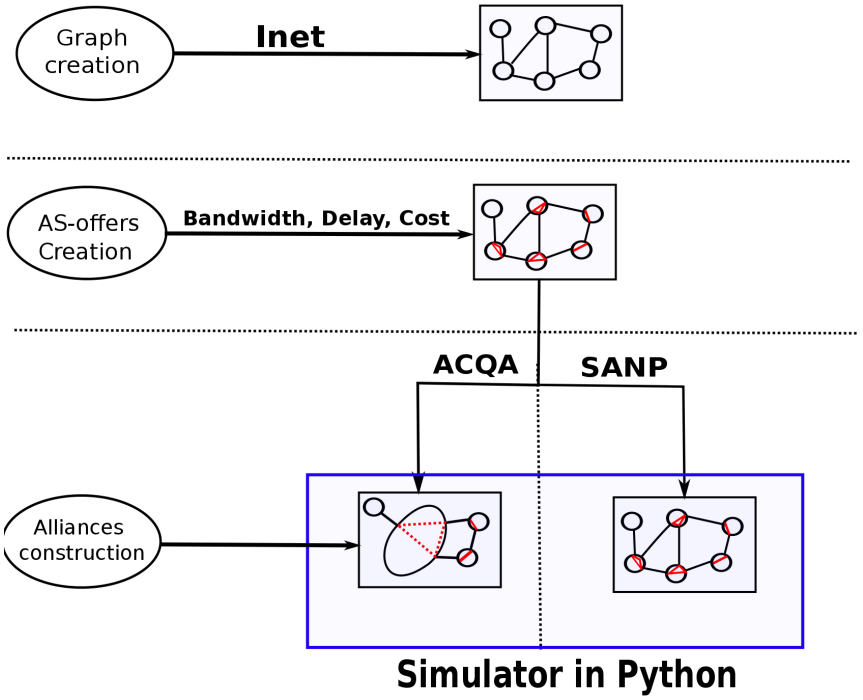


FIGURE 6.9: Simulation workflow

6.5.1 Alliance's Construction Algorithms

In order to construct the alliances, we have implemented two algorithms named DFS (Depth-First Search) and BFS (Breadth-First Search) and used the MCL (Markov CLuster) algorithm described in [130].

6.5.1.1 DFS Algorithm

DFS constructs N alliances composed each of exactly R nodes. Each node belongs to only one alliance and each alliance forms a connected graph. To do so, DFS starts to pick N "root nodes" corresponding the N nodes with the largest degree. Then for each root node, DFS adds its $R-1$ first neighbors ordered following a depth-first search algorithm that do not belong to any other alliance. This algorithm tends to construct alliances that are very stretched, i.e., the diameter of the alliance is very large. We define the diameter of an alliance as the longest shortest path between two nodes within the alliance.

Algorithm 6 gives the pseudo code of DFS. At line 2, it computes N nodes that correspond to the root nodes, one for each alliance. The *while* loop starting at line 3 is repeated for each alliance. At line 4, it starts to initialize the sub-graph with the corresponding root node. Then, it computes the neighbors of the root node following a depth-first search algorithm (line 5). By doing so, we ensure that the alliances form a connected sub-graph. Then, the loop starting at line 6 adds the $R-1$ neighbors of the root node. At line 12, it creates the sub-graph that correspond to the alliance. Finally, at line 13, it removes the nodes in the alliance just created to the graph. Hence, we ensure that a node does not belong to several alliances.

Pseudo Code

Algorithm 6 Creates N alliances of R nodes following a depth-first search algorithm
DFS(G,N,R):

1: $i \leftarrow 0$)
2: $B \leftarrow$RootNode(N) {RootNode(N) returns a list of N "root nodes".}
3: **while** $i < N$ **do**
4: $A_i \leftarrow (V_i \leftarrow b_i, E_i \leftarrow \emptyset)$ {initializes the sub-graphs of the alliances with the root node}
5: $tree \leftarrow$ DFSNodes(G,b_i) {DFSNodes(G,b_i) returns the list of ordered neighbors of b_i following a depth-first-search algorithm in G}
6: **while** $|V_i| < R - 1$ **do**
7: **if** $tree[0]$ not in RootNode(N) **then**
8: $V_i \leftarrow V_i \cup tree[0]$
9: $tree.remove(0)$ {remove the first node}
10: **end if**
11: **end while**
12: $A_i(V_i, E_i) \leftarrow$subgraphCopy(G,V_i) {subgraphCopy(G,V_i) creates a sub-graph composed of the nodes in V_i and all the edges between the nodes}
13: removeNodes(G,V_i) {removes the nodes in V_i of the graph G}
14: **end while**
15: **return** A

6.5.1.2 BFS Algorithm

BFS is similar to DFS, i.e., it creates N alliances of exactly R nodes. Like DFS, BFS starts to compute N "root nodes" corresponding the N nodes with the largest degree. However, unlike DFS that adds the neighbors ordered following a depth-first search algorithm, BFS adds the neighbors following a breadth-first search algorithm. Thus, BFS tends to construct "compact" alliances, i.e., the diameter of the alliances is very small.

Algorithm 7 gives the pseudo code of the construction of the alliances following a breadth-search first algorithm. The pseudo code of BFS is similar to DFS except the line 5 that orders the neighbors following a breath-first search algorithm.

Pseudo Code

Algorithm 7 Creates N alliances of R nodes following a breadth-first search algorithm **BFS(**G**,**N**,**R**)**:

1: $i \leftarrow 0$)
2: $B \leftarrow$RootNode(N)
3: **while** $i < N$ **do**
4: $A_i \leftarrow (V_i \leftarrow b_i, E_i \leftarrow \emptyset)$
5: $tree \leftarrow$ BFSNodes(G,b_i) {BFSNodes(G,b_i) returns the list of ordered neighbors of b_i following a breadth-first-search algorithm starting at source in G}
6: **while** $|V_i| < R - 1$ **do**
7: **if** $tree[0]$ not in RootNode(N) **then**
8: $V_i \leftarrow V_i \cup tree[0]$
9: $tree.remove(0)$
10: **end if**
11: **end while**
12: $A_i(V_i, E_i) \leftarrow$subgraphCopy(G,V_i)
13: removeNodes(G,V_i)
14: **end while**
15: **return** A

6.5.1.3 MCL Algorithm

The MCL algorithm [130] creates clusters within a graph. The authors define a cluster as a sub-graph in which each node has the majority of its neighbors within the sub-graph than are exterior to the sub-graph. In other words, by taking a node in a cluster and by randomly taking one of its neighbors, it is more likely that the neighbor belongs also to the same cluster than not. MCL uses a random walk to determine "attractive nodes" and "attracted nodes" for each cluster. An attractive node and its attracted nodes form an alliance. MCL ensures that a node cannot belong to different alliances and that each alliance is a connected graph. In the same way as DFS and BFS, MCL has the notion of "root node" around which a certain number of its neighbors join up to form an alliance. By doing this and by tuning the attractiveness of the attracted nodes with the help of a MCL parameter, it is easy to set the sizes and the number of the alliances.

6.5.2 Simulation Scenarios

For each simulation scenario, we are going to use Inet to generate a graph of 2000 nodes and simulate 1000 connection requests. Connection requests arrive according to a Poisson process, with an average of one connection each 20 seconds. We set the duration of the connection request long enough such that each request has started before the first one is terminated. Therefore, a connection request reserves the resources for the duration of the simulation. Each connection request specifies the source and destination ASes and the constraints for the delay, the cost and the bandwidth. When using the MCL algorithm, it builds nine alliances with 429, 188, 99, 65, 52, 42, 30, 28 and 25 nodes respectively. Hence, the total number of nodes belonging to an alliance is equal to 958.

In order to have approximately the same number of nodes within the alliances, when using the DFS and the BFS algorithms, we set N, the number of alliances, equal to 10 and R, the number of nodes within an alliance, equal to 90. DFS builds seven alliances with 90 nodes and the three left have 64, 60 and 47 nodes respectively. The reason why three alliances have a number of nodes less than 90 nodes is that DFS has to build alliances forming connected graph. DFS stops because no more nodes can be added while respecting this condition. BFS builds seven alliances with 90 nodes and the three left have 87, 85 and 67 nodes respectively. BFS also has to build alliances forming connected graph leading to alliances with a number of nodes less than 90.

We set $r_0 = 1$ and $\alpha = 50$. Recall that $\alpha \times L$, with L equal to the shortest path length between the source and the destination is equal to the maximum sub-graph' size. Moreover, for the sake of scalability, we set for both SANP and ACQA the maximum path length equal to 8 in term of number of ASes. For SANP, it is trivial to assess the number of AS traversed as the graph is composed only of ASes. However, ACQA computes a set of feasible paths at the alliance-level. In order to be fair, we must determine the length of the path in term of ASes computed by ACQA. For instance, let assume that ACQA has found the path [AS a, alliance 1, alliance 2] at the Alliance-level shown Figure 6.10. The path [AS a, alliance 1, alliance 2] at the alliance-level corresponds to the path [a, b, c, d, e, f] at the AS-level. As the path length at the AS-level is smaller than 8, it is considered valid.

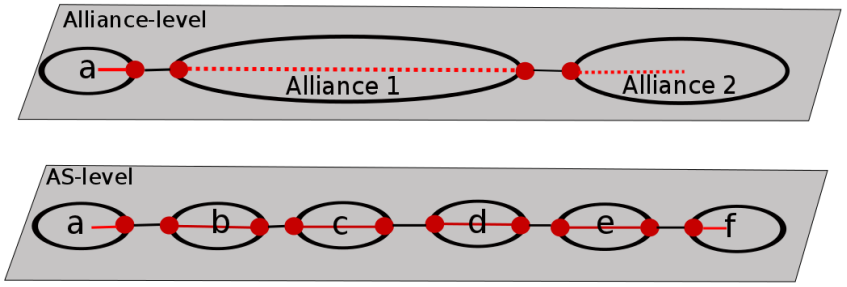


FIGURE 6.10: Example of path found

For each simulation, we want to determine the number of requests for which an algorithm is better than the other one. We consider that ACQA (resp. SANP) is **better** than SANP (resp. ACQA) if there is at least one path found by ACQA (resp. SANP) that dominates all the paths found by SANP (resp. ACQA). If ACQA (resp. SANP) is better than SANP (resp. ACQA), we consider that ACQA (resp. SANP) **wins** the request. Conversely, if SANP (resp. ACQA) is better than ACQA (resp. SANP) then ACQA (resp. SANP) **looses** the request. If neither ACQA is better than SANP nor SANP is better than ACQA, then the request is considered **neutral**. If only ACQA (resp. SANP) finds at least one path, the request is considered **only** for the benefit of ACQA (resp. SANP).

As the results are similar between the three construction alliances algorithms and between the two alliances-offers construction metrics, we only present results of the simulation using MCL as construction alliances algorithms and ℓ^2 as alliances-offers construction metric for the sake of clarity.

6.5.3 Influence of the Delay and Cost Constraints

In this section, we describe the first scenario for which we want to compare the paths found with ACQA and SANP when the delay and/or the cost are the limiting factor. We set both the delay and the cost of the intra-domains links to be uniformly distributed on $[1, 50]$ and the capacity equal to 1000. Moreover, we set the bandwidth constraint equal to 0 such that it is never the limiting factor. We run three simulations and in the first one, we set both the delay and the cost constraints equal to 1×10^6 such that they are never the limiting factor. In the second (resp. third) simulation, the delay and cost constraints are equal to 150 (resp. 100). For the first simulation we want to assess

both SANP and ACQA with loose constraints, i.e., when both have reached either the destination or the maximum path length, not because the constraints are not satisfied.

Delay and Cost are not the Limiting Factors

Figure 6.11 shows a summary of the requests when the delay and/or cost are not the limiting factors. Clearly, the vast majority of the requests are neutrals. ACQA wins 22 requests, SANP wins 5 requests and both ACQA and SANP satisfy the 1000 requests. In order to characterize the 973 $(1000-(22+5))$ neutrals requests for which it is impossible to directly affirm which one is the best algorithm, we are going to utilize the Zitzler and the Generational distance metrics described in Section 5.4.4.1.

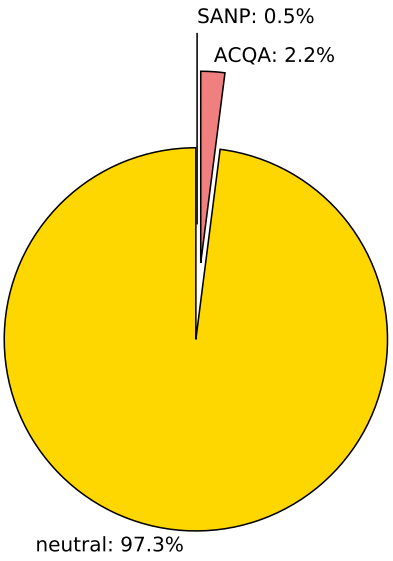


FIGURE 6.11: Summary of the requests when the delay and cost are not the limiting factors

Figures 6.12 and 6.13 show the CDF of the Zitzler and the Generational distance metrics respectively for the 971 neutral requests. $C(SANP, ACQA)$ is always above that $C(ACQA, SANP)$ meaning that the paths found with ACQA tend to dominate the paths found with SANP. For sake of clarity, when computing the CDF of $G(SANP, ACQA)$ and $G(ACQA, SANP)$, we do not take into account the cases where $G(SANP, ACQA) = 0$ and $G(ACQA, SANP) = 0$. $G(SANP, ACQA) = 0$ (resp $G(ACQA, SANP) = 0$) means that no solution found with SANP (resp ACQA) taken individually dominates or is dominated by its closest solution found with ACQA (resp SANP). As we are interested

in the distance between two paths from different sets with one that dominates the other, we can skip this information.

First, the CDF of $G(SANP, ACQA)$ is negative for 70% of the cases while $G(ACQA, SANP)$ is negative for 30% of the cases confirming that the paths found by ACQA tend to dominate the paths found by SANP. Second, given that the average cost (and delay) of a link is 25, by looking at the x-axis we can conclude that, when $G(SANP, ACQA)$ is negative (i.e., the solution found by ACQA dominates the solution found by SANP), the path found by ACQA is relatively better than the one found by SANP.

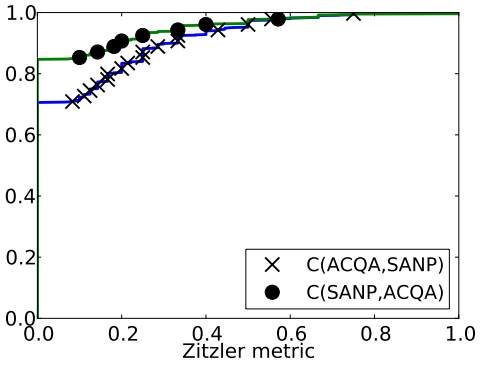


FIGURE 6.12: Zitzler CDF when the delay and the cost are never the limiting factors

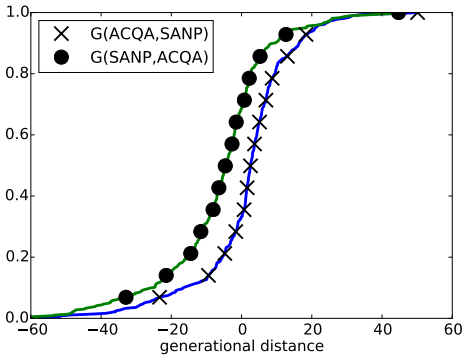


FIGURE 6.13: Generational Distance CDF when the delay and the cost are never the limiting factors

Delay and Cost Constraints equal to 150

Figure 6.14 shows the summary of the requests when the delay and cost constraints are equal to 150. Again, the vast majority of the requests are neutrals. Among the 891 requests for which both ACQA and SANP find a solution, ACQA wins 13 requests and SANP wins 9 requests. For 26 requests (resp 10) only ACQA (resp SANP) finds a solution. In order to characterize the 869 $(891 - (13 + 9))$ neutrals requests, we utilize the Zitzler and the Generational distance metrics.

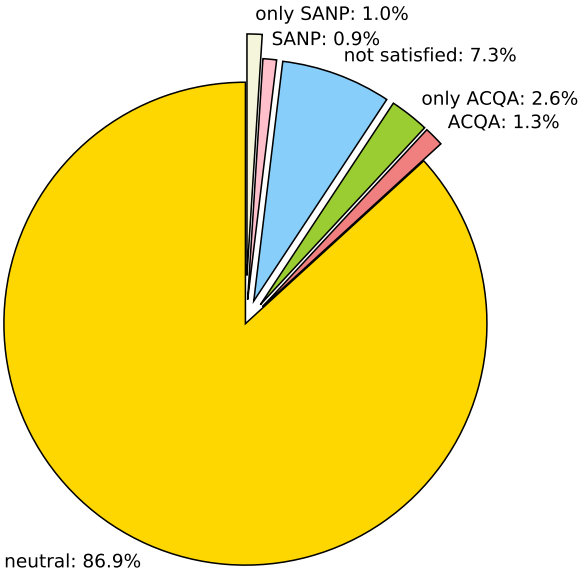


FIGURE 6.14: Summary of the requests when the delay and cost constraints = 150

Figure 6.15 and Figure 6.16 show the CDF of the Zitzler and the Generational distance metrics respectively for the 855 neutral requests. Figure 6.15 shows that $C(ACQA, SANP)$ is slightly better than $C(SANP, ACQA)$ and Figure 6.16 shows that $G(ACQA, SANP)$ is slightly greater than $G(SANP, ACQA)$ meaning that the paths found with ACQA are slightly better than SANP.

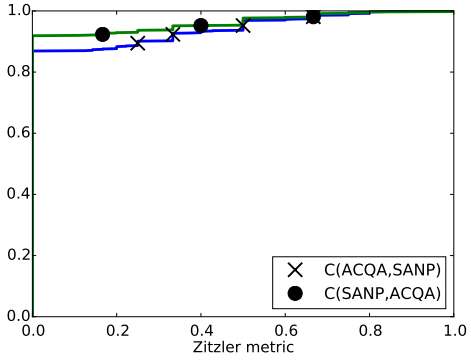


FIGURE 6.15: Zitzler CDF with the delay and cost constraints = 150

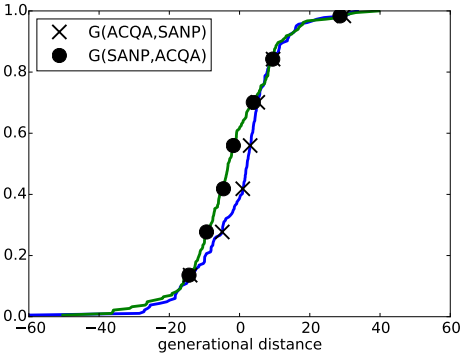


FIGURE 6.16: Generational Distance CDF with the delay and cost = 150

Delay and Cost Constraints equal to 100

Figure 6.17 shows the summary of the requests when the delay and cost constraints are equal to 100. As the constraints are very strict, there is only 283 requests for which

both ACQA and SANP find a solution. Among these requests, ACQA wins 5 times and SANP wins 2 times. For 23 requests (resp 10) only ACQA (resp SANP) finds a solution. For the 276 $(283 - (5 + 2))$ neutrals requests, we do not show the corresponding Zitzler and the Generational distance CDF as both are almost equal to 0. The very strict constraints limit the number of solutions restraining the possibility to dominate or to be dominated and thus $C(SANP, ACQA) \approx 0$ and $C(ACQA, SANP) \approx 0$.

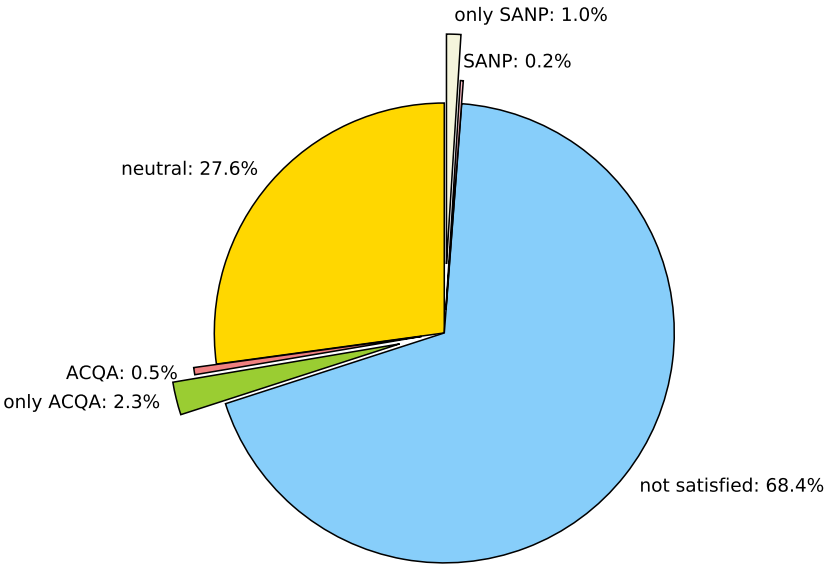


FIGURE 6.17: Summary of the requests when the delay and cost constraints = 100

In the case of the loose constraints, ACQA outperforms SANP. The explication lies on the fact that the alliances can offer a greater diversity of paths in term of ASes. However, when the delay and/or the cost are the limiting factors, as the tighter the constraints are, the closer the paths found with ACQA and SANP are.

6.5.4 Influence of the Bandwidth Constraint

In this section, we want to compare the paths found with ACQA and SANP when the bandwidth is the limiting factor. We set both the delay and the cost of the intra-domains links uniformly distributed on $[1, 50]$ and the capacity equal to 1000. The delay and the cost of each connection requests are equal to 1×10^6, such that they are never the limiting factor. We run three simulations with the bandwidth constraint uniformly distributed on $[200, 400]$, $[400, 800]$ and equal to 999 respectively. Recall that a request

reserves the bandwidth for the duration of the simulation. Therefore, the case where the bandwidth constraint is equal to 999 correspond to the case of each link can support only one connection since the capacity of each link is equal to 1000.

Bandwidth Request Uniformly Distributed on $[200, 400]$

Figure 6.18 shows the summary of the requests when the bandwidth constraint is uniformly distributed on $[200, 400]$. Among the 951 requests for which both ACQA and SANP find a solution, ACQA (resp. SANP) is better for 22 (resp. 10) requests. For 17 requests (resp. 5) only ACQA (resp. SANP) finds a solution. We utilize the Zitzler and the Generational distance metrics to characterize the 919 $(951 - (22 + 10))$ neutrals requests.

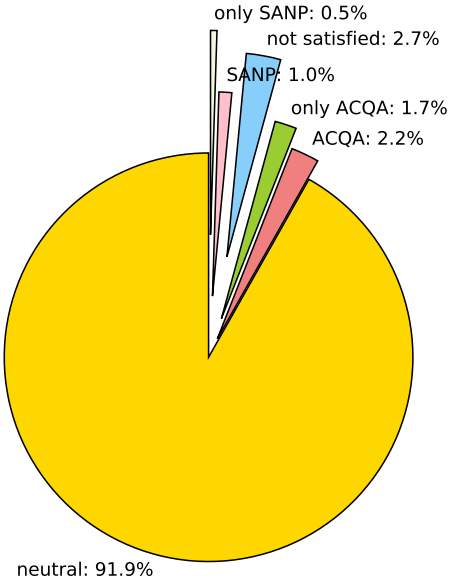


FIGURE 6.18: Summary of the requests with the bandwidth constraint uniformly distributed on $[200, 400]$

Figures 6.19 and 6.20 show the CDF of the Zitzler and the Generational distance metrics respectively for the 919 aforementioned requests. $C(ACQA, SANP)$ is below $C(SANP, ACQA)$ meaning that the paths found with ACQA tend to dominate the paths found with SANP. Given that the average cost (and delay) of a link is 25, Figure 6.20 shows that the paths found with ACQA are clearly better than the ones found by SANP.

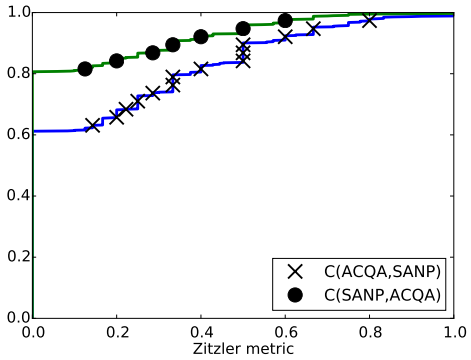


FIGURE 6.19: Zitzler CDF with the bandwidth constraint uniformly distributed on $[200, 400]$

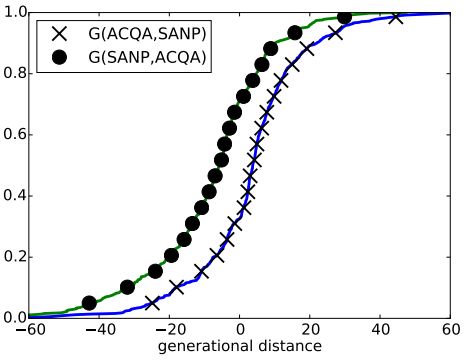


FIGURE 6.20: Generational distance CDF with the bandwidth constraint uniformly distributed on $[200, 400]$

Bandwidth Request Uniformly Distributed on $[400, 800]$

Figure 6.21 shows the summary of the requests when the bandwidth constraints is uniformly distributed on $[400, 800]$. Among the 592 requests for which both ACQA and SANP find a solution, ACQA (resp. SANP) is better for 34 (resp. 14) requests. For 74 requests (resp 41) only ACQA (resp SANP) finds a solution.

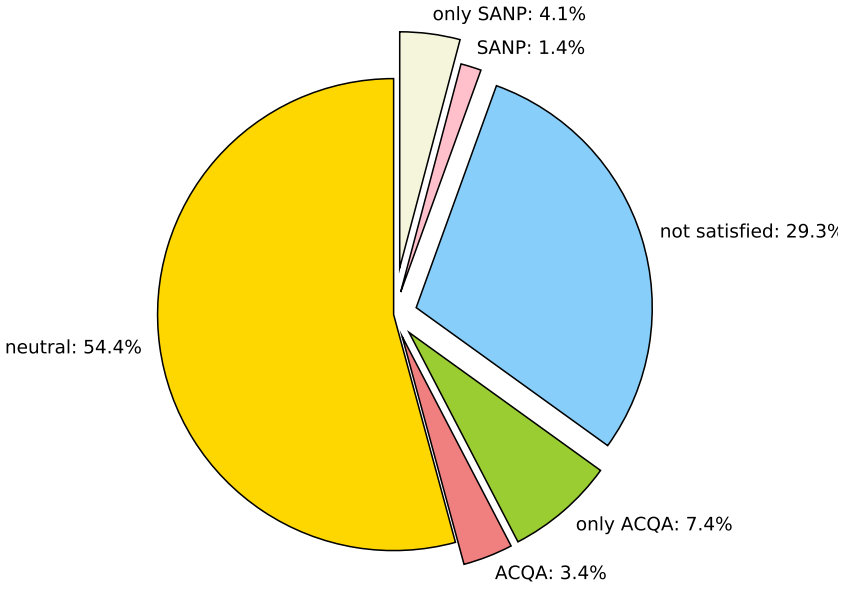


FIGURE 6.21: Summary of the requests with the bandwidth constraint uniformly distributed on $[400, 800]$

Figure 6.22 and Figure 6.23 show the CDF of the Zitzler and the Generational distance metrics respectively for the 544 $(592-(34+14)$ neutrals requests. As $C(ACQA, SANP)$ is distinctly below $C(SANP, ACQA)$, meaning that the paths found with ACQA tend to dominate the paths found with SANP, Moreover, Figure 6.23 shows that paths found with ACQA are clearly better than the paths found with SANP.

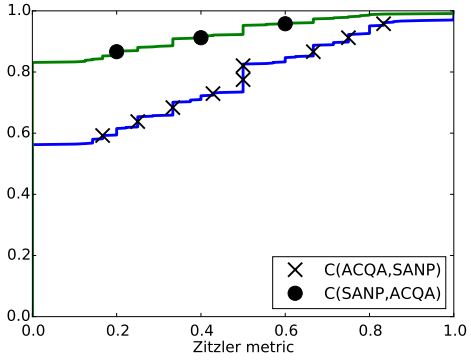


FIGURE 6.22: Zitzler CDF with the bandwidth constraint uniformly distributed on $[400, 800]$

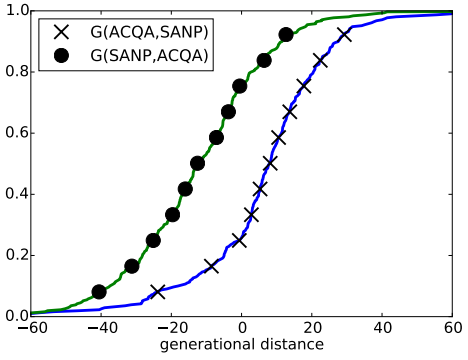


FIGURE 6.23: Generational distance CDF with the bandwidth constraint uniformly distributed on $[400, 800]$

Bandwidth Request Equal to 999

Figure 6.24 shows the summary of the requests when the bandwidth constraints is equal to 999 i.e. a link can support only one connection. Among the 566 requests for which both ACQA and SANP find a solution, ACQA (resp. SANP) is better for 42 (resp. 13) requests. For 94 requests (resp 49) only ACQA (resp SANP) finds a solution.

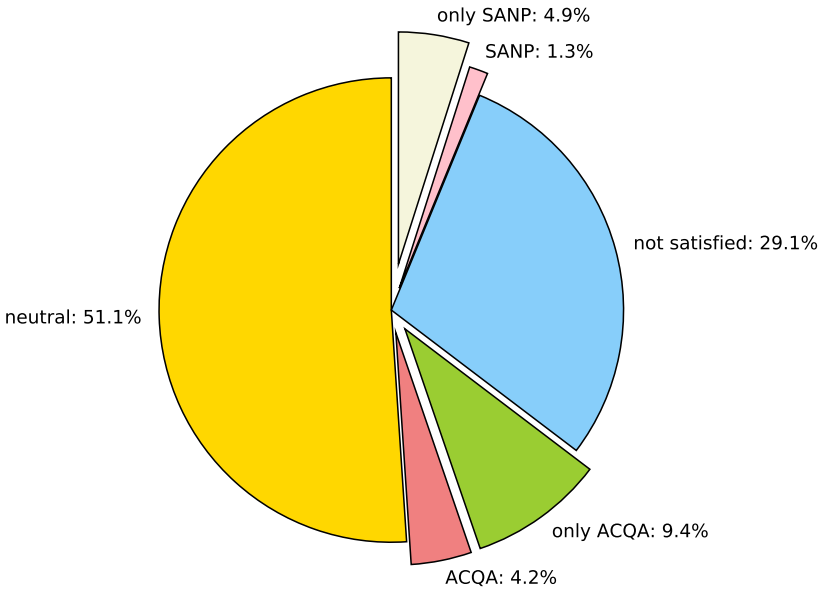


FIGURE 6.24: Summary of the requests when the bandwidth constraint = 999

Figure 6.25 and Figure 6.26 show the CDF of the Zitzler and the Generational distance metrics respectively for the 511 $(566 - (42 + 13))$ neutrals requests. Like the other scenarios, ACQA outperforms SANP.

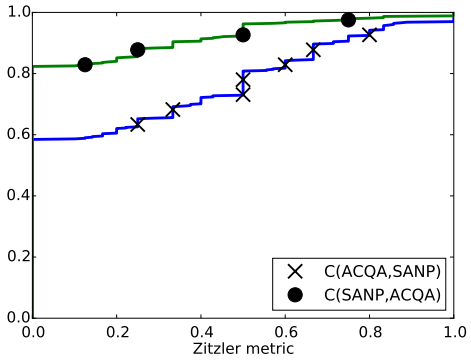


FIGURE 6.25: Zitzler CDF with the bandwidth constraint = 999

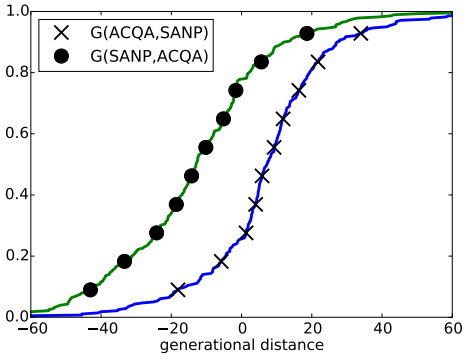


FIGURE 6.26: Generational distance CDF with the bandwidth constraint = 999

As in the case in which the delay and the cost were the limiting factor, the case in which the bandwidth is the limiting factor, we have shown by simulations that ACQA outperforms SANP.

6.5.5 Influence of the Delay and Cost Distributions

In this section, we want to compare the path quality found with ACQA and SANP when the delay and the cost are not uniformly distributed. In order to avoid that the shortest path has the greatest probability to have the lowest delay and/or cost, we do not use a uniform distribution like $[1, c]$, with $c > 1$, for the delay and the cost. As both the delay and the cost are additive metrics, the lower the number of links a path has, the higher the probability to have a low delay and/or cost is.

Hence, we set the delay and cost distributions as follow: for 30% of the cases, the delay (resp. the cost) is distributed on $[1, 10]$ and the for the remaining 70% of the cases, the delay (resp. the cost) is distributed on $[1000, 1500]$. By doing this, some links with either a low cost or a low delay or both are scattered all over the graph. Hence, a path has a high probability to be a feasible non-dominated path if it has link(s) with low delay and/or cost rather than a small number of links. We run three simulations with delay and cost equal to 4000, 5000 and 10×10^6 respectively. The third one corresponds to the case for which the delay and the cost are never the limiting factor and thus ACQA and SANP stop only when they have reached either the destination or the maximal path length.

Delay and Cost are Never the Limiting Factors

Figure 6.27 shows the summary of the requests when the delay and cost are never a limiting factor. Both ACQA and SANP satisfy all the requests and ACQA (resp SANP) wins 38 (resp 6) requests.

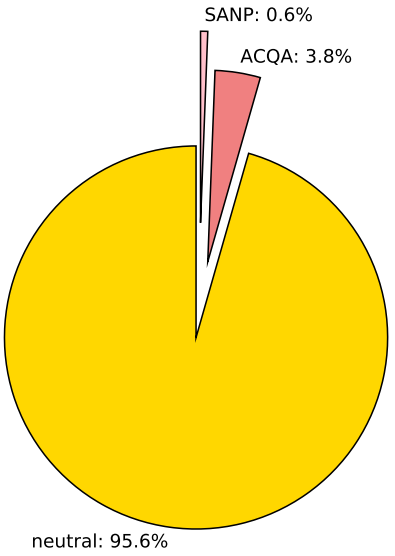


FIGURE 6.27: Summary of the requests when the delay and cost are never the limiting factors

Figures 6.28 and 6.29 show the CDF of the Zitzler and the Generational distance metrics respectively for the 956 $(1000 - (38 + 6)$ neutrals requests

By comparing the results when the delay and the cost are uniformly distributed without limiting constraints (Figure 6.12 and 6.13), we can remark that the paths found with ACQA tend to dominate the paths found with SANP.

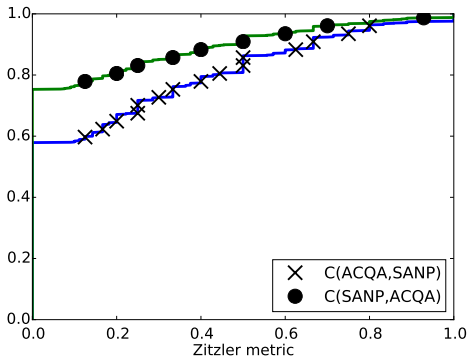


FIGURE 6.28: Zitzler CDF when the delay and cost are never the limiting factors

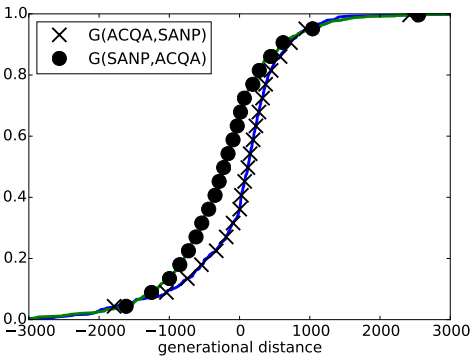


FIGURE 6.29: Generational distance when the delay and cost are never the limiting factors

Delay and Cost Constraints Equal to 5000

Figure 6.30 shows the summary of the requests when the delay and cost constraints are equal to 5000. Among the 822 requests for which both ACQA and SANP find a solution, ACQA wins 49 requests and SANP wins 22 requests. For 47 requests (resp 24) only ACQA (resp SANP) finds a solution.

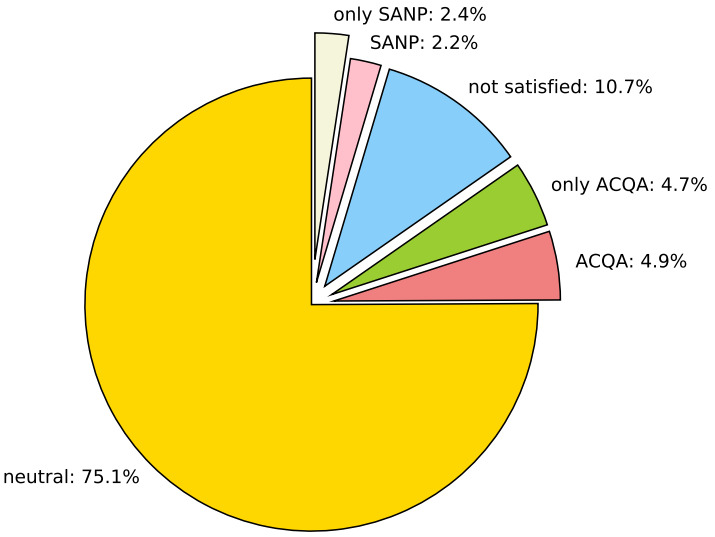


FIGURE 6.30: Summary of the requests when the delay and cost = 5000

Figures 6.31 and 6.32 show the CDF of the Zitzler and the Generational distance metrics respectively for the 751 $(822-(49+22))$ neutrals requests. $C(SANP, ACQA)$ is always above $C(ACQA, SANP)$, thus the paths found with ACQA dominate the paths found with SANP.

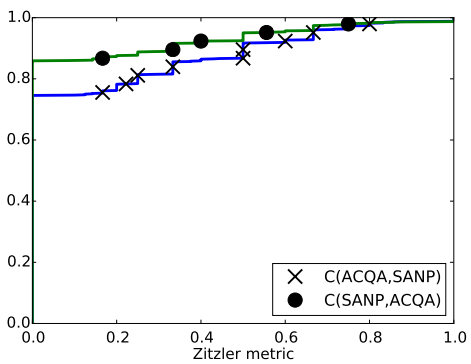


FIGURE 6.31: Zitzler CDF when the delay and cost constraints = 5000

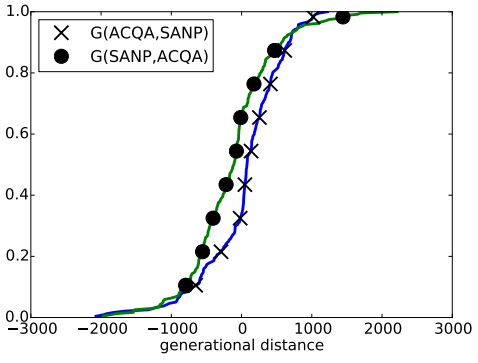


FIGURE 6.32: Generational distance when the delay and cost constraints = 5000

Delay and Cost Constraints Equal to 4000

Figure 6.33 shows the summary of the requests when the delay and cost constraints are equal to 4000. Among the 561 requests for which both ACQA and SANP find a solution, ACQA wins 25 requests and SANP wins 14 requests. For 81 requests (resp 22) only ACQA (resp SANP) finds a solution.

As $C(SANP, ACQA)$ and $C(ACQA, SANP)$ are almost equal to 0 for the 519 $(557 - (20 + 18))$ neutral requests we do not show the CDFs of the Zitzler nor the Generational distance metrics. $C(SANP, ACQA)$ and $C(ACQA, SANP)$ are almost equal to 0 due to very strict constraints limiting the number of solutions and thus $C(SANP, ACQA) \approx 0$ and $C(ACQA, SANP) \approx 0$.

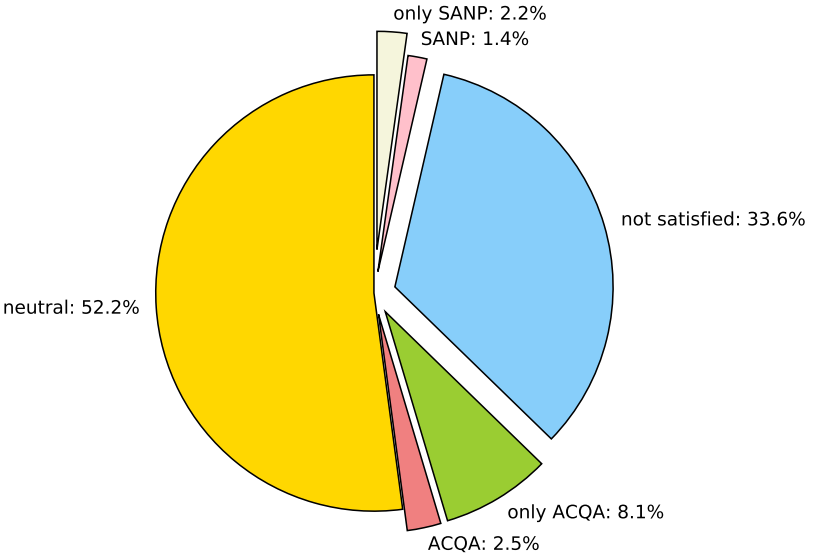


FIGURE 6.33: Summary of the requests when the delay and cost = 4000

To conclude, by changing the distributions of the delay and the cost in order to avoid that the shortest path in term of number of ASes has the greatest probability to be a non-dominated path, we improve the performances of ACQA compared to SANP.

Summary of the Simulations

Table 6.1 shows a summary of the simulations run in this chapter. In each simulation, the lines corresponds to the different scenarios and the columns corresponds to the different results of a request. By comparing the column ACQA with the column SANP

as well as the column only ACQA with column only SANP, it appears clearly that ACQA outperforms SANP.

		Neutral	Not satisfied	ACQA	SANP	only ACQA	only SANP
Delay and cost	No constraints	971	0	22	5	0	0
constraints in	150	855	109	13	9	10	10
simulation	100	250	717	5	2	23	10
Section 6.5.3							
Bandwidth	[200, 400]	919	27	22	10	17	5
constraint in	[400, 800]	544	293	34	14	74	41
simulation	999	511	434	42	13	94	49
Section 6.5.4							
Delay and cost	No constraints	956	0	38	6	0	0
constraints in	5000	751	178	49	22	47	24
simulation	4000	519	443	20	18	71	26
Section 6.5.5							

TABLE 6.1: Summary of the Simulations

6.5.6 Scalability

In this section we want to assess the ability of ACQA to scale up compared to SANP. Generally, the routing algorithm's scalability is assessed in terms of both space and time complexity. Concerning SANP and ACQA, the space complexity is related to the size of the message carried from the destination to the source along the shortest path. The essential information contained within the message is the sub-graph itself. Hence, the size of the message increases with the size of the sub-graph. Thus, in a first part, we are going to focus on the size of the sub-graphs constructed by both SANP and ACQA.

The time complexity of both SANP and ACQA is directly related to the number of offers explored during the path search phase. Hence, in a second part, we focus on the number of offers explored for each simulation presented above.

6.5.6.1 Space Complexity

As previously mentioned, in order to assess the space complexity of both SANP and ACQA, we compute the size of the sub-graphs constructed by the algorithms. As the sizes the sub-graph constructed by both SANP and ACQA are very similar for all the simulations presented above, we are going to present only the first simulation (Section 6.5.3).

Figure 6.34 shows the CDFs of the size of the sub-graphs constructed by SANP and by ACQA using the MCL, DFS and BFS algorithms. Recall that the size of the sub-graph is limited for both SANP and ACQA to $M = \alpha \times L$ with $\alpha = 50$ in our simulations and L equal to the length of the shortest path between the destination and the source. In the case of SANP, we consider each AS as a node while in the case of ACQA, both the alliances and the ASes are considered as a node. We can note that there are fewer nodes in the sub-graphs constructed by ACQA than SANP.

The average of the shortest paths length followed by SANP is equal to 4.64. Considering ACQA, it is equal to 3.28 (resp. 2.94, 3.33) with MCL (resp. BFS, DFS) alliance construction algorithm. In other words, $L_{ACQA} < L_{SANP}$ then $M_{ACQA} < M_{SANP}$ and thus the sub-graphs constructed by ACQA are smaller than the sub-graphs constructed by SANP.

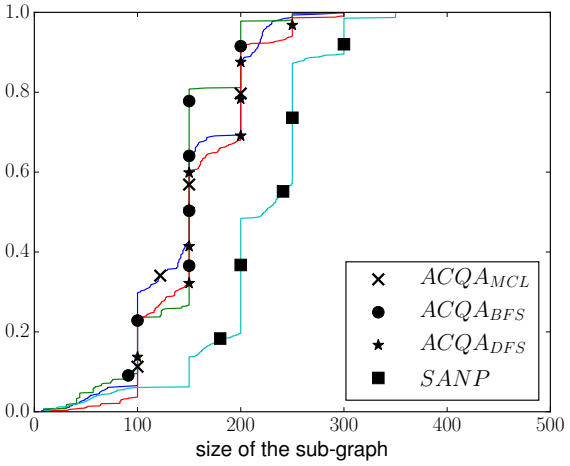


FIGURE 6.34: CDF of the sub-graph' size

6.5.6.2 Time Complexity

As previously mentioned, the time complexity of both SANP and ACQA is directly related to the number of offers explored during the path search phase performed by the algorithms. Hence, in the section, we are going to compute the number of offers explored for each simulation for both SANP and ACQA.

Simulation in Section 6.5.3

We focus on the number of offers explored in the simulation scenario presented in Section 6.5.3, in this scenario, the delay and the cost are the limiting factors.

Figure 6.37 (resp. Figure 6.36, Figure 6.35) shows the CDF of the number of offers explored when the delay and the cost constraints are equal to 100 (resp. 150, not a limiting factor) for both SANP and ACQA.

First, in Figure 6.37, the CDFs equal to 0 means that no solution has been found for a specific request. Second, as expected, the stricter the constraints are, the fewer the offers are explored. Figure 6.37 show that SANP and ACQA are of the same order of magnitude. However, in the case when the constraints are less strict (Figure 6.36)) or with loose constraints (Figure 6.35) ACQA is one order of magnitude greater than SANP. As the scenarios in which the cost and the delay are limiting factors is the most realistic, we can conclude that ACQA is still as scalable as SANP in this scenario.

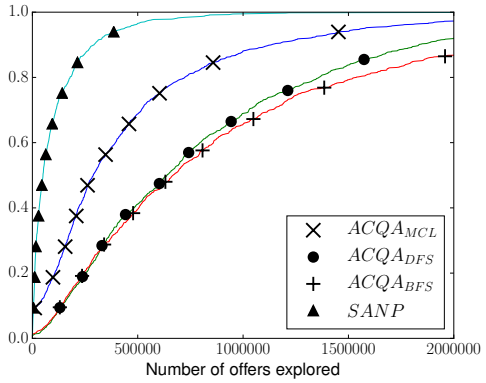


FIGURE 6.35: CDF of the number of requests explored for the simulation presented in Section 6.5.3 when delay and cost are not a limiting factors

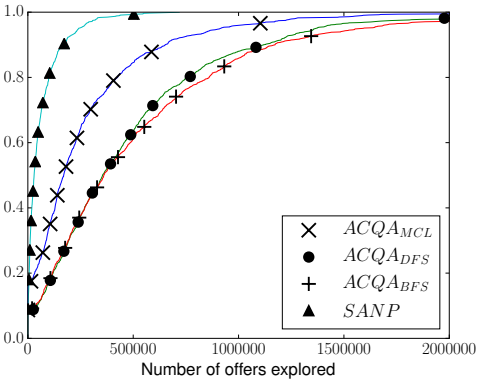


FIGURE 6.36: CDF of the number of requests explored for the simulation presented in Section 6.5.3 with delay and cost constraints = 150

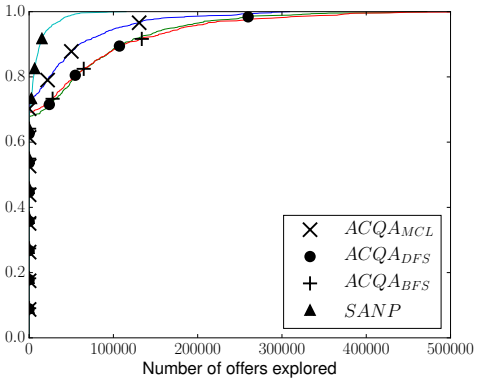


FIGURE 6.37: CDF of the number of requests explored for the simulation presented in Section 6.5.3 with delay and cost constraints = 100

Simulation in Section 6.5.4

We now focus on the number of offers explored in the simulation scenario presented in Section 6.5.4, in which the bandwidth is the limiting factor. Figure 6.38 (resp. Figure 6.39, Figure 6.40) shows the CDF of the number of offers explored when the bandwidth constraints is uniformly distributed on $[200, 400]$ (resp. $[400, 800]$, 999).

The results as similar to the results the previous simulation. First, as expected, the tighter the bandwidth constraint is, the small the number of offers explored. Second, ACQA is one order of magnitude greater than SANP.

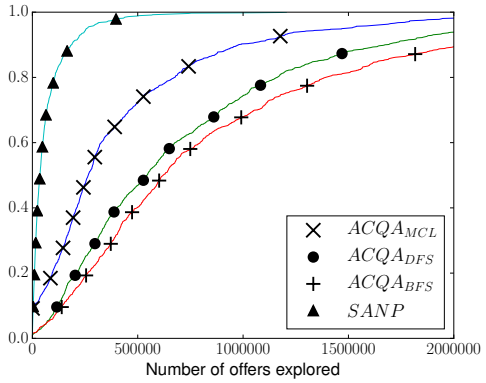


FIGURE 6.38: CDF of the number of requests explored for the simulation presented in Section 6.5.4 with bandwidth constraint uniformly distributed on $[200, 400]$

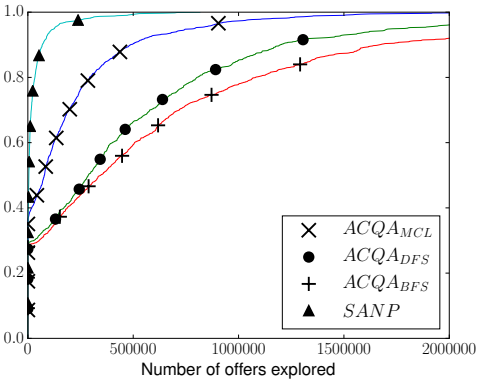


FIGURE 6.39: CDF of the number of requests explored for the simulation presented in Section 6.5.4 with bandwidth constraint uniformly distributed on $[400, 800]$

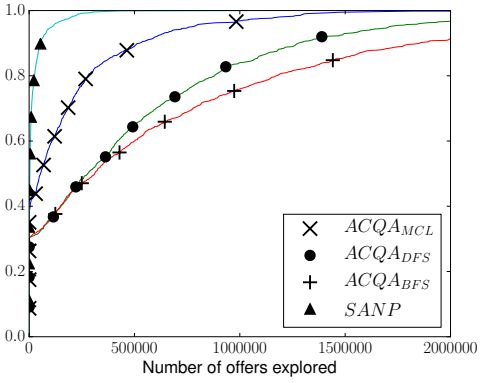


FIGURE 6.40: CDF of the number of requests explored for the simulation presented in Section 6.5.4 with bandwidth constraint = 999

Simulation in Section 6.5.5

We now focus on the number of offers explored in the simulation scnenario presented in Section 6.5.5. in which the delay and cost are not uniformly distributed in order to avoid giving the advantage to the shortest path.

Figure 6.41 (resp. Figure 6.42, Figure 6.43) shows the CDF of the number of offers explored when the delay and the cost constraints are never a limited factor (resp. equal to 5000, 4000). Again, as expected, the tighter the constraints are, the smaller the number of offers explored. As the scenarios for which the delay and the cost are the

limiting constraints are the most realistic (Figures 6.42,6.43), in this simulation, SANP and ACQA are of the same order of magnitude.

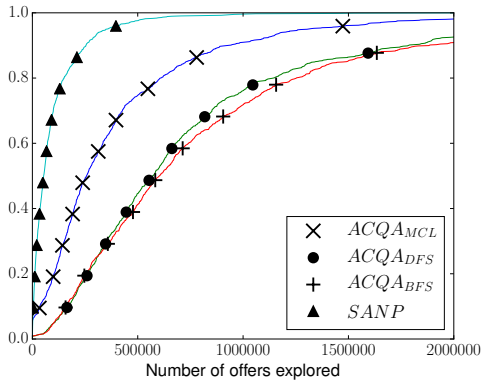


FIGURE 6.41: CDF of the number of requests explored for the simulation presented in Section 6.5.5 when the delay and the cost are never the limiting factor

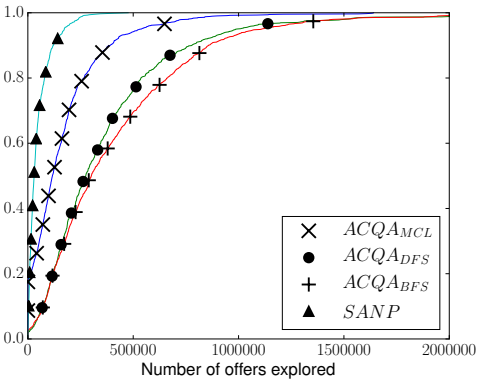


FIGURE 6.42: CDF of the number of requests explored for the simulation presented in Section 6.5.5 when the delay and the cost constraints = 5000

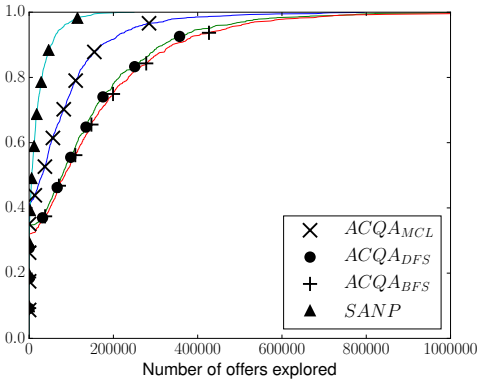


FIGURE 6.43: CDF of the number of requests explored for the simulation presented in Section 6.5.5 when the delay and the cost constraints = 4000

6.5.6.3 Concluding Remarks on the Scalability

Concerning the space complexity, ACQA and SANP are of the same order of magnitude, i.e., the sub-graphs constructed by both the algorithms have similar sizes. However, concerning the time complexity, i.e., the number of offers explored, ACQA is one order of magnitude greater than SANP for most of the scenarios. The main reason is that we use a brute-force search algorithm for searching a set of feasible non-dominated paths for both SANP and ACQA. The evident drawback of using a brute-force search algorithm

is its large complexity. However, its main advantage is that it makes no assumptions regarding to the QoS metrics: they are taking into account equivalently in order to compute end-to-end QoS paths. Plenty of solutions (presented in Section 3.3.3),that make assumptions regarding to the QoS metrics, have been proposed to not use a brute-force search algorithm to find end-to-end QoS paths.

6.6 Conclusion

In this chapter we have presented ACQA, an algorithm capable of computing end-to-end QoS paths in a graph comprised of alliances and ASes. Both ASes and alliances publish a list of offers describing the QoS guarantees that they can offer between their entry and exit points (ASBRIs). By constructing a sub-graph containing some of the nodes between the source and the destination, ACQA can compute a set of feasible non-dominated paths. The source can then select one of these paths based on its preferences.

By simulations, we have shown that for the vast majority of the scenarios ACQA outperforms SANP in terms of quality of paths and in terms of number of request satisfied. The main reason is that alliances can offer greater path diversity in terms of ASes but as well as in terms of offers (Figure 6.35 to Figure 6.43). As building alliances is more likely to better satisfy the customer's requests than independent ASes, the revenues of provisioning end-to-end services is higher for alliances than independent ASes. These results encourage the ASes to form alliances.

Furthermore, we have also shown that given that the number of nodes in the sub-graph is of the same order of magnitude in ACQA and SANP, ACQA is still as scalable as SANP. However, because we use a brute search force algorithm to find a set of feasible non-dominated paths within the sub-graph, the number of offers explored using ACQA is larger than using SANP. Nonetheless, we let to the ASes and alliances the choice to implement the algorithm it seems to fit to find a set of feasible paths in the graph. For instance, if an alliance is interesting in finding the cheapest cost without having consideration about the other QoS metrics, it can use the Dijkstra or Bellman-Ford algorithm. By doing so, the complexity of the path search phase is greatly improved compare to the brute-force search algorithm we use in our simulations.

Chapter 7

Conclusion

7.1 Thesis Outcomes

The problem of finding paths subject to multiple constraints covers many areas of research. As far as communication networks are concerned, finding paths with guaranteed performance have been largely studied since the emergence of the Internet. This problem raises many issues both at the technical and at the business levels, which are presented in Chapter 3. We have shown that the nature of the Internet is the main obstacle to offer end-to-end QoS. Indeed, the Internet is composed of a collection of ASes for which both the confidentiality and the autonomy properties must be respected. Moreover, any solution aiming at offering end-to-end QoS must be scalable.

The first main contribution of this thesis is SANP, presented in Chapter 5. It is an algorithm whose goal is to compute a set of feasible non-dominated paths satisfying given QoS constraints across several ASes. SANP assumes that the ASes are ready to publish offers between their edge nodes. It is important to note that in the services the ASes are ready to offer, no critical information have to be revealed. Only QoS guarantees need to be specified between entry and exit points, thus respecting the confidentiality of the ASes. Moreover, for the sake of autonomy, an AS would not need to announce how it implements its offers. SANP constructs a sub-graph around the route used by existing routing mechanisms between the source and destination. This sub-graph is received by the source that can use any algorithm capable of finding feasible non-dominated paths.

One of the key features of SANP is to be independent of a pre-determined sequence of ASes and can work with any underlying routing mechanism.

In order to assess SANP we have compared it with the exact solution. In order to run realistic simulations, we have investigated different solutions proposed in the literature to generate Internet-like graphs, as discussed in Chapter 4. Among five generators, we decided to chose Inet.

We have shown by simulations that SANP finds paths close to the optimal solution. Furthermore, α, a parameter for modifying the size of the sub-graph, enables us to tune the quality of the paths found by SANP. Yet even for reasonably small values of α (5,10,25) the results are fairly good.

In Chapter 6, we assumed that the ASes are ready to form alliances. The purposes of forming alliances are, first, to better satisfy the customers' requests and, second, to improve the revenue of the ASes belonging to the alliances. The second main contribution of this thesis is an algorithm named ACQA capable of computing end-to-end QoS paths in a graph comprised of alliances and ASes. Similarly to what we did for SANP, we assumed that both ASes and alliances publish a list of offers describing the QoS guarantees that they can offer between their entry and exit points (ASBRs). In the same way as SANP, ACQA constructs a sub-graph containing some of the nodes between the source and the destination, then it can compute a set of feasible non-dominated paths. The source can then select one of these paths based on its preferences.

In order to assess the benefit of forming alliances, we have compared the paths found with ACQA to the paths found with SANP in different simulations scenarios. First, we observed that ACQA satisfies more requests than SANP, thus, it is reasonable to affirm that the business of provisioning end-to-end services is more profitable if the ASes group together to form alliances. Moreover, we have shown that ACQA outperforms SANP regarding to the path's quality, i.e., the delay of the paths are lower and they are cheaper. Therefore, in a market based on supply and demand, the paths found with ACQA are more likely to be sold than the paths found with SANP. These results encourage the ASes to form alliances.

7.2 Future Works

The problem of provisioning end-to-end QoS paths is at the same time a business and a technical issue. In this dissertation, we have shown that it is technically possible to guarantee end-to-end QoS. However, we have not yet developed the protocols needed to use SANP and ACQA in the Internet. Recall that, both the algorithms assume that each node (AS and alliance) knows the topology of its neighborhood. These neighborhoods are constructed by exchanging topological and traffic engineering information within a radius. Therefore, a protocol enabling the exchange of information within a given radius must be implemented. BGP link-state is recent work described in an RFC draft [131], enabling BGP to share link state and TE information with external components. By adding a Time to Live parameter that decreased each time a node is traversed, BGP link-state is a feasible candidate protocol for gathering neighborhood typologies. In this draft, it is mentioned that the each operator has to specify the maximum rate at which the TE information will be advertised/withdrawn from neighbors. If the rate is low, it is likely that the resources of the path chosen by the source in the received sub-graph is not available anymore. An additional mechanism that verifies whether the resources are available by questioning the involved nodes is a possible research direction.

A good candidate in which both SANP and ACQA can be integrated is the PCE architecture. Moreover, the assumption of given sequence of ASes which constitutes the main limitation of the PCE architecture is solved since neither SANP nor ACQA rely on pre-determined AS-path.

In a classical network, when a packet arrives into a router, the last must look up in its routing table in order to know where the packet must be forwarded. In order to change the routing policies, manual intervention of an administrator must be operated. With the Software Defined Network (SDN) architecture these changes are automatized and even pre-defined. For that, the network's intelligence is put within a controller. The administrator defines policies in this controller which are delivered to the rest of the network equipment. Thus the network is dynamically managed and is centralized. Several works such as propose to use this architecture to build end-to-end services. It would be interesting to assess the performances of both SANP and ACQA in this architecture. 1 an inter-as routing component for software-defined network 2 outsourcing the routing control logic better internet routing based on sdn principles

Concerning the management of the AS-offers and alliance-offers several improvements can be performed. First, in order to perform a fine-grained management of available resources, the ASes and the alliances can implement more sophisticated systems. For instance, as it is commonly done by airlines, overbooking can be used without degrading the quality of the services by correctly estimating the traffic demand and its characteristics. Second, the choice to buy one offer rather than another should not be made solely on their QoS guarantees. Knowing that the AS will fulfill its contract is an important information. A reputation mechanism that rewards the entities that fulfill their contracts and sanctions the ones that are not reliable is a possible solution. Finally, in the ACQA algorithm, we have assumed that the ASes within an alliance are not ready to exchange critical information and thus proposed two straightforward alliance-offer construction methods. In a case of high trust between the members of the alliances, further studies could investigate improved strategies to build the alliance-offers.

Bibliography

[1] M. Boucadair, P. Morand, P. Levis, T. Coadic, H. Asgari, P. Georgatsos, D. Griffin, D. Spencer, and M. P. Howarth. QoS-enhanced border gateway protocol. 2005.

[2] Wen Xu and Jennifer Rexford. *MIRO: multi-path interdomain routing*, volume 36. ACM, 2006.

[3] X. Masip-Bruin, M. Yannuzzi, J. Domingo-Pascual, A. Fonte, M. Curado, E. Monteiro, F. Kuipers, P. Van Mieghem, S. Avallone, G. Ventre, P. Aranda-Gutiérrez, M. Hollick, R. Steinmetz, L. Iannone, and K. Salamatian. Research challenges in QoS routing. *Computer Communications*, 29(5):563–581, March 2006. ISSN 01403664. doi: 10.1016/j.comcom.2005.06.008.

[4] Youngseok Lee and Biswanath Mukherjee. Traffic engineering in next-generation optical networks. *Communications Surveys & Tutorials, IEEE*, 6(3):16–33, 2004.

[5] J. Vasseur, A. Ayyangar, and R. Zhang. A per-domain path computation method for establishing inter-domain traffic engineering (TE) label switched paths (LSPs). *draft-ietf-ccamp-inter-domain-pd-path-comp-05 (work in progress)*, 2007.

[6] G. Bertrand, S. Lahoud, G. Texier, and M. Molnár. A distributed exact solution to compute inter-domain multi-constrained paths. *The Internet of the Future*, pages 21–30, 2009.

[7] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *Selected Areas in Communications, IEEE Journal on*, 1996.

[8] Romain Jacquet, Géraldine Texier, and Alberto Blanc. SANP: An algorithm for selecting end-to-end paths with QoS guarantees. In *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, pages 1–10. IEEE, 2013.

[9] Jon Crowcroft, Steven Hand, Richard Mortier, Timothy Roscoe, and Andrew Warfield. QoS's downfall: at the bottom, or not at all! In *Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?*, pages 109–114. ACM, 2003.

[10] Eleni Mykoniati, Charalampos Charalampous, Panos Georgatsos, Takis Damilatis, Danny Goderis, Panos Trimintzios, George Pavlou, and David Griffin. Admission control for providing QoS in DiffServ IP networks: the TEQUILA approach. *Communications Magazine, IEEE*, 41(1):38–44, 2003.

[11] Michael P. Howarth, Paris Flegkas, George Pavlou, Ning Wang, Panos Trimintzios, David Griffin, Jonas Griem, Mohamed Boucadair, Pierrick Morand, Abolghasem Asgari, and others. Provisioning for interdomain quality of service: the MESCAL approach. *Communications Magazine, IEEE*, 43(6):129–137, 2005.

[12] Nicolas Le Sauze, Agostino Chiosi, Richard Douville, Hélia Pouyllau, Häkon Lonsethagen, Paola Fantini, Claudio Palasciano, Antonio Cimmino, MA Callejo Rodriguez, and Olivier Dugeon. ETICS: QoS-enabled interconnection for Future Internet services. *Future network and mobile summit*, 2010.

[13] A. Farrel, J. P. Vasseur, and J. Ash. A path computation element (PCE)-based architecture. Technical report, RFC 4655, August, 2006.

[14] J. P. Vasseur and J. L. Roux. Path computation element (PCE) communication protocol (PCEP). 2009.

[15] J. P. Vasseur, R. Zhang, N. Bitar, and J. L. Le Roux. A backward-recursive PCE-based computation (BRPC) procedure to compute shortest constrained inter-domain traffic engineering label switched paths. Technical report, RFC 5441, April, 2009.

[16] Manuel Domínguez Dorado, José Luis González Sánchez, Jordi Domingo Pascual, and others. PILEP: a contribution to PCE-based interdomain path computation. 2008.

[17] G. Bertrand. Mecanismes de routage inter-domaine multi-critere. Vers des services inter-operateurs a performances garanties. 2009.

[18] Jan Medved, Edward Crabbe, Ina Minei, and Robert Varga. PCEP Extensions for Stateful PCE. 2014.

[19] Daniel King and Adrian Farrel. The Application of the Path Computation Element Architecture to the Determination of a Sequence of Domains in MPLS and GMPLS. 2012.

[20] N. B. Djarallah, H. Pouyllau, S. Lahoud, and B. Cousin. Multi-Constrained Path Computation for Inter-Domain QoS-capable Services. 2011.

[21] N. B. Djarallah, N. L. Sauze, H. Pouyllau, S. Lahoud, and B. Cousin. Distributed E2e QoS-Based Path Computation Algorithm over Multiple Inter-domain Routes. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*, pages 169–176, 2011.

[22] Romain Jacquet, Geraldine Texier, and Alberto Blanc. Computing end-to-end QoS paths in the Internet considering multiple alliances. In *Telecommunications Network Strategy and Planning Symposium (Networks), 2014 16th International*, pages 1–6. IEEE, 2014.

[23] Aemen Lodhi, Amogh Dhamdhere, and Constantine Dovrolis. Analysis of peering strategy adoption by transit providers in the Internet. In *INFOCOM Workshops*, page 177, 2012.

[24] Aemen Lodhi, Amogh Dhamdhere, and Constantine Dovrolis. Open Peering by Internet Transit Providers: Peer Preference or Peer Pressure? In *IEEE INFOCOM*, 2014.

[25] Jonathan Liebenau, Silvia Elaluf-Calderwood, and Patrik Kärrberg. European internet traffic: problems and prospects of growth and competition. 2013.

[26] Nikolaos Chatzis, Georgios Smaragdakis, Anja Feldmann, and Walter Willinger. On the importance of Internet eXchange Points for today's Internet ecosystem. *ACM SIGCOMM Computer Communication Review*, 43(5):19–28, 2013.

[27] Lixin Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking (ToN)*, 9(6):733–745, 2001.

[28] Ricardo Oliviera, Dan Pei, Walter Willinger, Beichuan. Zhang, and Lixia Zhang. Quantifying the Completeness of the Observed Internet AS-level Structure. Technical Report TR-080026-2008, 2008.

[29] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 618–627. IEEE, 2002.

[30] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, and George Riley. AS relationships: Inference and validation. *ACM SIGCOMM Computer Communication Review*, 37(1):29–40, 2007.

[31] Amogh Dhamdhere and Constantine Dovrolis. The Internet is flat: modeling the transition from a transit hierarchy to a peering mesh. In *Proceedings of the 6th International COnference*, page 21, 2010.

[32] Yuval Shavitt and Udi Weinsberg. Topological trends of internet content providers. In *Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners*, pages 13–18. ACM, 2012.

[33] Miao Li, Hui Wang, and Jiahai Yang. Flattening and preferential attachment in the internet evolution. In *Network Operations and Management Symposium (APNOMS), 2012 14th Asia-Pacific*, pages 1–8. IEEE, 2012.

[34] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *Passive and Active Network Measurement*, pages 1–10. Springer, 2008.

[35] Jinjing Zhao, Peidong Zhu, Xicheng Lu, and Lei Xuan. Does the Average Path Length Grow in the Internet? In *Information Networking. Towards Ubiquitous Networking and Services*, pages 183–190. Springer, 2008.

[36] B. Edwards, S. Hofmeyr, G. Stelle, and S. Forrest. Internet Topology over Time. *arXiv preprint arXiv:1202.3993*, 2012.

[37] Hyung-Woo Choi and Young-Tak Kim. Configuration Managements for BGP/MPLS VPN and DiffServ-aware-MPLS VPN.

[38] John Moy. OSPF version 2. 1997.

[39] Gary Malkin. RIP version 2-carrying additional information. 1994.

[40] Fernando Matos, Alexandre Matos, Paulo Simões, and Edmundo Monteiro. Provisioning of Inter-Domain QoS-Aware Services. *Journal of Computer Science and Technology*, 30(2):404–420, March 2015. ISSN 1000-9000, 1860-4749. doi: 10.1007/s11390-015-1532-3.

[41] Faisal Aslam, Zartash Afzal Uzmi, and Adrian Farrel. Interdomain path computation: challenges and solutions for label switched networks. *IEEE Communications Magazine*, 45(10):94, 2007.

[42] Li Xiao, King-Shan Lui, Jun Wang, and Klara Nahrstedt. QoS extension to BGP. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 100–109. IEEE, 2002.

[43] Hélia Pouyllau and Richard Douville. End-to-end QoS negotiation in network federations. In *NOMS Wksps, 2010 IEEE/IFIP*, 2010.

[44] H. Ould-Brahim, D. Fedyk, and Y. Rekhter. BGP Traffic Engineering Attribute. *Internet Engineering Task Force, RFC*, 5543, 2009.

[45] H. Fujinoki. Multi-path BGP (MBGP): A solution for improving network bandwidth utilization and defense against link failures in inter-domain routing. In *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, pages 1–6, 2008.

[46] M. Yannuzzi, X. Masip-Bruin, and O. Bonaventure. Open issues in interdomain routing: a survey. *Network, IEEE*, 19(6):49–56, 2005.

[47] Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin. Resource reservation protocol (RSVP)–Version 1 functional specification. *Resource*, 1997.

[48] Tzi-cker Chiueh, Anindya Neogi, and Paul Stirpe. Performance analysis of an RSVP-capable router. In *Real-Time Technology and Applications Symposium, 1998. Proceedings. Fourth IEEE*, pages 39–48. IEEE, 1998.

[49] H. Jonathan Chao. *Quality of service control in high-speed networks*. Wiley, New York, 2002. ISBN 0471003972.

[50] Jozef Babiarz, K. Chan, and Fred Baker. Configuration guidelines for DiffServ service classes. *IETF-Request for Comments (RFC 4594)*, (4594), 2006.

[51] Van Jacobson, Kathleen Nichols, Kedarnath Poduri, and others. An expedited forwarding PHB. 1999.

[52] Juha Heinanen, Fred Baker, Walter Weiss, and John Wroclawski. Assured forwarding PHB group. Technical report, RFC 2597, june, 1999.

[53] K. Nichols and V. Jacobson. A Two-bit Differentiated Services Architecture for the Internet. Technical report, RFC 2638 (informational), July 1999.

[54] Zhenhai Duan, Zhi-Li Zhang, Yiwei Thomas Hou, and Lixin Gao. A core stateless bandwidth broker architecture for scalable support of guaranteed services. *Parallel and Distributed Systems, IEEE Transactions on*, 15(2):167–182, 2004.

[55] M. Mahajan and M. Parashar. Managing QoS for Multimedia Applications in the Differentiated Services Environment. *Journal of Network and Systems Management 11*, pages 469–498, 2003.

[56] Z.-l. Zhang, Z. Duan, L. Gao, and Y. T. Hou. Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services. *ACM SIGCOMM Computer Communication*, (30):71–83, 2000.

[57] Chamil PW Kulatunga, Jesse Kielthy, Paul Malone, and M. O. Foghlu. Implementation of a simple Bandwidth Broker for DiffServ networks. In *Proceedings 2nd International Workshop on Inter-Domain Performance and Simulation*, 2004.

[58] Ibrahim Khalil and Torsten Braun. Implementation of a bandwidth broker for dynamic end-to-end capacity reservation over multiple diffserv domains. In *Management of Multimedia on the Internet*, pages 160–174. Springer, 2001.

[59] Daniel Awduche, Angela Chiu, Anwar Elwalid, Indra Widjaja, and XiPeng Xiao. Overview and principles of Internet traffic engineering. Technical report, RFC 3272, may, 2002.

[60] DGTLVSD Awduche, Lou Berger, D. Gan, Tony Li, Vijay Srinivasan, and George Swallow. RSVP-TE: extensions to RSVP for LSP tunnels. Technical report, RFC 3209, December, 2001.

[61] K. Kompella, Y. Rekhter, and others. Label switched paths (LSP) hierarchy with generalized multi-protocol label switching (GMPLS) traffic engineering (TE). Technical report, RFC 4206, October, 2005.

[62] Arthi Ayyangar. Label Switched Path Stitching with Generalized Multiprotocol Label Switching Traffic Engineering (GMPLS TE). 2008.

[63] Faisal Aslam, Zartash Afzal Uzmi, Adrian Farrel, and Michal Pióro. Inter-domain path computation using improved crankback signaling in label switched networks. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 2023–2029. IEEE, 2007.

[64] Helia Pouyllau, Armen Aghasaryan, Laurent Ciarletta, and Stefan Haar. X-domain QoS budget negotiation using dynamic programming. In *AICT-ICIW*. IEEE, 2006.

[65] Young-Jun Seo, Hwa-Young Jeong, and Young-Jae Song. A study on web services selection method based on the negotiation through quality broker: A maut-based approach. In *Embedded Software and Systems*. Springer, 2005.

[66] Cao Yuanming, Wang Wendong, Gong Xiangyang, and Que Xirong. Initiator-Domain-Based SLA Negotiation for Inter-domain QoS-Service Provisioning. pages 165–169. IEEE, March 2008. ISBN 978-0-7695-3094-9. doi: 10.1109/ICNS.2008.43.

[67] Douglas S. Reeves and Hussein F. Salama. A distributed algorithm for delay-constrained unicast routing. *Networking, IEEE/ACM Transactions on*, 8(2):239–250, 2000.

[68] David Blokh. approximate algorithm for combinatorial optimization with two parameters. 1996.

[69] Guoliang Xue and S. Kami Makki. Multiconstrained QoS routing: a norm approach. *Computers, IEEE Transactions on*, 56(6):859–863, 2007.

[70] P. VanMieghem and F.A. Kuipers. Concepts of Exact QoS Routing Algorithms. *IEEE/ACM Transactions on Networking*, 12(5):851–864, October 2004. ISSN 1063-6692. doi: 10.1109/TNET.2004.836112.

[71] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for QoS routing. *Communications Magazine, IEEE*, 40(12):50–55, 2002.

[72] Fernando Kuipers, Turgay Korkmaz, Marwan Krunz, and Piet Van Mieghem. Performance evaluation of constraint-based path selection algorithms. *Network, IEEE*, 18(5):16–23, 2004.

[73] Whay C. Lee, Michael G. Hluchyi, and Pierre A. Humblet. Routing subject to quality of service constraints in integrated communication networks. *Network, IEEE*, 9(4):46–55, 1995.

[74] Jeffrey M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14(1):95–116, 1984.

[75] Lachlan LH Andrew and AAN Ananda Kusuma. Generalised analysis of a QoS-aware routing algorithm. In *Global Telecommunications Conference, 1998. GLOBECOM 1998. The Bridge to Global Integration. IEEE*, volume 1, pages 1–6. IEEE, 1998.

[76] P. Van Mieghem, H. De Neve, and F. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37(3):407–423, 2001.

[77] Liu Gang and RAMAKRISHNAN, K. G. A* Prune: an algorithm for finding K shortest paths subject to multiple constraints. *Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2:743–749, 2001.

[78] Shigang Chen and Klara Nahrstedt. On Finding Multi-constrained Paths. volume 2, pages 874–879, Atlanta, Ga, USA, June 1998.

[79] Xin Yuan and Xingming Liu. Heuristic algorithms for multi-constrained quality of service routing. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 844–853. IEEE, 2001.

[80] Hans De Neve and Piet Van Mieghem. TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm. *Computer Communications*, (23):667–679, 2000.

[81] Ronghui Hou, King-Shan Lui, Ka-Cheong Leung, and Fred Baker. Approximation Algorithms for QoS Routing with Multiple Additive Constraints. *IEEE Transactions on Computers*, pages 603–607, 2006.

[82] Xue, G., Zhang, W., Tang, J., and Thulasiraman. K. Polynomial time approximation algorithms for multi-constrained QoS routing. *IEEE/ACM Transactions on Networking*, pages 656–669, 2008.

[83] Turgay Korkmaz and Marwan Krunz. A randomized algorithm for finding a path subject to multiple QoS constraints. In *Global Telecommunications Conference, 1999. GLOBECOM'99*, volume 3, pages 1694–1698. IEEE, 1999.

[84] Jean-Jacques Pansiot and Dominique Grad. On routes and multicast trees in the Internet. *ACM SIGCOMM Computer Communication Review*, 28(1):41–50, 1998.

[85] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet map discovery. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1371–1380, 2000.

[86] Neil Spring, Mira Dontcheva, Maya Rodrig, and David Wetherall. How to Resolve IP Aliases. Technical Report UW-CSE-TR 04-05-04, 2004.

[87] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 133–145. ACM, 2002.

[88] M.H. Gunes and K. Sarac. Resolving IP Aliases in Building Traceroute-Based Internet Maps. *IEEE/ACM Transactions on Networking*, 17(6):1738–1751, December 2009. ISSN 1063-6692, 1558-2566. doi: 10.1109/TNET.2009.2014227.

[89] Ken Keys. Internet-scale IP alias resolution techniques. *ACM SIGCOMM Computer Communication Review*, 40(1):50–55, 2010.

[90] Ken Keys, Young Hyun, Matthew Luckie, and Kim Claffy. Internet-scale IPv4 alias resolution with MIDAR. *IEEE/ACM Transactions on Networking (TON)*, 21(2):383–399, 2013.

[91] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, A. Vahdat, et al. The Internet AS-level topology: Three data sources and one definitive metric. *ACM SIGCOMM Computer Communication Review*, 36(1):17–26, 2006.

[92] Leslie Daigle. WHOIS protocol specification. 2004.

[93] Georgos Siganos and Michalis Faloutsos. Analyzing BGP policies: Methodology and tool. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1640–1651. IEEE, 2004.

[94] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Xenofontas Dimitropoulos, and Amin Vahdat. Lessons from three views of the Internet topology. *arXiv preprint cs/0508033*, 2005.

[95] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger. *Towards capturing representative AS-level Internet topologies*, volume 30. ACM, 2002.

[96] Hamed Haddadi, Damien Fay, Almerima Jamakovic, Olaf Maennel, Andrew W. Moore, Richard Mortier, Miguel Rio, and Steve Uhlig. Beyond node degree: evaluating AS topology models. *arXiv preprint arXiv:0807.2023*, 2008.

[97] B. M. Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622, 1988.

[98] Jared Winick and Sugih Jamin. Inet-3.0: Internet topology generator. Technical report, Technical Report CSE-TR-456-02, 2002.

[99] Ibrahim Matta and Liang Guo. QDMR: An efficient QoS dependent multicast routing algorithm. *Communications and Networks, Journal of*, 2(2):168–176, 2000.

[100] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi, and A. Sprintson. Algorithms for computing QoS paths with restoration. *IEEE/ACM Transactions on Networking*, 13(3):648–661, June 2005. ISSN 1063-6692. doi: 10.1109/TNET.2005.850217.

[101] Marcus Kaiser. A tutorial in connectome analysis: topological and spatial features of brain networks. *Neuroimage*, 57(3):892–907, 2011.

[102] Cigdem Demir and Bülent Yener. Automated cancer diagnosis based on histopathological images: a systematic survey. *Rensselaer Polytechnic Institute, Tech. Rep*, 2005.

[103] Y. Shavitt and T. Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. *IEEE/ACM Transactions on Networking*, 12(6):993–1006, December 2004. ISSN 1063-6692. doi: 10.1109/TNET.2004.838597.

[104] Hamed Haddadi, Damien Fay, Steve Uhlig, Andrew Moore, Richard Mortier, Almerima Jamakovic, and Miguel Rio. Tuning topology generators using spectral distributions. In *Performance Evaluation: Metrics, Models and Benchmarks*, pages 154–173. Springer, 2008.

[105] Albert-László Barabási. Emergence of scaling in random networks.pdf. 1999.

[106] Bu Tian and Towsley Don. On distiguishing between Internet power law topology generators. *In Proceedings of IEEE Infocom 2002*, June 2002.

[107] Alberto Medina, Ibrahim Matta, and John Byers. On the origin of power laws in Internet topologies. *ACM SIGCOMM computer communication review*, 30(2): 18–28, 2000.

[108] Georgos Siganos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. Power laws and the AS-level internet topology. *IEEE/ACM Transactions on Networking (TON)*, 11(4):514–524, 2003.

[109] Shi Zhou and Raúl J. Mondragón. The positive-feedback preference model of the as-level internet topology. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 1, pages 163–167. IEEE, 2005.

[110] M. Piraveenan, M. Prokopenko, and A. Y. Zomaya. Local assortativity and growth of Internet. *The European Physical Journal B*, 70(2):275–285, July 2009. ISSN 1434-6028, 1434-6036. doi: 10.1140/epjb/e2009-00219-y.

[111] S. Zhou and R. J. Mondragon. Towards Modelling the Internet Topology - The Interactive Growth Model. Berlin, 2003.

[112] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):2–es, June 2006. ISSN 03600300. doi: 10.1145/1132952.1132954.

[113] Joylan Nunes Maciel and Cristina Duarte Murta. NIT: A New Internet Topology Generator. In *The Internet of the Future*, pages 108–117. Springer, 2009.

[114] Shi Zhou and Raúl J. Mondragón. The rich-club phenomenon in the Internet topology. *Communications Letters, IEEE*, 8(3):180–182, 2004.

[115] Shi Zhou and Raúl J. Mondragón. Accurately modeling the Internet topology. *Physical Review E*, 70(6):066108, 2004.

[116] Hamed Haddadi, Miguel Rio, Gianluca Iannaccone, Andrew Moore, and Richard Mortier. Network topologies: inference, modeling, and generation. *Communications Surveys & Tutorials, IEEE*, 10(2):48–69, 2008.

[117] Dapeng Hao and Chuanxing Li. The Dichotomy in Degree Correlation of Biological Networks. *PLoS ONE*, 6(12):e28322, December 2011. ISSN 1932-6203. doi: 10.1371/journal.pone.0028322.

[118] Ricardo V. Oliveira, Beichuan Zhang, and Lixia Zhang. Observing the evolution of Internet AS topology. *ACM SIGCOMM Computer Communication Review*, 37 (4):313–324, 2007.

[119] Joanna Tomasik and Marc-Antoine Weisser. aSHIIP: Autonomous Generator of Random Internet-like Topologies with Inter-domain Hierarchy. pages 388–390. IEEE, August 2010. ISBN 978-1-4244-8181-1. doi: 10.1109/MASCOTS.2010.47.

[120] Aemen Lodhi, Amogh Dhamdhere, and Constantine Dovrolis. GENESIS: An agent-based model of interdomain network formation, traffic flow and economics. In *INFOCOM, 2012 Proceedings IEEE*, pages 1197–1205. IEEE, 2012.

[121] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions*, 1999.

[122] David A. Van Veldhuizen. Multiobjective Evolutionary Algorithms Classifications, Analyses, and New Inovations. 1999.

[123] Xiangjiang Hu, Peidong Zhu, Kaiyu Cai, and Zhenghu Gong. AS Alliance in Inter-Domain Routing. pages 151–156. IEEE, 2008. ISBN 978-0-7695-3096-3. doi: 10.1109/WAINA.2008.209.

[124] H. E. I. Yuichiro, Akihiro Nakao, Tomohiko Ogishi, Toru Hasegawa, and Shu Yamamoto. AS Alliance for Resilient Communication over the Internet. *IEICE transactions on communications*, 93(10):2706–2714, 2010.

[125] Nishant Kumar and Girish Saraph. End-to-end QoS in interdomain routing. In *ICNS*, 2006.

[126] Hélia Pouyllau and Giovanna Carofiglio. Inter-carrier SLA negotiation using Q-learning. *Telecommunication Systems*, June 2011. ISSN 1018-4864, 1572-9451. doi: 10.1007/s11235-011-9505-5.

[127] Jeroen Famaey, Steven Latré, Tim Wauters, and Filip De Turck. End-to-End Resource Management for Federated Delivery of Multimedia Services. *Journal of Network and Systems Management*, September 2013. ISSN 1064-7570, 1573-7705. doi: 10.1007/s10922-013-9288-y.

[128] Nick Feamster, Hari Balakrishnan, Jennifer Rexford, Aman Shaikh, and Jacobus Van Der Merwe. The case for separating routing from routers. In *ACM SIGCOMM*, 2004.

[129] Dominique Barth, Thierry Mautor, and Daniel Villa Monteiro. Impact of alliances on end-to-end QoS satisfaction in an interdomain network. In *ICC*, 2009.

[130] Stijn Van Dongen. A Cluster Algorithm for Graphs, 2000.

[131] Jan Medved, Saikat Ray, Hannes Gredler, Adrian Farrel, and Stefano Previdi. North-Bound Distribution of Link-State and TE Information using BGP. 2014.

List of Publications

International Conferences

- Romain Jacquet, Geraldine Texier, and Alberto Blanc. SANP: An algorithm for selecting end-to-end paths with QoS guarantees. In Future Network and Mobile Summit (FutureNetworkSummit), 2013, pages 1?10. IEEE, 2013.
- Romain Jacquet, Geraldine Texier, and Alberto Blanc. Computing end-to-end QoS paths in the Internet considering multiple alliances. In *Telecommunications Network Strategy and Planning Symposium (Networks), 2014 16th International*, pages 1?6. IEEE, 2014.

Résumé

L'essor de nouvelles applications multimédia sur Internet tels que les vidéos à la demande ou la télévision a profondément changé le paysage économique et stratégique des acteurs de l'Internet. Ainsi, la nécessité d'avoir un réseau qui offre une qualité de service adaptée est devenue incontournable. Un mécanisme destiné à offrir de la qualité de service au niveau de l'inter-domaine doit respecter les propriétés de communications de l'inter-domaine qui sont la confidentialité et l'autonomie des domaines et la scalabilité. Ces propriétés rendent le problème de la qualité de service au niveau inter-domaine particulièrement difficile.

La thèse se concentre essentiellement sur la partie technique d'un mécanisme offrant de la qualité de service au niveau inter-domaine notamment en proposant un algorithme nommé SANP calculant des chemins de bout-en-bout satisfaisant un ensemble de contraintes. Pour cela SANP fait l'hypothèse que chaque domaine annonce des offres de qualité de service entre ses points d'entrées et de sortie.

L'idée de SANP est de construire un sous graphe entre la source et la destination dans lequel la source pourra choisir un chemin. Une deuxième algorithme nommé ACQA est proposé qui calcule des chemins de bout-en-bout satisfaisant un ensemble de contraintes dans un graphe de domaines et d'alliances. Une alliance est un groupe de domaines qui se sont regroupés pour différents intérêts. Nos simulations ont pour but de montrer la qualité et la quantité de chemins trouvés par SANP et ACQA.

Mots-clés : Qualité de service, Routage multi-contraint, Alliances

Abstract

In the Internet, value-added services such as HD video, online gaming or cloud computing represent an ever-increasing share of the total traffic. Hence, it is essential to adapt the Internet to offer end-to-end (e2e) quality of service (QoS) for these services. However, a mechanism designed to offer quality of service at the inter-domain level must respect the three inter-domain communications properties: confidentiality and autonomy of the domains and the scalability. These properties made the problem of provisioning e2e QoS particularly difficult.

In this thesis, we focus on the technical part of an e2e QoS mechanism by proposing an algorithm named SANP that computes e2e paths satisfying a set of QoS constraints. The idea of SANP is to construct a sub-graph between the destination and the source in which the latest can choose a path that satisfies its request. In a second part of the thesis we assume that the domains are ready to group together to form alliances. An alliance is defined as a group of domains that are ready to share business and/or technical policies. Thus, we propose a second algorithm named ACQA that computes e2e paths in a graph composed of alliances and independent domains.
By simulations, we want to assess the quality and the quantity of the paths found by both SANP and ACQA.

Keywords : Quality of service, Multi-constraint routing, Alliance

TELECOM Bretagne

n° d'ordre : 2015telb0357
Télécom Bretagne
Technopôle Brest-Iroise - CS 83818 - 29238 Brest Cedex 3
Tél : + 33(0) 29 00 11 11 - Fax : + 33(0) 29 00 10 00