École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Thèse

Présentée pour obtenir le grade de docteur
de Télécom ParisTech
Spécialité : Informatique et Réseaux

Federico LARROCA

Techniques d'Ingénierie de Trafic Dynamique pour l'Internet

Soutenue le 18 décembre 2009 devant le jury composé de

Rapporteurs	Fernando PAGANINI	Universidad ORT
	Peter REICHL	Telecommunications Research Center Vienna
Examinateurs	Annie GRAVEY	Télécom Bretagne
	Daniel KOFMAN	Télécom ParisTech
	Ariel ORDA	Israel Institute of Technology
	James ROBERTS	INRIA
Directeur de thèse	Jean-Louis ROUGIER	Télécom ParisTech
Invitée	Sara OUESLATI	Orange Labs

Acknowledgements - Remerciements - Agradecimientos

Está clarísimo que sin el apoyo (implícito o explícito, creo que a los efectos es más o menos lo mismo) de muchas personas, nunca hubiera podido terminar la tesis. Aquí voy a nombrar a ciertas de esas personas, en un orden entre aleatorio y cronológico que no muestra necesariamente la importancia relativa de cada una. Es seguro que me estoy olvidando de alguien, que espero sepa entender y se vea representado en el último item.

Primero que todo, y más que obviamente, tengo que agradecerle a la Pao. Es obvio que sin ella jamás hubiera podido soportar el transplante de Montevideo a Paris, y menos que menos hubiera podido terminar la tesis en tiempo y forma. Su apoyo fue a todo nivel (desde lo moral a lo científico) y, no puedo recalcarlo lo suficiente, vital para mí.

Al Belza también tengo que agradecerle, que fue quien me recomendó a Daniel para venir a hacer el doctorado.

Hablando de Daniel, es a él quien tengo que agradecerle haberme traído y confiado en mí. Creo que, por ahora, no lo he defraudado.

C'est clair que sans Jean-Louis je n'aurais jamais fini la thèse non plus. Encore une fois, son support tant humain que scientifique a été vital pour le bon déroulement de cette thèse.

Je tiens aussi à remercier Jim et Sara. Sans eux, la prèmiere partie de cette thèse n'aurait jamais été ecrite.

Il faut que je remercie également mes amis à Dareau (Masood, Paolo, Salma, Silvio, Stefano et al.). La bonne ambiance reignante au 5ème m'a beaucoup aidé, et les déjeuners à 13 heures vont me manquer.

También tengo que agradecer a todos los "uruguayos" que estando en París hicieron que la estadía se pasara casi volando. En particular a Chloé (que además corrigió el abstract en francés), Chupete, Edú, Jairo, Pedro, Rafa y Rose.

Vaya un muchas gracias también a la familia en Uruguay et ailleurs, e incluyo también a toda la patota de amigotes. A la distancia (y a veces ni tanto) el apoyo también fue importante. También incluyo en este agradecimiento a todas las visitas que pasaron por nuestro apartamento, que ayudaron a cortar la monotonía y a repasar lo turístico de la ciudad Luz.

I would also like to thank Fernando, Peter, Annie, Daniel, Ariel, Jim and Sara that kindly accepted to be part of the Jury.

Y bueno, por último agradezco a todos aquellos de los que me estoy olvidando y creen que deberían estar en esta página.

Résumé

Contexte et Motivation

L'Ingénierie Trafic (*Traffic Engineering* ou TE en anglais) est définie comme le domaine de l'ingénierie des réseaux traitant du problème de l'évaluation et de l'optimisation des performances des réseaux [1]. En particulier, nous nous intéressons à ce dernier aspect, i.e. l'amélioration de la performance d'un réseau opérationnel. Ce problème a été largement étudié, surtout pendant la première moitié des années 1990 quand les ressources étaient considérées comme rares et chères. Cependant, le coût des capacités des liens au coeur du réseau a beaucoup baissé pendant les années qui ont suivi, ce qui a rendu économiquement viable le surprovisionnement des capacités du réseau.

Néanmoins, au fur et à mesure que les services offerts sur ces réseaux et les applications utilisées ont évolué, le trafic transporté est devenu de plus en plus complexe et dynamique. La convergence des services de données, de téléphonie et de télévision (formant ce qu'on appelle le *triple-play*) vers un seul et unique réseau a eu comme effet une augmentation importante de la variabilité et de la complexité du trafic injecté dans le réseau. De plus, des études récentes sur le trafic interne, publiées par un important vendeur d'équipement réseau (Cisco Systems), prévoient l'avènement de l'ère Exabyte [2, 3]. Cette nouvelle ère de l'internet aura comme principale caractéristique une énorme croissance du trafic, poussée par la généralisation de la vidéo haute définition visionnée en ligne par les utilisateurs. Pour aggraver la situation, la possibilité d'événements inattendus comme la panne d'équipements, des attaques réseau de gros volumes, des *flash crowds*[1] ou encore des modifications du routage externe, constituent des sources supplémentaires d'incertitude vis-à-vis du volume et de la nature du trafic à transporter. Enfin, le déploiement de la fibre optique à l'accès (i.e. Fiber To The Home) accroît cette incertitude.

Les opérateurs doivent également faire face à d'autres problèmes. La capacité d'accès offerte aux utilisateurs que l'on vient de mentionner est telle que l'hypothèse

[1] Un événement précis qui attire une soudaine attention du public, générant un volume anormal de trafic en un point du réseau.

selon laquelle la capacité au coeur du réseau est infinie peut devenir obsolète dans un futur proche. En outre, il est important de noter que pour d'autres architectures d'accès (e.g. tous les réseaux sans fil), la capacité est intrinsèquement limitée de sorte qu'une simple augmentation de la capacité peut s'avérer impossible ou non-viable d'un point de vue économique.

Dans ce contexte, l'hypothèse de grande capacité ne suffit plus. Les opérateurs réseau sont maintenant, et peut-être plus que jamais, en besoin de mécanismes d'Ingénierie Trafic qui soient **efficaces** (conduisant à une bonne utilisation des ressources disponibles), **robustes** par rapport aux variations de trafic injecté dans le réseau (changements dans les volumes ou les caractéristiques des flux transportés) et plus **tolérants** (en cas de panne d'un noeud ou un lien). Au fur et à mesure que la taille du réseau augmente, la gestion du réseau peut devenir une tâche très complexe. Les administrateurs réseau sont donc très intéressés par l'**automatisation** (un minimum de configuration nécessaire) des mécanismes TE qu'ils mettent en pratique dans leur réseau.

Le *Partage de Charge Dynamique* (*Dynamic Load-Balancing* ou DLB en anglais) est un mécanisme TE qui satisfait tous les critères que l'on vient d'énumérer. Si une paire Origine-Destination (OD) est connectée par plusieurs chemins, le problème est simplement comment distribuer le trafic associé parmi ces chemins, en fonction d'un objectif prédéterminé (généralement par l'optimisation d'une fonction objectif). Dans ce mécanisme dynamique, les chemins sont fixés a priori et la quantité de trafic acheminée sur chaque route (distribution de trafic) est déterminée dynamiquement en fonction de la demande de trafic et de la situation actuelle du réseau. Idéalement, la distribution de trafic est telle qu'à chaque instant la fonction objectif est optimisée. L'**efficacité** du DLB vient de cette dernière caractéristique. Dans ce sens, il existe plusieurs fonctions objectif possibles. Par exemple, les auteurs de [4] proposent un mécanisme appelé MATE utilisant une fonction croissante, continue et convexe $f_l(\rho_l)$ qui mesure la congestion dans le lien l (où ρ_l est la charge dans le lien). Leur objectif est de minimiser la congestion totale dans le réseau, qui naturellement est $\sum_l f_l(\rho_l)$. Un autre objectif possible est celui utilisé dans [5, 6] (appelés TeXCP et REPLEX respectivement), qui est simplement de minimiser l'utilisation du lien ($u_l = \rho_l/c_l$ où c_l est la capacité du lien) maximale dans le réseau.

Dans la pratique, cette caractéristique d'être "toujours optimisé" est remplie par l'utilisation d'un algorithme distribué que chaque router d'entrée exécute périodiquement, et qui se base sur des informations de *feedback* provenant du réseau. Tant que la distribution de trafic est actualisée à une fréquence suffisamment grande, DLB sera **robuste** et **tolérant**. Finalement, et comme conséquence directe de sa nature distribuée, ce type de mécanisme est aussi **automatisé**. Par contre, c'est également cette nature distribuée qui constitue le défi le plus important de DLB. Le déploiement de DLB a d'ailleurs été limité, les opérateurs réseau étant peu disposés à utiliser des mécanismes dynamiques dans leurs réseaux par crainte de possibles oscillations. De fait, les premières expériences menées dans le cadre d'ArpaNet avec ce type de mécanismes ont été très négatives de ce point de vue [7]. Il faut néanmoins souligner que l'algorithme utilisé n'avait pas fait l'objet d'études théoriques approfondies.

Il est certain que tous les algorithmes distribués présentent un compromis entre adaptabilité (vitesse de convergence) et stabilité qui peut s'avérer difficile à trouver, surtout dans des situations où de grands changements du volume de trafic peuvent se produire. Dans ce sens, l'alternative la plus importante à DLB est sans conteste le *Routage Robuste* (RR) [8, 9]. Dans le RR, l'incertitude du trafic est prise en compte directement dans l'optimisation du routage. On détermine une configuration *unique* de routage pour toutes les demandes de trafic appartenant à un certain ensemble d'incertitude dans lequel on suppose que le trafic varie. Cet ensemble d'incertitude peut se définir de plusieurs manières, en fonction de l'information disponible : les charges les plus grandes observées pendant une certaine période de temps, un ensemble de demandes de trafic observées avec antériorité, etc. L'objectif du RR est généralement de minimiser la plus grande utilisation du lien dans le réseau, pour toutes les demandes dans l'ensemble d'incertitude. Il est à noter que l'optimisation avec incertitude étant beaucoup plus difficile que l'optimisation classique, des objectifs plutôt simples sont toujours choisis. Bien que la configuration du routage résultant n'est pas optimale pour aucune demande en particulier, la pire des performances est maximisée globalement pour toutes les demandes. Ceux qui préconisent l'utilisation de RR soutiennent qu'il n'y a pas besoin de mettre en oeuvre des algorithmes d'optimisation distribués prétendument compliqués. L'argument généralement mis en évidence est également que la perte en performance subie à cause de l'utilisation d'une unique configuration de routage est négligeable par rapport à la complexité induite par la mise en place d'algorithmes distribués.

Dans cette thèse sont étudiés et proposés plusieurs mécanismes de DLB. Tout d'abord, il est fait une distinction entre les architectures pour lesquelles la capacité et les ressources sont réservées pour chaque chemin, et celles pour lesquelles aucune réservation n'est effectuée (c'est-à-dire à ressources partagées). Cette simplification va nous permettre de proposer l'utilisation d'un nouvel mécanisme pour gérer ces chemins. Partant de ce mécanisme, nous allons définir un nouvel algorithme de DLB. Dans le cas de ressources partagées, nous allons étudier et comparer plusieurs fonctions objectif, celles présentées ci-avant et une nouvelle fonction que nous avons proposée. Nous allons proposer et étudier un nouvel algorithme distribué pour atteindre l'optimum de ces fonctions objectifs. Sa caractéristique la plus importante, et son avantage par rapport à des propositions antérieures, est sa capacité d'auto-configuration (i.e. la convergence de l'algorithme est garantie sans aucun besoin de réglage préalable des paramètres). Dans les sections qui suivent on détaille les propositions les plus importantes de notre thèse.

Première Partie : Ressources Réservées

La thèse compte deux parties. La première suppose que chaque chemin utilise des ressources qui lui sont allouées exclusivement (comme par exemple avec MPLS-TE, GMPLS, GELS). Dans ce type d'architecture, les chemins sont traditionnellement gérés par un mécanisme appelé *token bucket* (seau à jetons). Il a été montré que cette technique de Qualité de Service (QoS) est très inefficace et peu appropriée pour caractériser le trafic [10]. En effet, pour obtenir une certaine probabilité de non-conformité, le paramètre de *burst* doit être exagérément grand, même lorsque le paramètre de *rate* est déjà plusieurs fois plus grand que le débit moyen du trafic. Il apparaît donc qu'on ne peut se fier aux paramètres du *token bucket* déclarés par le client, sous peine de gaspiller des ressources.

Nous proposons donc que les routers de bord utilisent le mécanisme *Cross-Protect* [11, 12] pour gérer l'accès à chacun de ses chemins. Le Cross-Protect permet de garantir une certaine performance pour le trafic de type élastique ou *streaming* en gardant l'interface utilisateur *best-effort* de l'internet. Autrement dit, il n'est pas nécessaire de marquer des paquets ou de signaler les flots pour pouvoir distinguer les flots du type *streaming* (qui ont besoin d'un bas délai et taux de pertes) et du type élastique (qui nécessitent le plus grand débit possible). A la place, le cross-protect identifie les flots à la volée, et met en oeuvre un ordonnancement (scheduling) au niveau flot en utilisant *Priority Fair Queueing* (PFQ). De plus, un mécanisme de contrôle d'admission basé sur des mesures est mis en place pour garantir un fair rate (i.e. le débit obtenu par les flots élastiques) minimum et des conditions optimales pour les flots du type *streaming.*

L'algorithme de PFQ classifie les flots en *streaming* ou élastiques implicitement et en ligne selon le principe qui suit : si un flot est transmis à un débit plus petit que le fair-rate instantané actuel (cette situation se manifeste par le fait de que les paquets du flot ne sont pas accumulés dans le router), celui-ci est classifié comme *streaming* et est envoyé en priorité. Les flots restants (ou tous les flots dont les paquets s'accumulent dans le router) partagent à égalité la capacité restante. De cette façon, un partage juste des ressources est atteint grâce à l'algorithme d'ordonnancement, et ne dépend pas d'algorithmes de contrôle de congestion mis en oeuvre par les utilisateurs finaux. Par contre, l'utilisation de TCP est nécessaire pour que les flots trouvent leur fair-rate.

Le contrôle d'admission est utilisé pour maintenir le nombre de flots gérés par le router au dessous d'une quantité raisonnable. De plus, il sert à garantir un QoS minimal pour tous les flots. Plus précisément, on veut limiter la quantité de trafic prioritaire (une quantité trop élevée entraînerait une mauvaise performance pour ce trafic) et garantir un débit minimal pour les flots élastiques. Donc, si à l'arrivée d'un nouveau flot le trafic prioritaire est au dessus d'un certain seuil ou le fair-rate au dessous d'un autre, on ne l'accepte pas et on bloque ses paquets. La Fig. 1 présente un diagramme simplifié du mécanisme de cross-protect. Il faut noter que l'algorithme de classification est appliqué en continu, et pas seulement à l'arrivée des flots.

En résumé, on dispose avec le cross protect d'un mécanisme qui garantit un débit minimum pour les flots élastiques et de très bas délai pour le trafic *streaming.* Il est important néanmoins d'évaluer la probabilité de blocage des flots causée par le contrôle d'admission, afin de mieux appréhender le compromis entre QoS (débit minimum pour les flux élastiques, etc.) et probabilité de rejet des flux entrants. Nous avons pour cela analysé un router équipé du mécanisme cross-protect et nous avons obtenu une formule pour la probabilité de blocage en fonction des seuils et des principales caractéristiques

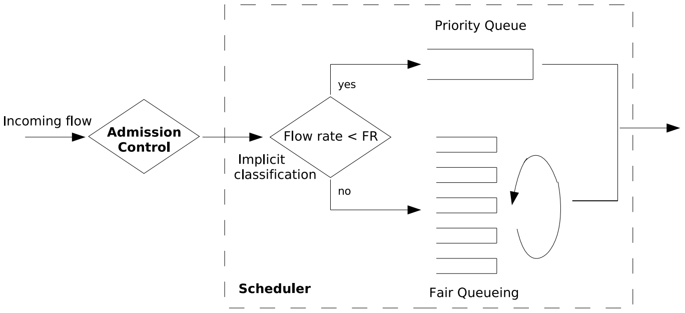


Figure 1: Diagramme simplifiée du Cross-Protect

du trafic. L'analyse est basée sur l'hypothèse que les flots *streaming* ont une échelle temporelle beaucoup plus grande que celle des flots élastiques. Cette supposition nous permet alors d'analyser la file d'attente des flots élastiques comme si la quantité de flots *streaming* était fixe (chacun de ces flots étant caractérisé par un débit fixe r).

Après avoir analysé un router cross-protect, nous avons étudié la possibilité de définir un mécanisme de partage de charge (DLB) tirant profit de ce mécanisme. Plus exactement, dans le cas où plusieurs chemins sont disponibles pour une paire OD, le problème est vers quel chemin aiguiller chaque flot entrant. Étant donné qu'avec le cross protect les mesures sur l'état des chemins sont effectuées pour le contrôle d'admission, il semble naturel de réutiliser ces résultats de mesure et donc de définir un mécanisme de DLB à partir du priority load (PL) et du fair-rate (FR). En se basant sur des analyses de la politique de routage optimale dans le cas d'un réseau de file d'attente du type *Processor Sharing*, nous avons proposé l'algorithme suivant, que nous avons appelé SGP (*Simple Greedy Policy*, politique gloutonne simple) :

$$\text{Acheminer au chemin } i \Leftrightarrow i = \arg\max_j fair\_rate_j$$

Intuitivement, l'algorithme minimise la probabilité de blocage pour les flots élastiques et, en même temps, maximise leur performance (en les envoyant par le chemin avec le plus grand débit). Par contre, la même chose ne peut pas être dite à propos des flots *streaming*. Néanmoins, étant donné qu'ils constituent la minorité du trafic, on sacrifie un peu sa performance pour favoriser un mécanisme qui reste toujours implicite vis-à-vis de la classification des flots.

L'analyse mathématique de SGP s'avére beaucoup plus compliquée que celle d'un router cross-protect qui n'utilise qu'un seul chemin. Néanmoins, nous sommes arrivés à calculer des bornes pour la probabilité de blocage des flots, qui finalement constituent une bonne approximation de cette probabilité. Nous avons aussi comparé SGP avec d'autres algorithmes de partage de charges statiques, et vérifié que la probabilité de blocage pour SGP peut être plus petite, de plusieurs ordres de grandeur, que pour les autres algorithmes.

Deuxième Partie : Ressources Partagées

Dans cette deuxième partie, on va être plus général dans le sens où on ne va pas supposer que les chemins ont des ressources réservées. Le premier aspect à prendre en compte est l'échelle temporelle à laquelle on va travailler. Dans ce cas, l'échelle sera celle du routage, c'est-à-dire quelques secondes ou quelques minutes. Etant donné ces temps, on ne va pas prendre en compte le contrôle de congestion, et supposer que le trafic dans le réseau est bien représenté par la Matrice de Trafic (TM, *Traffic Matrix*). Autrement dit, si chaque noeud dans le réseau est numéroté, cette matrice contient dans son entrée i–j la moyenne de trafic envoyé du noeud i au noeud j.

Dans la suite de ce résumé, on va tout d'abord présenter deux fonctions objectif possibles pour DLB dans ce contexte. Ensuite, on discutera comment obtenir leur optimum de façon distribuée. Finalement, on présentera un bref résumé des principaux résultats obtenus à partir de nos simulations.

Partage de Charge Basé sur la Maximisation de l'Utilité

Imaginons que chaque paire OD est numérotée par l'index $s = 1, \ldots, S$. Chacune de ces paires OD peut utiliser n_s chemins d'un ensemble P_s (où chacun de ses éléments va être noté comme P_{si} pour $i = 1, \ldots, n_s$). Imaginons pour l'instant que chaque paire OD est constituée d'un total de N_s flots TCP, dont $N_{P_{si}}$ utilisent le chemin P_{si} . Donc, le problème de contrôle de congestion peut être écrit de la façon suivante :

$$\begin{aligned} \underset{x}{\text{maximiser}} \quad & \sum_{s=1}^{S} \sum_{i=1}^{n_s} N_{P_{si}} U_{P_{si}} \left(\frac{x_{P_{si}}}{N_{P_{si}}} \right) \\ \text{s. à} \quad & \sum_{s} \sum_{i:l \in P_{si}} x_{P_{si}} \leq c_l \end{aligned} \tag{1}$$

Où $x_{P_{si}}$ est le débit total obtenu par tous les $N_{P_{si}}$ flots dans le chemin P_{si} (on a supposé qu'ils partagent à égalité ce débit) et $U_{P_{si}}$ est une fonction continue, croissante et concave.

Il faut noter que le problème maximise en fonction de la variable $x_{P_{si}}$ étant donné les $N_{P_{si}}$ (c'est-à-dire, le partage de charge est donné et TCP calcule le débit). Cependant, on pourrait essayer de maximiser en même temps en $x_{P_{si}}$ et $N_{P_{si}}$ pour améliorer encore l'utilité, et avec elle la performance obtenue par les flots. Une première idée serait que les utilisateurs finaux se chargent de choisir le chemin qu'ils vont emprunter. Malheureusement, cela est trop compliqué d'un point de vue technique et politique (une seule route est généralement disponible avec l'architecture de routage actuelle). Donc, nous proposons de laisser la séparation entre partage de charge (maximisation en $N_{P_{si}}$) et contrôle de congestion (maximisation en $x_{P_{si}}$), et d'essayer de toute façon de résoudre (1).

Plusieurs difficultés surgissent alors, surtout liées à l'échelle temporelle considérée. Notamment, il est impossible de considérer que le nombre de flots est fixé, et on ne peut pas non plus connaître la valeur instantanée de $x_{P_{si}}/N_{P_{si}}$. Il faut donc trouver une estimation de la moyenne temporelle de ces deux quantités. Le problème résultant d'une telle approximation est le suivant (où $d_{P_{si}}$ est le trafic envoyé par la paire OD s dans son chemin P_{si} d'un total d_s) :

$$\underset{d}{\text{maximiser}} \sum_{s=1}^{S}\sum_{i=1}^{n_s} d_{P_{si}} U\left(\min_{l\in P_{si}}\{c_l-\rho_l\}\right) \tag{2}$$
$$\text{s. à} \quad d_P \geq 0\ \forall P\in\mathcal{P}_s \text{ and } \sum_{P\in\mathcal{P}_s} d_P = d_s \quad \forall s=1,\dots,S$$

Dans ce cas, on a supposé que le débit par flot est approximativement égal à la bande passante disponible (ABW, *available bandwidth*) dans le chemin [13, 14], et on a substitué $N_{P_{si}}$ par $d_{P_{si}}$. Pour caractériser l'optimum du problème (2), nous avons utilisé les conditions de Karush-Kuhn-Tucker (KKT) [15], et nous avons obtenu la condition suivante pour chaque paire OD (où $\lambda^*_{P_{si}} > 0$ et ν^*_s sont des constantes) :

$$\phi^*_{P_{si}} = -U\left(\min_{l\in P_{si}}\{c_l-\rho^*_l\}\right) + \sum_{l:l\in P_{si}} \widehat{\theta}_l = \begin{cases} -\nu^*_s & \text{if } d^*_{P_{si}} > 0. \\ -\nu^*_s + \lambda^*_{P_{si}} & \text{if } d^*_{P_{si}} = 0 \end{cases} \tag{3}$$

$$\text{où} \quad \theta^*_{P_{si}l} = \begin{cases} d^*_{P_{si}} & \text{si } l = \underset{l\in si}{\operatorname{argmin}}\{c_l-\rho^*_l\} \\ 0 & \text{autrement} \end{cases}$$
$$\widehat{\theta}_l = \sum_{s=1}^{S}\sum_{i:l\in P_{si}} \theta^*_{P_{si}l} U'(c_l-\rho^*_l)$$

Autrement dit, pour chaque paire OD s, les chemins utilisés dans la distribution du trafic optimal ont le même coût $\phi^*_{P_{si}}$. De plus, ce coût est plus petit que pour les chemins qui ne sont pas utilisés. Comme on le verra plus tard, cette formulation va nous permettre de concevoir un algorithme distribué.

Partage de Charge de Congestion Minimale : Pourquoi choisir une fonction de congestion arbitraire ?

Dans cette section, on va s'intéresser à la fonction objectif "classique" utilisée, par exemple, dans MATE [4]. Comme présenté ci-avant, dans ce cas on définit une fonction croissante, convexe et continue $f_l(\rho_l)$ qui mesure la congestion dans le lien. L'objectif est de minimiser la congestion totale dans le réseau $\sum_l f_l(\rho_l)$. Typiquement $f_l(\rho_l)$ est choisie comme le délai de la file d'attente. Un tel choix est justifié par sa simplicité (le délai total du chemin est l'addition des délais dans chaque lien) et sa versatilité (de gros délais indiquent une mauvaise performance pour tous les types de trafic). Par contre, la grande majorité des mécanismes DLB ont besoin d'une expression analytique de ce délai. Pour trouver une telle expression, on utilise généralement des modèles simples [4] comme le *M/M/1* [16].

Nous proposons au contraire d'apprendre la fonction $f_l(\rho_l)$ à partir des mesures. Supposons que le délai de la file d'attente dans le lien l est donné par la fonction $D_l(\rho_l)$. Alors, le problème dans ce cas est le suivant :

$$\begin{aligned} \underset{d}{\text{minimiser}} & \sum_{l=1}^{L} D_l(\rho_l)\rho_l := \sum_{l=1}^{L} f_l(\rho_l) \\ \text{s. à} \quad & d_{P_{si}} \geq 0 \quad \sum_{P \in \mathcal{P}_s} d_P = d_s \end{aligned} \tag{4}$$

C'est-à-dire, minimiser le délai moyen où chaque lien est pondéré par la quantité de trafic qu'il transporte. Nous avons préféré minimiser cette moyenne pondérée plutôt qu'une simple addition parce qu'elle mesure plus correctement la performance perçue par le trafic. Il faut noter que $f_l(\rho_l) = \rho_l D_l(\rho_l)$ est proportionnel au nombre moyen d'octets dans la file d'attente du lien l. Nous allons donc utiliser cette fonction $f_l(\rho_l)$ qui est beaucoup plus facile à mesurer que le délai, et qui est directement disponible dans les routers du marché.

La caractérisation de l'optimum dans ce cas peut être obtenue d'après les conditions KKT, ce qui résulte dans la condition suivante :

$$\phi^*_{P_{si}} = \sum_{l \in P_{si}} \phi_l(\rho^*_l) = \sum_{l \in P_{si}} f'_l(\rho^*_l) = \begin{cases} -\nu^*_s & \text{si } d^*_{P_{si}} > 0. \\ -\nu^*_s + \lambda^*_{P_{si}} & \text{si } d^*_{P_{si}} = 0 \end{cases} \tag{5}$$

On obtient donc la même condition que dans la section précédente, sauf que le coût du chemin est plus simple. En définissant le coût du lien $\phi_l(\rho_l) = f'_l(\rho_l)$, le coût du chemin est donné par son addition (i.e. $\phi_P = \sum_{l \in P} \phi_l(\rho_l)$). Par contre, il faut prendre en compte le fait que la condition est nécessaire et suffisante uniquement si la fonction $f_l(\rho_l)$ est convexe et de dérivée continue.

Le problème que nous abordons maintenant est comment déduire $\phi_l(\rho_l) = f'_l(\rho_l)$ des observations de la taille moyenne de la file d'attente. En fait, comme la quantité observable est justement $f_l(\rho_l)$, nous allons d'abord apprendre $f_l(\rho)$ et estimer ensuite sa dérivée par dérivation de cette estimation. Comme la procédure pour chaque lien est exactement la même, on va omettre le sous-index l dans la suite. Supposons que nous avons un ensemble d'observations $\{(\rho_1, Y_1)\ (\rho_2, Y_2) \ldots (\rho_N, Y_N)\}$ (aussi appelé *ensemble d'apprentissage*), et supposons que la variable de réponse Y (la taille moyenne de la file d'attente mesurée) est reliée à la variable ρ par l'équation suivante :

$$Y_i = f(\rho_i) + \epsilon_i \qquad i = 1, \ldots, N \tag{6}$$

L'erreur de mesure ϵ_i est une variable aléatoire telle que $\mathbb{E}\{\epsilon_i\} = 0$ et $\text{Var}\{\epsilon_i\} = \sigma_i < \infty$. Le problème des *Moindres Carrés Pesés* (WLS, *Weighted Least Squares*) consiste à trouver la fonction $\widehat{f}(\rho)$ qui minimise la moyenne pondérée des erreurs élevées au carré, tout en supposant que $\widehat{f}(\rho)$ appartient à une famille de fonctions $\mathcal{F}$ donnée :

$$\min_{f} \sum_{i=1}^{N} w_i \left(Y_i - f(\rho_i)\right)^2 \quad \text{s. à } f \in \mathcal{F} \tag{7}$$

où w_i indique le poids relatif de l'observation i par rapport au reste des observations dans l'ensemble d'apprentissage.

Nous allons présenter deux méthodes pour résoudre ce problème, qui diffèrent par leur famille $\mathcal{F}$. La première méthode que nous avons étudiée s'appelle *Convex Nonparametric Weighted Least Squares* (CNWLS), une version pondérée du CNLS original [17]. Dans ce cas, $\mathcal{F}$ est le plus général possible, c'est-à-dire la famille des fonctions continues, croissantes et convexes. Dans ce cas, on peut démontrer que le minimum de la moyenne pondérée des erreurs est obtenu par une fonction linéaire par morceaux (i.e. $\widehat{f}(\rho) = \max_{i=1,\ldots,N} \alpha_i + \beta_i\rho$) qui peut être obtenu par le problème de programmation quadratique (QP, *quadratic programming*) suivant :

$$\begin{aligned} &\min_{\epsilon,\alpha,\beta} \sum_{i=1}^{N} w_i\epsilon_i 2 && (8)\\ \text{s. à} \quad & Y_i = \alpha_i + \beta_i\rho_i + \epsilon_i \; \forall i = 1,\ldots,N \\ & \alpha_i + \beta_i\rho_i \geq \alpha_j + \beta_j\rho_i \; \forall j, i = 1,\ldots,N \\ & \beta_i \geq 0 \; \forall i = 1,\ldots,N \end{aligned}$$

Même si cette méthode obtient la meilleure précision possible, le problème QP à résoudre pour trouver la solution peut être trop grand. En fait, le deuxième ensemble de restrictions, qui garanti la convexité de $\widehat{f}(\rho)$, est quadratique dans le nombre d'observations disponibles dans l'ensemble d'apprentissage. L'alternative est de fixer le nombre maximal de morceaux à k (i.e. $\widehat{f}(\rho) = \max_{i=1,\ldots,k} \alpha_i + \beta_i\rho$) et d'essayer de résoudre le problème résultant. Ce problème, appelé *Convex Piecewise Linear Fitting* [18], ne peut pas se résoudre exactement comme CNWLS. Il faut plutôt utiliser des heuristiques, comme celui proposé dans [18]. Bien que la vitesse de calcul soit fortement réduite, la précision de la fonction résultante est également moindre.

Obtention de l'Optimal : Jeu de Routage et Algorithmes Sans Regret

L'optimal pour (2) et (4) est caractérisé par une certaine fonction de coût du chemin ϕ_P. Plus exactement, pour chaque paire OD, les chemins qui sont utilisés sont ceux qui présentent le plus petit coût parmi tous les chemins de chaque paire OD. En fait, comme discuté par la suite, cette situation peut être interprétée comme l'équilibre résultant des paires OD qu'utilisent des algorithmes gloutons.

Dans ce type d'algorithme, chaque paire OD essaye toujours de minimiser le coût associé à ses chemins. Ce contexte constitue un cas d'étude idéal pour la Théorie des

Jeux, et on l'appelle *Jeu de Routage* [19]. Dans un jeu comme le nôtre, où le trafic peut être réparti entre les chemins de manière arbitraire, on suppose que les paires OD sont constituées pour une infinité d'agents. Chacun de ces agents contrôle une quantité infinitésimale du trafic, et décide du chemin qu'il veut prendre.

Si chaque agent est glouton, le système va être en équilibre quand aucun agent ne peut décroître son coût en changeant de chemin. Une telle situation constitue ce qu'on appelle un équilibre de Wardrop (WE, *Wardrop Equilibrium*) [20], qui est formellement défini comme suit :

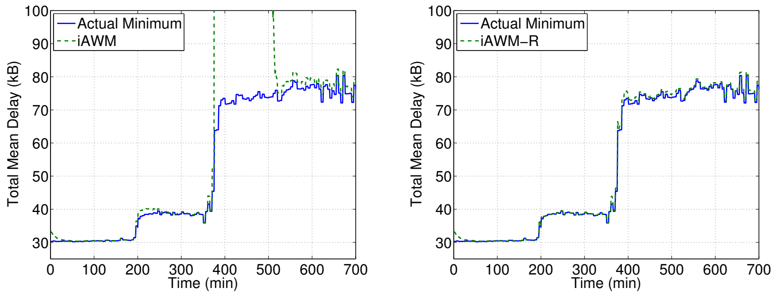
Définition 1. *Une distribution trafic d est un équilibre de Wardrop si pour chaque paire OD $s = 1 \dots S$ et pour chaque chemin P_{si} avec $d_{P_{si}} > 0$ il est vrai que $\phi_{P_{si}} \leq \phi_{P_{sj}}$ pour tout $P_{sj} \in \mathcal{P}_s$.*

Il est clair que l'optimum recherché et le WE (pour le coût de chemin correspondant) sont les mêmes. On va donc étudier les différents algorithmes possibles permettant de converger vers cet équilibre. En particulier, on va s'intéresser aux algorithmes du type *Sans Regret*. Les auteurs de [21] ont prouvé que si toutes les paires OD utilisent ce type d'algorithme, la convergence au WE est garantie. En particulier, nous avons choisi l'algorithme *Incrementally Adaptive Weighted Majority* (iAWM) proposé dans [22]. Cet algorithme a la particularité d'être complètement auto-configuré. Autrement dit, il a la particularité de ne nécessiter aucun calcul de paramètre pour garantir sa convergence, ce qui est un de nos objectifs. En fait, tous les algorithmes d'optimisation distribués proposés jusqu'à aujourd'hui ont un paramètre qui contrôle la vitesse de l'algorithme qui peut être très difficile à fixer.

Malheureusement, l'algorithme fonctionne uniquement dans des cas où le trafic est stationnaire, et a une très lente réaction en cas des changements brusques des demandes. Pour résoudre ce problème, nous avons proposé une variation de l'algorithme, que nous avons appelé *Incrementally Adaptive Weighted Majority with Restart* (iAWM-R). Le principe est très simple : l'algorithme est le même que iAWM, sauf que quand on détecte un changement brusque, on le réinitialise. Dans la Fig. 2 on peut voir une comparaison entre les deux algorithmes, utilisant le réseau Abilene. On peut remarquer que la réinitialisation de iAWM-R accélère énormément la convergence par rapport à iAWM.

Évaluation et Conclusions

Nous avons finalement effectué une comparaison de trois fonctions objectif vis-à-vis de leurs performances. Nous avons tout d'abord considéré la maximisation de l'utilité (cf. (2)) qu'on va appeler ici MaxU (*Maximum Utility*). Nous avons également considéré le partage de charge de congestion minimale basé sur la régression (cf. (4)) qu'on va noter MinQ (*Minimum Queue*). Finalement, nous avons considéré le WE obtenu en utilisant le coût de chemin $\phi_P = \max_{l \in P} u_l$ qui modélise les algorithmes classiques comme TeXCP [5] ou REPLEX [6] qu'on va appeler MinMaxU (*Minimum Maximum Utilization*). D'après cette étude, conduite sur deux véritables topologies (Abilene et Géant)



(a) iAWM

(b) iAWM-R

Figure 2: Délai moyen total en fonction du temps pour le deux algorithmes de partage de charge.

et en utilisant des matrices de trafic réelles, on peut obtenir plusieurs conclusions. En premier lieu, en ce qui concerne la bande passante obtenue sur les chemins (ABW), elle est toujours plus grande pour MaxU que pour les autres algorithmes. Plus précisément, la différence avec MinQ n'est pas très grande, mais la différence avec MinMaxU par contre est significative. Deuxièmement, et en ce qui concerne l'utilisation des liens (u_l), les résultats sont très similaires entre MaxU et MinMaxU. Les résultats pour MinQ sont très similaires aux deux autres en moyenne et pour le dernier décile. Par contre, l'utilisation maximale obtenue peut être très différente de celle de MinMaxU. Cet écart est dû à la réticence de MinQ à utiliser les chemins les plus longs. En termes de délai d'attente moyen total ($D(d)$, un indicateur de performance très important pour tous les types de trafic), la différence peut être très importante entre ces différentes fonctions objectif. On peut constater une augmentation de plus de 50% entre MinMaxU ou MaxU et MinQ. De plus, même si la différence pour MaxU est généralement plus petite que celle de MinMaxU, c'est cet algorithme qui obtient la pire des différences (plus de 100%).

Nous avons étudié le gain en $D(d)$ du partage de charge basé sur la régression par rapport à l'utilisation de fonctions $f_l(\rho_l)$ simplistes, en particulier basées sur le modèle *M/M/1*. Nous avons prouvé qu'en termes d'utilisation des liens et de la bande passante offerte aux différents chemins, ce choix n'est pas très important, tant que la fonction est convexe, croissante et “explose” à mesure que $f_l(\rho_l)$ s'approche de la capacité du lien. Par contre, en termes du délai d'attente total moyen ce choix s'avère crucial. Par exemple, des augmentations de plus de 50% par rapport au $D(d)$ optimal ne sont pas rares. Finalement, nous avons aussi vérifié que l'augmentation subite de $D(d)$ à cause du manque de précision de CPLF n'est pas si importante (généralement moins de 15%).

Globalement, la fonction objectif MinQ semble être la plus équilibrée, dans le sens où sa performance est généralement la meilleure, et que dans dans le cas contraire, la différence n'est jamais trop significative. De plus, une fois que sont caractérisées les $f_l(\rho_l)$ pour chaque lien, l'algorithme de partage de charge n'est pas plus compliqué que celui de MinMaxU (considéré comme le plus simple de tous). Ce dernier aspect est justement un des points faibles de MaxU : pour calculer sa fonction de coût des chemins ϕ_P, chaque lien doit mesurer le trafic de chaque paire OD. Il faut également mettre en oeuvre un mécanisme pour qu'un lien détecte s'il constitue le *bottleneck* d'un chemin. Même si dans un réseau MPLS ces mécanismes sont envisageables (par exemple mesurer $d_{P_{si}}$ est très simple), c'est une complexité additionnelle qui doit être prise en compte.

Afin de conclure sur ces travaux sur le partage de charge, nous avons conduit une étude comparative entre des mécanismes de DLB et de RR (Routage Robuste). La conclusion la plus importante est qu'une seule et unique configuration de routage n'est pas suffisante quand le trafic est relativement dynamique. La performance de RR est très pauvre quand il est confronté à des demandes de trafic qui n'étaient pas dans l'ensemble d'incertitude considéré, ou quand cet ensemble d'incertitude veut prendre en compte un nombre trop élevé de possibilités de demandes. Une forme de dynamisme est nécessaire, que ce soit par l'utilisation de DLB ou par un Routage Réactif et Robuste (RRR, *Robust Reactive Routing*) [23].

RRR calcule une configuration de routage nominale pour le trafic normal, et a un routage alternatif (en utilisant les mêmes chemins) pour des situations anormales. Pour détecter des telles situations, il est nécessaire de mesurer la charge des liens et de les envoyer à une entité centrale. DLB fait ces mêmes mesures, mais il doit actualiser la distribution de trafic à une échelle temporelle relativement courte. La complexité supplémentaire est donc la distribution de ces mesures aux routers d'entrée, et l'actualisation de la distribution trafic en temps réel. Nos résultats démontrent que cette complexité peut ne pas être nécessaire si la variabilité du trafic est limitée. Par contre, l'utilisation de DLB est très intéressant quand la variabilité est importante, et obtient des résultats meilleurs que RRR. De plus, si les anomalies (permettant de basculer sur des configurations de routage alternatives) peuvent ne pas être correctement détectées, la seule solution efficace est alors DLB. Il faut aussi noter que RR considère uniquement la variabilité au niveau trafic, alors que DLB est plus général puis qu'il considère tout type de variabilité (e.g. pannes de liens).

Abstract

Network convergence and new applications running on end-hosts result in increasingly variable and unpredictable traffic patterns. By providing origin-destination pairs with several possible paths, Dynamic Load-Balancing (DLB) has proved itself an excellent tool to face this uncertainty. The objective in DLB is to distribute traffic among these paths in real-time so that a certain criteria or objective is fulfilled. In these dynamic schemes, paths are established a priori and the amount of traffic sent through each of them (traffic distribution) depends on the current traffic demand and network condition. This thesis studies several possible DLB schemes. In this sense, it is separated in two parts, depending on the considered architecture.

The first part considers a network in which resources are reserved and used exclusively by each path. This simplification allows us to propose the use of a novel mechanism to manage these paths: *Cross-Protect*. This flow-aware Traffic Engineering (TE) technique enables performance guarantees for both streaming and elastic traffic and its configuration relies on only two parameters, which may be easily mapped to target performance guarantees. The advantage of Cross-Protect with respect to, for instance, DiffServ, is that no packet marking is required. Instead flows are classified implicitly based on their transmission rate. We analyze a Cross-Protect router and derive analytical formulae for its most important QoS parameters. Based on this mechanism, we propose a new flow-aware DLB scheme. In this case, approximate formulae for its QoS parameters are derived. Our simulations indicate that the results obtained by this new scheme are much better than classical static load-balancing schemes.

In the second part we consider the more general case of a “shared-resources” network. In this case, DLB is mathematically defined in terms of an objective function that must be optimized. The first contribution of this part is a new objective function, based on ideas from congestion control. The rationale behind it is to further maximize the utility obtained by each flow, but without changing the congestion control mechanism. Instead, this maximization is performed by load-balancing, which indirectly controls the flow’s obtained rate by choosing its path.

We also study another possible objective function, known as *Minimum Congestion*

Load-Balancing. In this case, congestion in each link is measured by a certain continuous, increasing and convex function $f_l(\rho_l)$, and the objective is to minimize the total congestion on the network (i.e. $\sum_l f_l(\rho_l)$). Typically, due to its simplicity and versatility, the link mean queue size is used as this link congestion function. However, most DLB mechanisms require an analytical expression of this mean size, for which simplistic models are assumed (e.g. *M/M/1*). Instead of such an arbitrary choice, we present a framework that does not assume any particular model for the queue size function, and instead learns it from measurements. This way, we converge to an excellent approximation of the real minimum congestion traffic distribution.

The optimum of these two objective functions may be characterized as the equilibrium of greedy OD pairs (i.e. each OD pair strives to minimize a certain path cost function ϕ_P). This means that distributed greedy algorithms may be used to converge to the optimum. In particular, we propose and study a variation of the *Incrementally Adaptive Weighted Majority* Algorithm (iAWM), an implementation of the so-called *no-regret* algorithms. The advantage of this new algorithm is that it is completely self-regulated. That is to say, no complicated parametrization is needed to guarantee its rapid convergence to the optimum. The performance of the algorithm is verified by means of several packet as well as flow-level simulations.

We also present a thorough study of the schemes discussed so far. Firstly, we compare three objective functions in terms of their obtained performance: our utility maximization proposal, the minimum-congestion schemes, and the equilibrium of greedy OD pairs that strive to minimize the maximum link utilization in the path (which models other previously proposed load-balancing schemes). The obtained results indicate that the the minimum-congestion objective function is the most balanced of the three, in the sense that it generally outperforms the other two, and when it does not, the difference is not significant. Secondly, we present several simulations regarding our regression based minimum-congestion framework. An important conclusion of this study is that the choice of $f_l(\rho_l)$ is not crucial in terms of the link utilization, but is very important with respect to the obtained total congestion. For instance, if we were to use the *M/M/1* model for $f_l(\rho_l)$, the incurred increase in the total congestion may easily exceed 10%, and can go as high as more than 100%.

The final contribution of the thesis is to present a comparative analysis of DLB and Robust Routing (RR), which many consider as the alternative to DLB. RR copes with traffic uncertainty in an off-line preemptive fashion, computing a stable routing configuration that is optimized for a large set of possible traffic demands. Much has been said and discussed about the advantages and drawbacks of each approach, but very few works have tried to compare the performance of both mechanisms, particularly in the same network and traffic scenarios. Our study brings insight into several Robust Routing algorithms, evaluating their virtues and shortcomings and presenting new mechanisms to improve previous proposals. Among others, such a study intends to help network operators in choosing an adequate mechanism to cope with traffic uncertainty.

Contents

Chapter 1

Introduction

1.1 Context and Motivation

Traffic Engineering (TE) is defined as the aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational networks [1]. In particular, we will focus on the last aspect; i.e. enhancing the performance of an operational network. This problem was extensively studied in the early 90s, since resources were considered scarce and expensive. However, the cost of core link capacities dramatically decreased in subsequent years, which resulted in considering a simple upgrade in this capacity as the panacea of all problems.

Nevertheless, as network services and Internet applications evolved, network traffic has become increasingly complex and dynamic. The convergence of data, telephony and television services on an all-IP network directly translated into a much higher variability and complexity of the traffic injected into the network. To make matters worse, the presence of unexpected events such as network equipment failures, large-volume network attacks, flash crowd occurrences and even external routing modifications induces large uncertainty in traffic patterns. Moreover, current evolution and deployment-rate of broadband access technologies (e.g. Fiber To The Home) only aggravates this uncertainty.

But these are not the only problems network operators are confronted with. The ever-increasing access rates available for end-users we just mentioned is such that the assumption of infinitely provisioned core links could soon become obsolete. In fact, recent Internet traffic studies from major network technology vendors like Cisco Systems forecast the advent of the Exabyte era [2, 3], a massive increase in network traffic driven by high-definition video. Furthermore, there are new emerging architectures in which resources are intrinsically scarce (e.g. Wireless Mesh Networks). Thus, simply upgrading link capacities may no longer be an economically or technically viable solution. Moreover, even if overdimensioning would be possible, its environmental impact is not negligible. For instance, the Information and Communication Technology sector alone is responsible for around 2% of the man made CO_2, a similar figure to that of the airline industry, but with higher increasing perspectives [24]. An efficient and responsible usage of the resources is then essential[1].

In the light of this traffic scenario, TE has regained the interest of the academic as well as the industrial community. More to the point, network operators are now, more than ever, in need of Traffic Engineering mechanism which are **efficient** (make good use of resources), **robust** with respect to network variations (changes in traffic demands, or characteristics of transported flows) and more **tolerant** (in case of node/link failures). As the network size increases, its management may become a very complicated task. That is why network operators are also interested in the **automatization** (as much self-configuration as possible) of the TE mechanisms they implement.

Dynamic load-balancing (DLB) [4, 5, 6] is a TE mechanism that meets these requirements. If an origin-destination (OD) pair is connected by several paths, the problem is simply how to distribute its traffic among these paths in order to optimize a certain objective function. In these dynamic schemes, paths are configured a priori and the portion of traffic routed along each of them (traffic distribution) depends on the current traffic demand and network condition. Ideally, the traffic distribution is set so that at every instant the objective function is optimized. **Efficiency** in DLB flows from this last characteristic. In this sense, there exists several possible objective functions. For instance, [4] defines a certain increasing, continuous and convex function $f_l(\rho_l)$ that measures the congestion on link l (where ρ_l is the load on the link), and the objective is to minimize the total congestion on the network $\sum_l f_l(\rho_l)$. On the other hand, the objective in other DLB mechanisms such as [5, 6] is simply to minimize the maximum link utilization ($u_l = \rho_l/c_l$, where c_l is the capacity of the link).

In practice, this "always-optimized" characteristic is achieved by means of a distributed algorithm periodically executed by every ingress router based on feedback from the network. As long as the traffic distribution is updated frequently enough, DLB will be **robust**, and its dependence on the network condition makes it naturally **tolerant** too. Finally, its distributed nature makes it **automated**. However, it is precisely this last characteristic that constitutes the most challenging aspect of DLB. In fact, the deployment of DLB has been, to say the least, limited. Network operators are reluctant to use dynamic mechanisms mainly because they are afraid of a possible oscillatory behavior of the algorithm used by each OD pair to adjust their traffic distribution (as the early experiences in ArpaNet has proved [7], these concerns are not without reason). Indeed, for these adaptive and distributed algorithms, a tradeoff between adaptivity (convergence speed) and stability must be found, which may be particularly difficult in situations where abrupt traffic changes occur.

Concerning stability, the most prominent alternative to DLB is *Robust Routing* (RR) [8, 9]. In RR, traffic uncertainty is taken into account directly within the routing

[1] To learn more about this emerging discipline, the interested reader should consult works related to so called "green networking" [25]

optimization, computing a *single* routing configuration for all traffic demands within some uncertainty set where traffic is assumed to vary. This uncertainty set can be defined in different ways, depending on the available information: largest values of links load previously seen, a set of previously observed demands, etc. The criteria to search for this unique routing configuration is generally to minimize the maximum link utilization over all demands of the corresponding uncertainty set. While this routing configuration is not optimal for any single demand within the set, it minimizes the worst case performance over the whole set. Those who advocate the use of RR claim that there is actually no need to implement supposedly complicated dynamic routing mechanisms, and that the incurred performance loss for using a single routing configuration is small when compared with the increase in complexity. However, to date, no serious comparative study between DLB and RR has been performed.

In this thesis we will study and propose various DLB mechanisms, differing in two important aspects. The first difference resides in the assumption, or not, that resources are reserved for each path. The second lies on the objective function. The performance obtained from the network clearly depends on the choice of this function. However, to the best of our knowledge, a performance benchmarking of the most classic objective functions for DLB has not been carried out so far, a problem we address in this thesis. For the case in which no reservations are performed, we will study and compare several objective functions, including the ones we just discussed and a proposal of ours. We will also propose and study a new distributed algorithm to attain the optimum of these objective functions. Its most notable characteristic, and its advantage with respect to previous proposals, is its complete self-configuration (that is to say, convergence is guaranteed without any parametrization). Finally, we present the first complete and fair comparative study between DLB and RR. In particular, we will try to answer the question of which scheme is more convenient in each given situation, and highlight some of their respective shortcomings and virtues. In the following section we summarize the main contributions of the thesis.

1.2 Summary of Contributions

1.2.1 First Part: Reserved Resources

We have separated the thesis in two parts. The first one assumes that resources have been reserved and are exclusively used by each of the paths. A typical use of these *tunnel-based* architectures (e.g. MPLS-TE, T-MPLS, PBT, GELS) is VPN provisioning in metro and backbone networks. Up to now, resources associated with each of these tunnels are usually specified using token buckets. This classical QoS scheme has been described as inefficient and inappropriate for characterizing traffic aggregates [10]. It has been verified that the burst parameter needs to be excessively large, even when the rate parameter is significantly greater than the actual flow mean rate, to achieve a given non-conformity probability. This means that relying on declared token bucket traffic parameters is typically very conservative, which leads to wasted resources.

An attractive alternative is to use the *Cross-Protect* mechanism [11, 12]. This flow-aware TE technique allows performance guarantees for both streaming and elastic flows while preserving the user-network interface (i.e. neither packet marking nor per-flow signalling is required). Moreover, its configuration relies on only two parameters, which are easily mapped to target performance guarantees. These guarantees are given independently of the flow characteristics, a major difference with token buckets. We propose that provider edge routers implement the Cross-Protect mechanism on a per-tunnel basis in the aforementioned architectures. Since resources are reserved for each tunnel, the mechanism needs to be implemented in the edge only, minimizing deployment complexity. We analyze a cross-protect router and derive analytical formulae for its most important QoS parameters.

To improve resiliency and make a better resource utilization, we will extend the aforementioned TE scheme with a simple flow-aware load-balancing mechanism. This new DLB algorithm leads to better results than classical static load balancing schemes. We have also analyzed this algorithm and derived tight approximations for the most important QoS parameters in this case.

1.2.2 Second Part: Shared Resources

In the second part, we will not consider any tunnel-management mechanism in particular, and will analyze DLB mechanisms that use measurements that are available in most commercial routers (e.g. link load). The first contribution of this part is to propose a new objective function for DLB specifically designed for elastic traffic (which we shall note MaxU, as in Maximum Utility). In this proposal we aim at further maximizing the total utility obtained by flows, but without changing their congestion control mechanism. This maximization is achieved instead by changing the amount of traffic routed along each path, thus changing the flows' mean rate. We also present a characterization of this optimum, in the form of a path cost function ϕ_P (where P is a path). More precisely, if each OD pair greedily strives to minimize the cost it obtains from each of its paths, the resulting equilibrium will be the optimum.

As mentioned before, another possible objective function in DLB, and the second objective function we will consider, is to define a convex, continuous and increasing $f_l(\rho_l)$ and to minimize the total congestion $\sum_l f_l(\rho_l)$. Typically, the mean queue size is used as $f_l(\rho_l)$, mainly due to its versatility (a big queue means bad performance for all traffic) and simplicity (the mean queue size aggregates naturally across links as the sum). However, most DLB schemes require an analytical formula of this mean queue size, for which classic and oversimplistic models (e.g. *M/M/1* [26]) are used [4]. As we shall show, this simplistic choice results in an actual total congestion that may be significantly larger than the optimum. Instead of making a choice at all, and as a second contribution of this part, we propose a framework that *learns* this function from past measurements while making no assumption on it (other than the hypothesis on its shape mentioned before). To achieve this we present two different regression methods. We will first consider *Convex Nonparametric Weighted Least Squares* (CNWLS), a variation of the original unweighted CNLS [17]. Although this method obtains an excellent precision by means of solving a simple QP problem, it presents scalability issues as the number of available measurements increases. As an alternative, we present the *Convex Piecewise Linear Fitting* (CPLF) [18], a heuristic that very rapidly finds a regression function that reasonably adjusts the measurements. However, the precision of CPLF may be very poor when compared to CNWLS.

It can be proved that the optimum for this second objective function may also be characterized as the equilibrium of greedy OD pairs that strive to minimize a certain path cost function ϕ_P. We will then study possible methods to converge to this equilibrium. A third contribution of this part is to present a DLB algorithm based on so-called *no-regret* algorithms. The authors of a recent paper [21] proved that if all OD pairs use algorithms of this kind, convergence to this greedy equilibrium is guaranteed. In particular, we will use a variation of the *Incrementally Adaptive Weighted Majority Algorithm* (iAWM) [22], which presents the advantage of being completely self-regulated, thus avoiding any tricky parameter setting and the reactivity-stability tradeoff we mentioned before. Furthermore, we will present certain adaptations of the algorithm that are necessary to enable its use in the presence of non-stationary traffic. The algorithm is illustrated by several packet as well as flow level simulations.

A fourth contribution of this part is to provide a thorough performance study of all the schemes considered so far. We will first compare three objective functions, using two real topologies and several real traffic demands. The considered objective functions are MaxU, the regression-based minimum congestion framework described above, and the equilibrium of OD pairs that strive to minimize $\phi_P = \max_{l \in P} u_l$ (which models the DLB mechanisms presented before and originally introduced in [5, 6]). This is an important study that will allow us to better understand the tradeoffs between each objective function, and will help network operators at the moment of choosing a possible dynamic load-balancing mechanism. We will also study the regression-based framework more in detail and show, for instance, that using the *M/M/1* model instead of our learned function results in an increase in the total congestion that may easily exceed 10%, and can go as high as more than 100%. We will also study the impact of the imprecision of CPLF, concluding that it is not very significant.

The final contribution of the thesis is presenting a fair and comprehensive comparative analysis between RR and DLB mechanisms. The analysis is comprehensive as it evaluates the performance of both mechanisms based on different performance indicators and considering normal operation as well as unpredicted traffic events. We believe the comparison is fair because it considers the particular characteristics of each mechanisms under the same network and traffic conditions. To the date and to the best of our knowledge this is the first work that conducts such a comparative evaluation, necessary indeed not only from a research point of view but also for network operators who seek for proper solutions to face future network scenarios. Based on this comparative analysis we develop and evaluate new variants of RR, improving some of its shortcomings. These new variants, along with the legacy RR methods, will also be considered in the comparative study.

1.3 Structure of the Thesis

The next two chapters constitute the first part of the thesis, concerning the "reserved resources" model. Chapter 2 is devoted to the presentation and analysis of the Cross-Protect mechanism. We will discuss how to integrate this mechanism into the network, and derive a close formula for its most important QoS parameters based on some characteristics of the input traffic. Chapter 3 defines the dynamic load-balancing scheme based on Cross-Protect, and also derives analytical formulae to estimate the same parameters.

The second part of the thesis starts in Ch. 4, where we discuss the most relevant works related to Dynamic Load-Balancing. In particular, we will present some static routing schemes (including RR), discuss with more detail the most important proposals in DLB, and present the problem of congestion control. Some of the ideas presented in this last section will be revisited in Ch. 5, where we present MaxU.

Chapter 6 will begin by characterizing the demand vector that minimizes $\sum_l f_l(\rho_l)$. We will then discuss how to learn the congestion function $f_l(\rho_l)$ from measurements. In particular, we will present CNWLS and CPLF, and study their performance (in terms of computation time and precision). We will finish the chapter with a regression example that will be used as a reference in the following chapters.

A distributed method to converge to the equilibrium of greedy OD pairs will be presented in Ch. 7. We will introduce the concept of *Wardrop Equilibrium* and discuss the results pertaining to no-regret algorithms (in particular to its converge). We shall then present iAWM, and by means of a preliminary simulation, justify and introduce its final form: iAWM-R.

The performance study of the schemes will be presented in Ch. 8. By the end of this chapter we will present the comparative analysis between DLB and RR. Chapter 9 concludes the thesis and presents possible extensions and perspectives.

Part I

Reserved Resources

Chapter 2

Cross-Protect

2.1 Introduction

Current evolutions of network architecture show a growing domination of Ethernet technologies. Enterprises have a growing need of interconnecting their local networks (based on Ethernet), which are situated in different physical locations, and is the operator who has to transport the Ethernet frames between the different sites, giving the illusion of a dedicated Ethernet network to each of its clients. The MetroEthernet Forum [27] is presently working on the definition of such metropolitan Ethernet services. An Ethernet service consists of a Ethernet service type, one or more Ethernet service Attributes, and one or more parameter values associated with each Ethernet Service Attribute. Examples of Ethernet service types are Ethernet Line Service (E-Line Service) and the Ethernet LAN Service (E-LAN Service).

This kind of commercially critical services is a typical example of the use of "reserved resources" architectures, which as discussed in Ch. 1 will be the object of this first part of the thesis. One of the most critical parameters in this kind of services is how much traffic the customer can send or receive. In particular, for metropolitan Ethernet services this is defined in what is called the Bandwidth Profile, an Ethernet Service Attribute. Naturally, to make sure the user does not consume more resources than it has paid for, some form of access control is implemented in the ingress router. As is generally the case in this kind of architectures, a mechanism based on token buckets has been chosen. More exactly, two consecutive token buckets are used. The parameters that configure these token buckets, as well as the performance objectives, are part of the agreement between the service provider and the customer (the Service Level Agreement, SLA). If a packet is compliant with the first token bucket, it should be delivered to its destination and the SLA performance objectives apply. If the packet is not compliant with the first token bucket, but it is with second one, it should be delivered to its destination but no performance guarantees are assured. In any other case, the packet is discarded.

Studies have been carried out within both the IEEE [28, 29] and the IETF [30] standardization bodies to define architectures that provide such services. These approaches rely on tunnels to transport Ethernet frames either natively (e.g. PBT, GELS) or using MPLS (e.g. PWE3, VPLS). In such schemes, the use of token buckets as traffic descriptors for SLA specifications has been strongly criticized in the past. In particular, in [10] they have been described as inefficient and inappropriate for characterizing traffic aggregates. This *a priori* traffic descriptor is a very poor characterization of the actual traffic, which leads to the customer to systematically overestimate the traffic parameters. This means that their declared values are of little use for resource allocation.

We propose an alternative Flow Aware TE approach for carrier class Ethernet networks providing services as those defined by the Metro Ethernet Forum. We assume that it is possible to associate a capacity to a given “tunnel” using for instance RSVP-TE in the context of MPLS, or GMPLS for native Ethernet. We further assume that multiple paths could be established between origin-destination (OD) pairs, a case we shall analyze in Ch. 3.

We propose that provider edge routers implement the *Cross-Protect* mechanisms [11, 12] on a per tunnel (or Label Switch Path, LSP)[1] basis. Cross-Protect allows performance guarantees for streaming and elastic flows while preserving the user-network interface of the best effort Internet, i.e. neither packet marking nor per-flow signalling is required to differentiate explicitly between streaming flows, requiring low packet loss and delay, and elastic flows, requiring “as fast as possible” document transfer. Instead, Cross-Protect routers identify user-defined flows on the fly and implement per-flow scheduling using Priority Fair Queueing (PFQ). Measurement-based admission control is additionally employed to maintain the fair rate sufficiently high to provide streaming like quality for flows of peak rate up to a chosen threshold.

The adopted flow-aware networking approach provides precise QoS guarantees (suitable for SLA specifications) and is more cost-effective than substantially over-provisioned best-effort networks. Since resources are reserved on a per LSP basis, the Cross-Protect mechanisms need only be implemented at the network edge (i.e. transparent to internal nodes). In the following section we will describe Cross-Protect with more detail and discuss how to integrate it into the network. We will then analyze its performance, and derive a closed-form formula of its most important QoS parameter: the flow blocking probability due to its admission control.

Before finishing this introduction, let us highlight the fact that our results and ideas are applicable to any connection-oriented environment in which a certain capacity can be guaranteed to tunnels. We have considered Metro Ethernet in particular due to its flagrant lack of efficient TE mechanisms. Moreover, as mentioned before, Cross-Protect needs only be implemented in the edge routers. This means that another possible case

[1] In the sequel, the terms tunnel, virtual path, and LSP are used without differentiation.

scenario could be a simple DSL access connection, where Cross-Protect is implemented in the customer's modem (or, as discussed in[31], any other packet-scheduling algorithm with the same properties as Cross-Protect).

2.2 Proposed Architecture: Cross-Protect

IP traffic can be broadly classified into two categories with clearly different QoS requirements: *elastic* and *streaming*. Elastic traffic is generated by document transfers (e.g. web page, MP3 music file). Associated Elastic flows require "as fast as possible" transfers. Streaming traffic, on the other hand, is produced by real time audio and video applications (e.g. video streaming, VoIP conversation), and requires transparent delivery, i.e. low packet loss rate and delay.

IP networks are meant to support all kinds of service, each coming with its own specific requirements. These are almost always met in present IP backbone networks, in particular, thanks to substantial overprovisioning. Indeed, the inability of network operators to distinguish between kinds of traffic leads to undiscriminated handling of streaming and elastic packets (FIFO queueing). Of course, explicit marking using Diffserv would be possible, but this comes at a certain cost and raises several other issues such as trust in an inter-domain setting.

Overprovisioning is actually a quite satisfactory solution, as it satisfies most users' requirements, while inducing very low operational costs. However, the network remains vulnerable to ill-behaved users, as delivering reasonable QoS depends on users' cooperation in implementing end-to-end congestion control (TCP or alternative "TCP-friendly" protocols). More to the point, the network is oblivious to specific requirements relative to different kinds of traffic. In particular, in the event of an overload situation due, for instance, to a link or equipment failure (as overprovisioning may not take into account all kinds of risk), all traffic is likely to suffer QoS degradation, including critical applications such as voice or TV broadcast.

Moreover, following [10], we consider that traffic engineering and traffic control for predictable QoS is most conveniently performed at *flow* level, rather than *packet* or *aggregate* level. A flow corresponds to some application instance, transported by the network. It is precisely at this level that the user perceives QoS. A flow may, for instance, correspond to a web page download, a voice call, or a music or video streaming. Although in practice a more exact definition is needed, for the present study purpose, it is sufficient to define a flow as a stream of packets sharing common header attributes (e.g. the TCP/IP 5-tuple, or the IPv6 flow label combined with the source or destination address) and a maximum inter-packet time.

According to [10, 32, 33, 34], the integration of both streaming and elastic flows can be achieved without deteriorating their respective QoS, as long as bufferless multiplexing conditions are assured for streaming flows (handled with priority) and remaining resources are fairly shared between elastic flows. A possible way to achieve this integration is to use *Cross-Protect*, an implementation of the so called *Flow Aware Networking* described in [11]. A Cross-Protect router consists of two traffic control components. On the one hand, a Priority Fair Queueing (PFQ) scheduler, which is a simple adjustment of a fair queueing scheduler (e.g. Start time Fair Queueing (SFQ) [35] or Deficit Round Robin (DRR) [36]), that implicitly differentiates between streaming and elastic flows. On the other hand, an admission control mechanism that guarantees a minimum QoS to accepted (or protected) flows, as well as the scalability of the scheduler by limiting the number of flows that need to be handled by the scheduler at any given time.

The PFQ scheduler implicitly classifies flows as streaming or elastic on the fly based on the following principle. If a streaming flow is transmitting at a rate less than the current instantaneous fair-rate (in practice, this is manifested by the flow being not backlogged), then its packets are classified as streaming and given priority; whereas elastic flows (i.e. backlogged flows) share the remaining capacity in a processor sharing (PS) way. Hence, fair sharing is *enforced* by the queueing discipline, and does not rely on the TCP friendliness of the congestion control protocol implemented by end users.[2]

Admission control is used to limit the streaming load (referred to as *priority load* in the remainder, and denoted PL) to be under a relative threshold γ_s, and the current rate obtained by elastic flows (i.e. *fair rate*, denoted in the sequel FR) above another threshold γ_e. If any of these conditions is not satisfied, new flows are blocked. This way, a minimum bandwidth for elastic flows is guaranteed, and the load induced by streaming traffic is controlled. Typical values for γ_s and γ_e are around 80% and 1% respectively, which we have used throughout our simulations.

A simplified diagram of the mechanism is illustrated in Fig. 2.1. It is worth noting that the implicit classification is continuously applied to all ongoing flows and not only upon flow arrival to the router. This means that, as the rate of flows evolves, their classification may change too. For instance, the first few packets of a TCP connection in the Slow Start mode are typically assimilated to streaming packets.

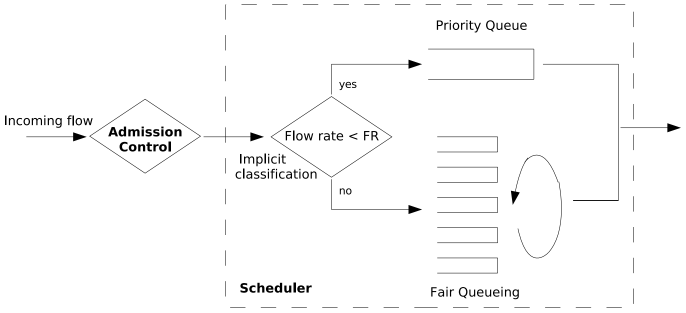


Figure 2.1: Cross-Protect diagram

A concern present in all flow-level scheduling mechanisms realizing fair bandwidth sharing is scalability. As discussed in [37], the complexity of such schedulers depends

[2] However, TCP is required for the elastic traffic to *find* its fair rate.

on the number of bottlenecked flows, and not in the number of active ones. Although the latter increases with the link capacity, the former does not and remains small (in the orders of tens). In PFQ in particular, another possible concern is the apparent need of per-flow calculations for the implicit flow classification. However, per-flow rate measurements are not made since the scheduling algorithm implicitly gives priority to flows transmitting at less than the fair-rate. FR and PL are periodically measured to be used for admission control purposes only. We refer the interested reader to [12] for more details.

Our proposal for TE in the MetroEthernet architectures is to implement Cross-Protect in the edge routers (thus, with no impact on core routers). Since edge routers are connected via tunnels of a given capacity, overload control (in the form of admission control) can be performed at ingress nodes only. This has the advantage of restricting flow-aware operations to the network edge, and preserving the simplicity of packet forwarding in the core. An example of such a TE architecture, in the case of MPLS tunnels, is illustrated in Fig. 2.2. The Bandwidth Profile can be now specified with the total path capacity and both thresholds (γ_s and γ_e).

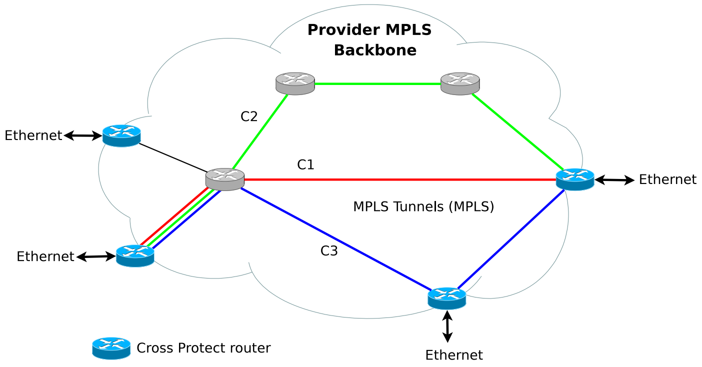


Figure 2.2: Proposed architecture

We have already mentioned that a minimum throughput is guaranteed to elastic traffic, and negligible delay and jitter is assured to streaming traffic. The only remaining parameter is the flow blocking probability due to the admission control mechanism. There is a clear trade-off between these parameters which needs to be understood. In what follows, we analyze a Cross-Protect router and derive an analytical formula for the flow blocking probability, which depends on the chosen thresholds and in the main traffic characteristics.

2.3 Performance Analysis

2.3.1 Model Description

We model the arrival of elastic flows into our system as a Poisson process of intensity λ_e. Each flow of this type is characterized by the workload offered to the system, distributed as σ_e. We denote by $b_e = \mathbb{E}(\sigma_e)$ the mean offered workload and by $d_e = \lambda_e b_e$ the elastic load in the system. Typically, a reasonable choice for the distribution of σ_e will be heavy tailed, where most flows are very small and the majority of the traffic is contained in few but very long flows.

Streaming flows are modeled as circuit type traffic in telephone networks, and are therefore characterized by an intrinsic rate r (that we suppose fixed so as to simplify the analysis) and a random duration, with mean τ_s. We further assume that these flows arrive also as a Poisson process of intensity λ_s. The mean workload introduced by streaming traffic is then $r\tau_s$, thus $d_s = \lambda_s \tau_s r$ represents the streaming load of the system.

Suppose C is the virtual path (or LSP) capacity. At the flow time scale, we suppose that C is fixed. The described model for the system can be represented as a PS network of queues with state dependent service rate as shown in Fig. 2.3. We denote by $x = (x_s, x_e)$ the number of ongoing streaming and elastic flows, respectively.

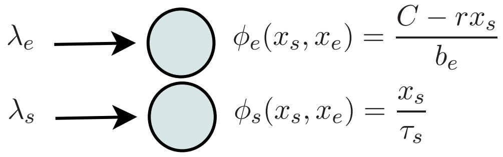

λ_e
$\phi_e(x_s, x_e) = \frac{C - rx_s}{b_e}$
λ_s
$\phi_s(x_s, x_e) = \frac{x_s}{\tau_s}$

Figure 2.3: PS network representing a Cross-Protect router

One node represents the streaming service component, with arrival rate λ_s and service rate $\phi_s(x_s, x_e) = x_s/\tau_s$, which is the service rate of an Erlang system. The other node represents the elastic service component, with arrival rate λ_e and service rate $\phi_e(x_s, x_e) = (C - rx_s)/b_e$, which is the service rate of a PS queue with the workload characterized by $\mu_e = 1/b_e$ and a variable capacity, that depends of the current available bandwidth left by the streaming flows (we are supposing that elastic flows realize their fair rate instantly).

In terms of the state x, the admission control conditions introduced in the previous subsection constrain the state space, which is given by:

$$rx_s \leq \gamma_s C \quad \text{and} \quad \frac{C - rx_s}{x_e} \geq \gamma_e C \tag{2.1}$$

It is important to observe that the implicit classification of the Cross-Protect router is not modeled. This means that in our model flows are known to be elastic or streaming a priori, which is valid as long as all streaming flows' transmission rate r remains smaller than the fair rate.

2.3.2 Analysis

The general analysis of the integrated model as presented before is not feasible, even if we make the simplifying hypothesis of exponential workloads. Instead, we apply the same idea as in [38] and assume that we can separate the time scales of streaming and elastic flows. This allows for the study of the elastic queue as if the streaming queue were in the stationary regime (also known as the Quasi-Stationarity (QS) assumption).

The justification behind the QS assumption is the following: in general, the mean duration (τ_s) of streaming flows is much greater than the corresponding mean duration of elastic flows. For a given load level d_s, d_e, this means that the arrival rate of streaming and elastic flows verify that $\lambda_s \ll \lambda_e$. This relationship implies that the events associated to streaming flows occur sparsely in time, and allows the elastic queue to behave in a *quasistationary regime*, under which the elastic flows see the queue as a constant rate server.

The QS assumption allows us to analyze the elastic queue as an $M/G/1-PS$ queue with capacity $C - rx_s$ for x_s streaming flows present in the system. As a consequence, we can write the probability of x_e elastic flows in the system, given there are x_s flows in the streaming queue, as follows:

$$P(N_e = x_e | N_s = x_s) = \frac{1 - \rho_e(x_s)}{1 - \rho_e(x_s)^{N_e^{max}(x_s)+1}} \rho_e(x_s)^{x_e} \tag{2.2}$$

for $0 \leq x_e \leq N_e^{max}(x_s)$, where:

$$\rho_e(x_s) = \frac{\lambda_e b_e}{C - rx_s} \tag{2.3}$$

$$N_e^{max}(x_s) = \left\lfloor \frac{C - rx_s}{\gamma_e C} \right\rfloor \tag{2.4}$$

We can therefore write the blocking probability conditioned to x_s from (2.2) by making $N_e = N_e^{max}(x_s)$:

$$B_e(x_s) = \frac{1 - \rho_e(x_s)}{1 - \rho_e(x_s)^{N_e^{max}(x_s)+1}} \rho_e(x_s)^{N_e^{max}(x_s)} \tag{2.5}$$

To derive the blocking probability under stationary regime, we must average the conditional blocking probability $B_e(x_s)$ with respect to the stationary distribution $\pi_s(x_s)$ of the streaming queue:

$$B = \sum_{x_s=0}^{N_s^{max}} B_e(x_s) \pi_s(x_s) \tag{2.6}$$

where $N_s^{max} = \lfloor \gamma_s C/r \rfloor$.

The streaming queue would behave exactly like an Erlang one with $A_s = \lambda_s \tau_s$ load (in Erlangs), if streaming flows were only rejected due to the priority load condition. But this is not the case, since both conditions on priority load and fair rate are applied independently of the traffic type. So, the resulting process is a birth-death one, with the birth rate equal to $\lambda_s(1 - B_e(x_s))$ (to account for the blocking state of the elastic queue) and the death rate is equal to x_s/τ_s in state x_s. The stationary distribution then becomes:

$$\pi_s(x_s) = \pi_s(0) \frac{A_s^{x_s}}{x_s!} \prod_{i=0}^{x_s-1} (1 - B_e(i)) \tag{2.7}$$

where $\pi_s(0)$ can be obtained from the normalization condition.

Equations (2.5)), (2.6) and (2.7) then give the blocking probability B of the system.

2.3.3 Packet-level simulations

In order to verify the previous analysis, we conducted several packet level simulations using ns-2 [39], with the Cross-Protect implementation used in [12]. A comparison between the blocking probability obtained by simulation and the corresponding estimation can be seen in Fig. 2.4 for the single-server case, where the x-axis is the link utilization (i.e. $\rho = (d_e + d_s)/C$). The case scenario is a mix of elastic traffic whose workload follows a Pareto distribution with mean 20 kB and streaming traffic with a fixed rate of 10 kbps and a duration of 20 sec. Streaming traffic represents 20% of traffic and the channel has a total capacity of 1Mbps.

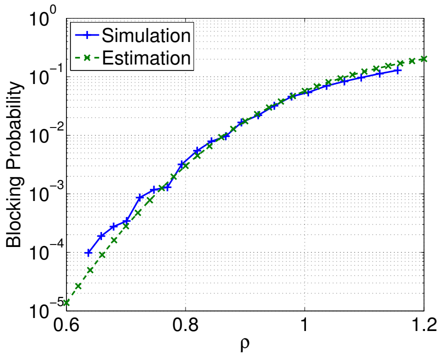


Figure 2.4: Blocking probability estimation and simulation results for an isolated Cross-Protect router

Although the analysis did not take into account packet level dynamics of TCP (e.g. Slow Start), the estimation proves to be very accurate. However, in the presence of TCP packet level dynamics, the implicit classification system assimilates part of TCP traffic to streaming traffic, notably during the Slow Start phase. This results in more priority traffic than expected by the model. If the probability that a flow is blocked due to the PL condition is negligible, then the estimation model will yield accurate predictions. Otherwise, the model tends to underestimate the blocking probability. This effect is more clear at low loads, where it takes more time to TCP flows to converge to their faire rate, thus resulting in more priority traffic than expected by the model as explained before.

Chapter 3

Load-Balancing with Cross-Protect: Simple Greedy Policy

3.1 Introduction

As discussed in Ch. 1, we may improve performance and enhance resilience to sudden traffic fluctuations and failure-induced overload events by means of dynamic load balancing. For instance, Fig. 2.2 illustrates how several LSPs can connect a pair of edge nodes. Such methods allow the network to respond to changing demands and failures by changing the routing pattern depending on the state of the paths. Since the fair rate and the priority load of each LSP are measured for admission control purposes, it is simple and natural to route each flow depending on these measurements.

In the following sections we propose and analyze a load balancing policy for the case when there are several possible tunnels (or LSPs) between source and destination. As in the previous chapter, we study the dependence of the flow blocking probability in the proposed mechanism with the chosen thresholds and the main traffic characteristics. Although the analysis will be similar to that of Sec. 2.3.2, the presence of several paths and their coupled behavior will complicate the analysis and we will only be able to derive approximations.

3.2 Multiple Tunnels and Load balancing Analysis

3.2.1 Related work

Let us assume that traffic consists of elastic traffic only. In this situation, the problem becomes a routing problem between multiple processor sharing queues in parallel. This problem was first studied by Bonomi (see [40] and references therein), and more recently by Koole et al. [41]. Their results show that the optimal policy, in terms of blocking probability and throughput, based on present state information is to send the arriving flows to the queue with the least number of flows/customers (*Join the Shortest Queue*, JSQ) for the case of two identical servers and exponential workloads.

For the general setting, i.e. general workloads and non symmetric systems, the problem is open and the optimal policy is unknown. Hajek in [42] analyzed a more general system allowing different capacities under a Markovian setting. He showed, based on a dynamic programming approach, that the optimal policy is always switch type (i.e. the probability of choosing any of the queues is either 1 or 0), but explicit formulae for the switch curve were not provided. We performed numerical calculations to approximate the switch curve of the Hajek model in a routing setting. The result for exponential workloads, in the two server case, is that the optimal policy in terms of throughput is given by the *exact greedy policy* (EGP):

$$\text{Route to queue } i \Leftrightarrow i = \arg\max_j \frac{C_j}{x_j + 1} \tag{3.1}$$

where x_j is the number of customers in the j–th queue. This is a greedy policy, in the sense that each incoming flow will maximize the fair rate it will receive upon entering service in the system. In the symmetric case, this boils down to the JSQ policy.

Another approach to load balancing in the elastic case is studied by Jonckheere et al. [43]. The authors propose another class of policies, the *balanced routing*, with the property that they are insensitive to the distribution of the flow workloads (meaning that, for instance, the blocking probability only depends on the first moment of the input process). In the class of balanced routing policies, the optimal in terms of blocking probability is defined and is characterized by the following probabilities:

$$\text{Route to queue } i \text{ with prob. } p_i(x) = \frac{n_i - x_i}{\sum_j n_j - x_j} \tag{3.2}$$

where x_j is again the number of customers in the j–th queue and $n = (n_1, \ldots, n_N)$ is the maximum number of customers allowed in each queue. We shall call this policy the *optimal balanced policy* (OBP). The blocking probability B_{obp} in this case can be easily computed.

3.2.2 Proposed scheme

In the proposed architecture described in Sec. 2.2, we assumed that each pair of ingress-egress nodes are connected via one or multiple LSPs, where the ingress node implements Cross-Protect mechanisms to realize QoS. From the flow perspective, this means that there are N virtual paths with capacities $C_1, \ldots, C_N$, that are fixed at the flow time scale. The load balancing between these virtual paths becomes then a routing problem between multiple queues. Note that these queues can be treated as equal choices in terms of resources, that is, the objective here is to minimize the flow blocking probability by choosing the right routing policy. At this time scale, we are not concerned with minimizing the number of physical links composing each route (e.g. via shortest path routing), as resources associated to virtual paths are supposed reserved in advance.

In the Cross-Protect setting, where the current fair rate is already measured, we propose to use the following simplified policy, that we call *simplified greedy policy* (SGP):

$$\text{Route to queue } i \Leftrightarrow i = \arg\max_{j} fair\_rate_j \tag{3.3}$$

This policy is clearly intuitive and is also very simple to implement in the ingress node, not requiring additional measurements. It will also "inherit" the optimality of EGP and, as will be outlined later, OBP. However, corresponding analytical results are hard to derive and some approximations will be needed.

We now focus our attention on the streaming flows long time scale. These flows do not gain any particular advantage from our SGP policy, which is devised with elastic flows in mind. However, streaming flow routing is less critical because it represents the minority of traffic. We sacrifice optimality at the streaming time scale in favor of an implicit policy, which makes no distinction in the type of arriving flows. It must be noted once again that the optimality here refers to the blocking probability in the routing case, the other QoS requirements associated with streaming traffic (i.e. low packet loss and delay) are guaranteed by priority handling and admission control.

3.2.3 Analysis

We model and analyze the load balancing between paths of equal or different capacities. To clarify the exposition, we will only present here in full detail the load balancing between two paths. The case in which more paths are present can easily be derived from the results presented.

We will assume the two thresholds ($\gamma_{si}C_i$ and $\gamma_{ei}C_i$) are equal in both LSPs. Given the two paths are equivalent, it is reasonable to consider these two values to be the same, since they give the minimum QoS each protected flow will receive.

The analysis will be the same as in the isolated case. We will first analyze what happens in the short time scale of elastic flows, estimate the blocking probability given the value of x_{s1} and x_{s2} and then estimate the probability of having x_{s1} and x_{s2} streaming flows in the system so as to make the weighted sum.

Elastic Time-Scale

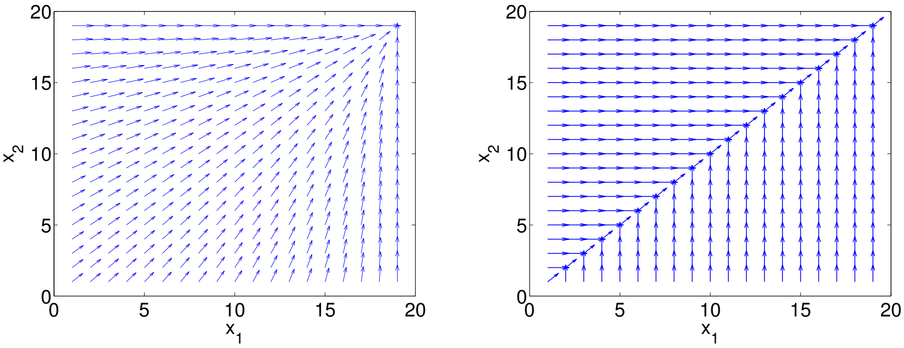
For the analysis of load balancing in the elastic case, we turn our attention to the aforementioned work of Jonckheere et al. [43]. In this case, the blocking probability when using OBP can be computed as $B_{obp} = 1/\delta(n)$ where δ is given by a simple recursive formula:

$$\delta(x) = 1 + \sum_{i=1}^{N} \frac{\phi_i(x_{si}, x_{ei})}{\lambda_e} \delta(x - e_i) \tag{3.4}$$

where

$$\phi_i(x_{si}, x_{ei}) = \frac{C_i - x_{si} r}{b_e}$$
$$x - e_i = (x_1, \ldots, x_i - 1, \ldots, x_N)$$
$$\delta((0, \ldots, 0)) = 1 \quad \text{and} \quad \delta(\hat{x}) = 0 \quad \forall\, \hat{x} \text{ invalid, e.g. } \hat{x} = (0, -1)$$

The usefulness of optimal balanced routing is that, as verified in [43], the equations above provide a very good approximation for the blocking probability of a greedy policy such as SGP. Intuitively, this may be verified by comparing the decision probability vector of OBP and that of a greedy policy. In Fig. 3.1 we present a comparison example, where each arrow is a vector with coordinates (p_1, p_2) for the corresponding x_1 and x_2 ($n_1 = n_2 = 19$). Note how as the number of flows in each queue increases (where the blocking probability is more important) the decision probability vector of both mechanisms is very similar.



(a) OBP

(b) Greedy

Figure 3.1: Comparison in the decision probability vector between OBP and a greedy scheme.

The advantage of OBP over SGP is that its blocking probability calculation rely on simple parameters of the input traffic, like the arrival intensity and mean workload, making design calculations easier. We shall then approximate the loss probability given x_{s1} and x_{s2} by:

$$B_e(x_{s1}, x_{s2}) = \frac{1}{\delta(N_{e1}^{max}, N_{e2}^{max})} \tag{3.5}$$

where N_{e1}^{max} and N_{e2}^{max} were defined in (2.4). Note that, although OBP was used as an approximation in our analysis, we believe that the integration of this policy into

Cross-Protect routers would not be as easy as the proposed one. Our proposal does not require any additional measures and the only operation necessary is to select the path with the highest fair rate. The few number of them limits the complexity of the path selection operation. On the other hand, OBP requires a random assignment with weights which have to be calculated each time a new flow arrives.

Streaming Time-Scale

Now the only remaining issue is to find the steady-state probability of the streaming queues. The main effect of the policy in the system is to maintain equal *fair rates* in each queue, which can be calculated as C_1/x_{e1} and C_2/x_{e2} respectively. If we equal the two fair rates then $x_{e1}/x_{e2} = C_1/C_2$.

So the proportion of flows to the i-th queue is approximately $C_i/(C_1 + C_2)$. Since streaming flows receive the same treatment as elastic ones, they are handed to each queue in the same proportion. This is obviously a gross approximation, but our simulations indicate that this proportion is very accurately maintained for the streaming flows.

Once again, we could calculate $\pi_s(x_{s1}, x_{s2})$ like an Erlang double queue (where each queue can be treated independently, leading to a product form probability) if we assume that the only reason for rejecting streaming flows is the *PL* condition. This is not the case, since streaming flows are rejected due to the *FR* condition too. So, analogous to the single server case, the arrival rate to each streaming queue is finally:

$$\lambda_{si} = \frac{C_i}{C_1 + C_2}\lambda_s(1 - B_e(x_{s1}, x_{s2}))$$

Thus, the streaming part of the system behaves as a birth-death process with transition rates:

$$\begin{aligned} x_s \to x_s + e_i \quad &: \quad \frac{C_i}{C_1 + C_2}\lambda_s(1 - B_e(x_{s1}, x_{s2})) \\ x_s \to x_s - e_i \quad &: \quad x_{si}\frac{1}{\tau_s} \end{aligned}$$

The blocking probability couples the behavior of the queues, making the exact analysis of the system not tractable. To get approximate values of the steady-state probability, we have to numerically solve the global balance equations ($\pi_s Q = 0$ plus the normalization condition $\sum_i \pi_s(i) = 1$). An alternative is to find an upper bound for the system by finding an easier system to analyze with a conservative behavior (i.e. yielding higher blocking probabilities). In this case, such a system is one which tends to hold more customers in the queue than the original one. This can easily be done by simply making the birth-rate bigger. For instance, if we change the original transition rates to:

$$
\begin{aligned}
x_s \to x_s + (1,0) : & \frac{C_1}{C_1+C_2}\lambda_s \left(1 - \arg\min_{x_{s2}} B_e(x_{s1}, x_{s2})\right) = \frac{C_1}{C_1+C_2}\lambda_s(1 - B_e(x_{s1}, 0)) \\
x_s \to x_s + (0,1) : & \frac{C_2}{C_1+C_2}\lambda_s \left(1 - \arg\min_{x_{s1}} B_e(x_{s1}, x_{s2})\right) = \frac{C_2}{C_1+C_2}\lambda_s(1 - B_e(0, x_{s2})) \\
x_s \to x_s - e_i \quad : & \quad x_{si}\frac{1}{\tau_s}
\end{aligned}
$$

This system has the advantage that a product-form holds for its stationary distribution and since it is balanced, it is also insensitive [43]. Its steady-state probability can be obtained by analyzing the system as if each queue were independent, so the joint probability is simply the multiplication of both probabilities. So, the complete system can be bounded as:

$$
\pi_s(x_{s1}, x_{s2}) \underset{st}{\leq} \tilde{\pi}_s(x_{s1}, x_{s2}) = \pi_{s1}(x_{s1})\pi_{s2}(x_{s2}) \tag{3.6}
$$

where

$$
\pi_{s1}(x_{s1}) = \pi_{s1}(0)\left(\frac{C_1}{C_1+C_2}A_s\right)^{x_{s1}} \frac{1}{x_{s1}!}\prod_{j=0}^{x_{s1}-1}(1 - B_e(j, 0)) \tag{3.7}
$$

$$
\pi_{s2}(x_{s2}) = \pi_{s2}(0)\left(\frac{C_2}{C_1+C_2}A_s\right)^{x_{s2}} \frac{1}{x_{s2}!}\prod_{j=0}^{x_{s2}-1}(1 - B_e(0, j)) \tag{3.8}
$$

where $\pi_{s1}(0)$ and $\pi_{s2}(0)$ are obtained from the normalization condition.

The $\underset{st}{\leq}$ means that the balanced system will tend to have more customers in the queue than the original one, thus it will have a higher average number. So, the blocking probability using this system will be an upper bound of the original one.

Total Blocking Probability Bound

Finally, with (3.5) and (3.6) we can bound the total blocking probability as:

$$
B = \sum_{x_{s1}, x_{s2}} \pi_s(x_{s1}, x_{s2}) B_e(x_{s1}, x_{s2}) \leq \sum_{x_{s1}, x_{s2}} \tilde{\pi}_s(x_{s1}, x_{s2}) B_e(x_{s1}, x_{s2}) \tag{3.9}
$$

The tightness of the upper bound depends on the difference between the maximum and minimum blocking probability $B_e(x_{s1}, x_{s2})$. As we shall see in the next section, in our simulations the upper bound can be as much as twice as the simulated blocking probability. However, this occurs only for high load values, where this difference is bigger. As the load decreases, the upper bound gets tighter.

3.3 Simulation results

3.3.1 Flow-level (or fluid) simulations

Several approximations were made in the SGP analysis. Notably, we made suppositions on the streaming flows arrival rate at each server and used OBP to calculate the blocking probability for elastic flows. To verify that the fluid system (i.e. without considering TCP dynamics) is still very well represented, we developed a flow-level simulator which implements a Cross-Protect router.

A comparison between simulation and analytical results for the two server case can be seen in Fig. 3.2. In the simulation, the capacity of a link is 1 and the other 2. The total traffic is a mix of elastic traffic whose workload follows an exponential distribution with mean 0.5 and streaming traffic whose rate is 0.05. The streaming part of traffic represents 20% of the total volume of traffic. In the graphs, there are two different analytical plots. One drawn using the product form bound $\tilde{\pi}_s(x_{s1}, x_{s2})$ (Eq. (3.6)) and the other one using the numerical solution of the global balance equations. It can be seen how the product form estimation is an acceptable upper bound for the real blocking probability. It is also clear that the numerical solution is a very tight approximation, which means that the proposed Markov chain accurately models the system.

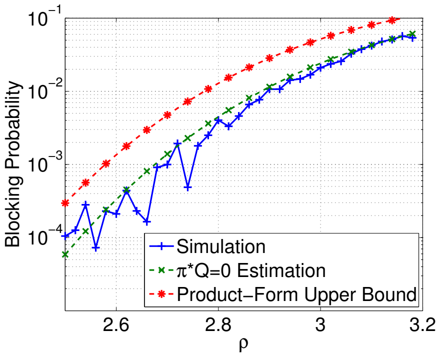


Figure 3.2: Blocking probability estimation and fluid simulation results for asymmetric server case

3.3.2 Packet-level simulations

As in Sec. 2.3.3, we conducted several packet level simulations using ns-2. A comparison between simulation and analytical results for the two server case can be seen in Fig. 3.3. The considered scenario is the same as the previous one, but with a different elastic load distribution. The figure features two different simulation plots: one is relative to an exponential elastic flow size distribution, the other plot corresponds to a Pareto distribution. Moreover, path capacities are 1 Mbps and 2 Mbps and the x-axis represents the total demand with respect to 1 Mbps (i.e. $\rho = d/1\,\text{Mbps}$). Results show that the scheme is a little bit sensitive to the distribution, all the more as load increases. As the load decreases, blocking probability tends to be the same for both. The graph also shows that the upper-bound is, in this case, actually an approximation. As we saw, in the flow-level simulations it was a strict upper-bound. This increase in the expected blocking probability is then due to the dynamic behavior of TCP.

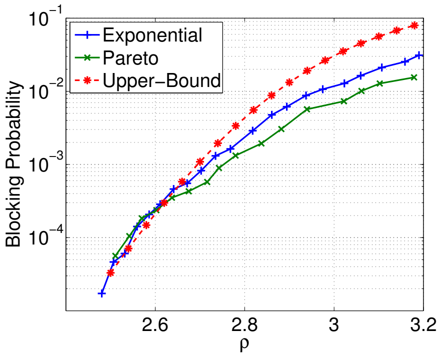


Figure 3.3: Blocking probability estimation and simulation results for SGP

We now compare the proposed scheme with other possible load balancing techniques. As we will see further in the thesis, most dynamic load-balancing schemes will converge to a situation where each incoming flow is assigned to a random path, where the probability to be sent to the i–th queue is $C_i/(\sum C_j)$. In the symmetric case, there is no great advantage in using the proposed scheme. But in the asymmetric case there will be a great gain in a load balancing scheme that takes into account the current state of each path. In Fig. 3.4 we present a comparison between the two load balancing schemes in terms of their flow blocking probability. In the same example as before, it can be seen that the gain in using the proposed load balancing scheme is considerable, especially when the system is not very loaded, which should be the operation point.

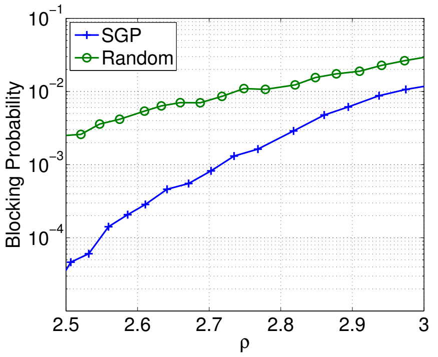


Figure 3.4: Blocking probability using SGP and random load balancing

A reasonable question is whether this gain in blocking probability would have a negative effect on TCP throughput. Since more flows are admitted in SGP, the share of bandwidth each flow gets is less. In Fig. 3.5 the mean throughput of TCP flows is shown as a function of the load for both load balancing schemes in the same scenario as before. It can be seen that the gain in throughput is not big enough to compensate for a blocking probability which can be two orders of magnitude bigger.

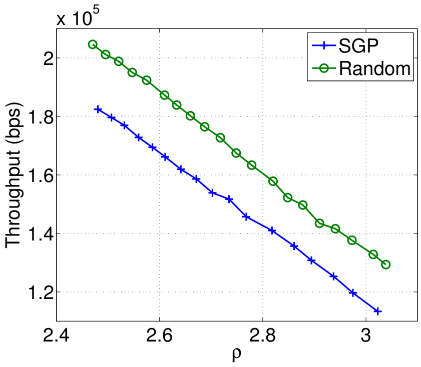


Figure 3.5: Mean TCP throughput for SGP and random load balancing

3.4 Conclusions

Bringing Ethernet into the Metropolitan Area Network introduces a lot of advantages to both the service provider and the customer (corporate and residential). However the lack of mature TE solutions is seriously delaying the emergence of a "carrier class Ethernet" network. In this first part of the thesis, we have addressed this critical aspect of Metro Ethernet architectures.

Drawing on the work of Roberts et al., we have discussed the application of the flow-aware networking paradigm, based on Cross-protect mechanisms, in the context of connection-oriented networks, such as MPLS. We believe this a simple and efficient alternative to the Diffserv-TE solution. Simplicity flows from the ability of differentiating streaming and elastic flows associated to a virtual path in an implicit manner. Hence, dispensing with the need of marking packets as in Diffserv. Efficiency, on the other hand, is the result of a better control of QoS, which is realized at flow-level rather than aggregate level (remember aggregates are difficult to characterize). Indeed, minimum QoS guarantees to streaming and elastic flows are enforced by means of admission control.

Note that the implementation of flow-aware networking is particularly interesting in a connection-oriented network, compared to a pure IP network, as Cross-protect mechanisms need only be implemented in edge routers. In fact, as mentioned before, our scheme is not restricted to MetroEthernet only, but is applicable to any connection-oriented environment in which a certain capacity can be guaranteed to tunnels. We concentrated on MetroEthernet in particular due to its lack of efficient TE schemes. In other architectures where such schemes do exist, our proposition can be seen as a complement or a substitute.

We have evaluated the performance of the proposed QoS architecture by means of analysis and simulations. In particular, we have derived a closed-form formula for estimating the blocking probability of a Cross-Protect router, as a function of the expected elastic and streaming loads.

In order to further improve network performance and resilience, we have proposed *Simplified Greedy Policy* (SGP), a simple dynamic load balancing scheme which is rather straightforward to implement in a Cross-protect router, as it exploits fair-rate measurements used by the admission control. Our results highlight the gain achieved by SGP over the static random load assignment strategy. In the asymmetric scenario (i.e., parallel LSPs with different capacities), for instance, the static reference strategy may yield a blocking probability that is orders of magnitude higher than that obtained with SGP.

In order to evaluate the performance of a Cross-protect router implementing SGP for balancing the load induced by both elastic and streaming traffic, we have developed a simple analytical fluid model to derive an approximate expression for the blocking probability. The analytical model was verified by means of fluid simulations and packet-level simulations. Fluid simulation results nicely fit the analytical results. The comparative evaluation with packet-level simulation shows that the derived blocking probability formula constitutes a reasonable approximation, though it does no longer constitute an upper bound as in the fluid setting.

Part II

Shared Resources

Chapter 4

Introduction and State of the Art

4.1 Introduction

In this second part of the thesis we shall discuss possible TE schemes for a network in which resources are shared among paths. This is actually the case for most operating networks, and as such the ideas proposed here may be applied to a broad group of networks. However, we will still assume that each OD pair is connected by several paths, of which they may obtain certain information (e.g. maximum link utilization). This means that the mechanisms we shall propose are designed for *intradomain* TE. The *interdomain* setting poses several problems (e.g. allowing multipath routing in internet, obtaining reliable path information in a multiprovider context, etc), and as such is left for future work. We will also assume that traffic may be arbitrarily distributed along paths.

As opposed to the first part of the thesis, we will not assume any mechanism in particular to manage the paths. In particular, the use of Cross-Protect in this context raises several questions. We may for instance, and as before, implement Cross-Protect at the ingress node of the path. The problem is then what to do at core nodes, where different paths share a link and the performance enforced at the ingress is not respected. If the core link capacities are such that we could assume a large overprovisionement, we would fall back to the case considered in the previous part, where implementation of Cross-Protect in the core was not necessary. If such assumption is unrealistic, we could also implement Cross-Protect in the core nodes (i.e. classify traffic, schedule and perform admission control). However, this would mean that we are duplicating tasks in the network. It could happen that the decision taken for a flow is not consistent throughout the routers it traverses. For instance, an admitted flow may be blocked downstream, or a flow that has been classified as elastic may be reclassified as streaming. Regarding classification, it should be noted that Cross-Protect performs its implicit classification based on the assumption that elastic flows will reach their fair rate. Unless the particular link is the bottleneck of the path, this will not be the case. This means for instance that all flows for which a link is not the bottleneck will be transmitted with priority, even when they are elastic but are bottlenecked elsewhere in the network.

Due to the above issues, we will not assume Cross-Protect at any level of the network, and will propose mechanisms that do not use measurements of the FR or PL, but more general ones, such as link load or queue size. Another very important assumption is that of the existence of the so-called Traffic Matrix (TM). If every node in the network is ordered by an index, this matrix contains in its ij–th entry the mean traffic demand from node i destined to node j. We will assume that traffic in the network is relatively well represented by this TM, and that the values it contains do not depend on the current network condition. This means that we will not consider congestion control in the problem. Furthermore, we will also assume that the TM does not change with routing. This could not be the true if we were for instance optimizing IGP weights, which have an influence on the egress node for BGP traffic (this is known as hot-potato [44]), and thus in the TM. In this thesis we have considered an MPLS network, and as a consequence we do not have this problem.

Before presenting our proposals, we will discuss and present several prior works that revolve around Dynamic Load-Balancing (DLB). Actually, DLB can be seen as a particular case of Multi-Path routing. Many papers fall under this category, and we will highlight only those closely related to our work. In particular, we will first introduce, among other static routing schemes, Robust Routing, a technique that many consider as the alternative to DLB. We will then present the more important proposals in the area of DLB. Finally, we will discuss the multi-path congestion control problem, which has many aspects in common with DLB.

4.2 Static Routing for Dynamic Traffic

4.2.1 Robust Routing

As mentioned in Ch. 1, due mainly to service convergence, the traffic injected to the network is increasingly complex and difficult to predict. *Robust Routing* (RR) [8, 9] deals precisely with this problematic aspect of traffic from a routing perspective. More exactly, the objective in RR is to find a single static routing configuration that fulfills a certain criteria for a wide range of TMs, generally the one that minimizes the maximum link utilization over all TMs.

Traditional algorithms rely on a small group of expected TMs (representative traffic demands from past observations) or estimated TMs to compute optimal and reliable routing configurations. An extreme case is presented in [45], where routing is optimized for a single estimated TM and it is then applied during long time-scales (e.g. daily routing). Traffic uncertainty is characterized by multiple TMs in [46] (e.g. set of TMs from previous day, same day of previous week, etc.), and different ways to find optimal routes for the set are presented.

Given the dynamic nature of current traffic demands, relying on a single or a small set of expected TMs is no longer suitable, and Robust Routing techniques should be used instead . In [8], the authors capture traffic variations by introducing a polyhedral set of demands. This specific set enables a fast and relatively simple computation of the optimal routing configuration. The authors of [47] apply this technique to compute a robust MPLS routing configuration without depending on TM estimation, and discuss corresponding methods for robust OSPF optimization. Oblivious Routing [9] also defines linear algorithms to optimize worst-case performance for different sizes of traffic uncertainty sets, aiming to handle dynamic changes.

It should be clear that, since a single traffic distribution is used for all TMs, resources will be wasted for any specific TM, and that shrinking the uncertainty set results in improved performance. For instance, [48] analyzes the use of RR through a combination of traffic matrix estimation and its corresponding estimation error bounds, in order to shrink the uncertainty set. The authors of [23] propose a multi-hour approach, in which the static routing configuration changes over the day, thus reducing the uncertainty set. In any case, this shrinking should be carefully done, because if the selected set is too small and the network faces an unforeseen TM, the resulting performance is unpredictable. In this sense, [23] proposes a scheme that is both robust and reactive. In addition to the multi-hour approach we just described, it studies the possibility of detecting anomalous or unexpected traffic demands, and reacting appropriately by reconfiguring the traffic distribution. Another interesting reference is [49], in which the authors present COPE, an approach to deal with this tradeoff in the size of the uncertainty set. COPE optimizes routing for predicted demands and bounds worst-case performance to ensure acceptable efficiency under unexpected traffic events. In any case, an inherent problem of RR is that optimization under uncertainty is more complex than classical optimization, which forces the use of simpler optimization criteria. More details on RR will be discussed in Sec. 8.5 where we present a comparison between RR and DLB.

4.2.2 The Hose Model

A problem related with RR is that of dimensioning and routing in the context of the so-called *Hose Model* [50]. In traditional VPN services the *Pipe Model* is used, where the customer specifies the maximum amount of traffic nodes will send to each other (in a way, it specifies its worst-case TM). Finding these values could be very complicated for the customer, who does not necessarily have the metrology infrastructure to determine them. Moreover, as the number of nodes in the VPN increases, estimating this TM could be problematic even for the NSP. In the hose model, the customer specifies instead the *total* traffic each node will receive or send, independently of its origin or destination. The advantages of this model are firstly its simplicity for the customer, but also its potential reduction in the size of the access links due to traffic multiplexing.

The main difficulty in this model is managing the resources required to support all possible TMs. For instance, the original paper of Duffield et al. [50] presents a measurement based resizing mechanism. Implementation of such dynamic reservation scheme is, to say the least, very difficult. Instead, several papers study the problem of designing minimum-cost routing schemes (generally based on trees) given the bounds specified by the customer (e.g. [51, 52, 53, 54]). In terms of efficiency, the total bandwidth required by these solutions is roughly 50% more than if there was a single TM [55, 56].

Similarly to RR, the information we have of the TM in this model is a set in which it is assumed to be. In addition to being more general in the considered uncertainty set, there are two important aspects that distinguish RR from the hose model problem. Firstly, one of the inputs to the former is the links capacity, which is one of the outputs of the latter (or similarly, the required bandwidth reservation). Secondly, RR strives to optimize a certain objective function (e.g. minimize the maximum link utilization), while the objective in the algorithms presented in this section is simply to design a network that can support all possible TMs.

4.2.3 Valiant Load-Balancing

The last static routing scheme we will discuss is *Valiant Load-Balancing* (VLB) [57] (also known as *Two-Phase Routing* [58] or *Randomized Load-Balancing* [59]). Similarly to the hose model, the only available information of the traffic is the maximum amount each node may receive (D_i) or send (U_i). However, the routing scheme considered, based on the ideas by Valiant for processor interconnection networks [60], is very specific.

The mechanism is simple: traffic entering the network is balanced among all nodes that in turn send the packets to their final destination (the amount balanced to each node depends on the values of D_i and U_i). The idea is to "uniformize" the demands so that dimensioning to support all possible TMs is very simple. Moreover, the network requires a capacity that, in the worst case, is twice the one obtained if we knew the exact TM [61]. Finally, due to the huge path diversity, the resiliency of the network is excellent. However, it also presents certain disadvantages. If there are N nodes in the network, $N-1$ paths have to be established for each OD pair (thus decreasing the scalability of VLB), and the propagation delay will increase with respect to the shortest path.

It is interesting to note the relation between VLB and the solutions of the hose model. As mentioned above, these solutions are generally tree type, arguably because dimensioning is easier in a tree than in an arbitrary graph. For instance, a very elegant algorithm to find this least-cost tree is as follows [51, 54]. For every node v, calculate the shortest path tree to all VPN nodes. In every path from root node v to VPN node n_i install U_i and D_i in each corresponding direction (the effect of installing capacities is cumulative). The optimum tree is the one with the minimum total cost of all these capacitated trees (note v_{opt} the root of this tree). The authors of [51] proved that the cost needed to route under a tree or under a hub with the same root (all traffic is first sent to an intermediate root node who in turn sends traffic to the final destination) is the same only when the root is v_{opt}. In the rest of the cases, the total cost for the hub is always bigger.

VLB can be seen as a combination of hub routing, where all nodes in the network act as the hub root for a portion of traffic. This means that, if the hub trees were constructed under the shortest-path criteria, the resulting capacity requirements would be a convex combination of the capacity of hubs rooted at all the nodes of the network[1]. Based on these observations, and trying to construct a solution between the optimum (but with very little resiliency) tree and the generally more expensive and with bigger delays (but very resilient) VLB, the authors of [62] proposed *Selective Load-Balancing*. The idea is to balance traffic only to a subset of nodes, instead of all as in VLB. That is to say, superpose the M less expensive hubs. This will result in a network that is cheaper than VLB and that generally has a smaller total delay (since less expensive trees tend to have their roots at the center of the network).

4.3 Dynamic Load-Balancing

In order to avoid choosing a representative set of TMs, dynamic load-balancing (DLB) [4, 5, 6] strives to optimize the traffic distribution for the *current* traffic demand. This way it completely avoids the performance-size tradeoff we mentioned above, obtaining the best performance from the available resources. Actually, DLB performs the optimization for the current network situation that, in addition to the TM, it also includes possible link or node failures. This makes DLB more general in the considered scenarios than the mechanisms we discussed above. Moreover, given the network resources, and as long as the capacity constraints are enforced, the set of supported TMs is *at least* the one supported by the static mechanisms.

Obviously, all these advantages come with a price. Due to the relatively short time-scale (in the order of some seconds), paths are established *a priori* and the only possible adaptation is changing the amount of traffic sent along each of them (i.e. establishing and tearing down paths are not possible). Moreover, this time-scale also requires that the algorithm that controls these adaptations is ran on each ingress router. The design of these distributed algorithms is probably the most challenging aspect of DLB. It is precisely due to its dynamic and distributed nature that the deployment of DLB has been almost null[2]. Network operators are reluctant to use it mainly because they are afraid of a possible oscillatory behavior of the load-balancing algorithm, even when recent works indicate the contrary through simulations and theory [4, 5, 6].

The three most notable proposals in this area are MATE [4], TeXCP [5] and REPLEX [6]. The first difference between them is the objective function. MATE defines a convex increasing link function $f_l(\rho_l)$ (where ρ_l is the load on link l), and its objective

[1] This explains the results obtained in [61] indicating that the optimum cost in VLB is attained by a star topology (where the center is v_{opt})

[2] In fact, practically the only load-balancing present in the internet today is ECMP (Equal Cost Multi-Path) , where traffic is evenly split between paths with the same routing cost.

is to minimize the sum over all links of $f_l(\rho_l)$. The rationale is that this function represents the congestion on the link, and that DLB should strive to minimize the total congestion on the network. Convexity is intuitively justified by the fact that at higher loads, an increase in load generates more congestion than at lower loads (although, as we shall discuss further on, this is actually a mathematical requirement). This objective function has become very popular, to the point that some authors define TE as the procedure through which the network operator minimizes $\sum_l f_l(\rho_l)$ [63]. On the other hand, the objective for both TeXCP and REPLEX is to minimize the maximum link utilization in the network, like in RR. The justification for this objective function is the same as in RR, namely that a link with a u_l close to one is operating near its capacity, and in order to be able to support sudden increases in traffic and link/node failures, it is preferable to keep the links utilization relatively low.

In addition to their objective function, the other important difference among these three proposals lies in their distributed load-balancing algorithm. MATE uses an algorithm based on the classic gradient projection algorithm [15]. That is to say, at each iteration the amount of traffic sent along path P is updated as $d_P(t+1) = [d_P(t) - \gamma\phi_P(t)]^+$, where the path cost ϕ_P is equal to $\sum_{l\in P} f'_l(\rho_l)$. The most important drawback of this algorithm is its convergence speed [5]. Moreover, for convergence to be guaranteed, the stepsize γ has to fulfill a condition that depends on $f_l(\rho_l)$, which is not necessarily known in advance. The algorithm used in TeXCP was specifically designed for its objective function, and it may be roughly described as increasing the amount of traffic sent along the path with the lowest maximum utilization. Regarding REPLEX, the load-balancing algorithm is based on the adaptive sampling methods presented in [64]. We will discuss REPLEX with more detail in Sec. 7.4, but it is interesting no note that although the original algorithm appearing in [64] was designed for the objective function of MATE, the authors of [6] use it to minimize the maximum link utilization. Moreover, they also discuss some interesting ideas to adapt these "tunnel-based" schemes to the context of all-IP routing.

4.4 Multi-Path Congestion Control

As illustrated in Fig. 4.1, if congestion control deals with flows from sender to receiver, load-balancing deals with flows from ingress to egress nodes. Based on this analogy we will develop in the following chapter a load-balancing mechanism that draws on the ideas of congestion control. Let us then recall the congestion control problem in its optimization perspective.

Assume that the network is used by S OD pairs, indexed by $s = 1, \ldots, S$. Each of these OD pairs may use any path from a set $\mathcal{P}_s$, where each of its elements will be noted as P_{si} with $i = 1, \ldots, n_s$. If the traffic generated by each OD pair is constituted of a fixed number N_s of TCP flows, with $N_{P_{si}}$ flows in path P_{si}, the congestion control

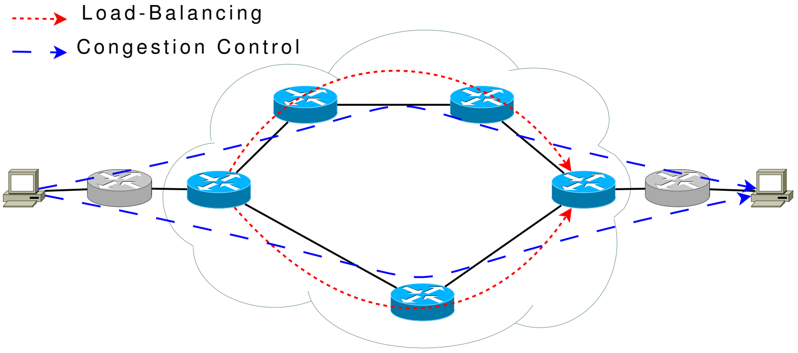


Figure 4.1: An illustration of a difference between Load-Balancing and Congestion Control

problem can be written as follows [65]:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{s=1}^{S} \sum_{i=1}^{n_s} N_{P_{si}} U_{P_{si}} \left(\frac{x_{P_{si}}}{N_{P_{si}}} \right) \\ \text{s.t.} \quad & \sum_{s} \sum_{i:l \in P_{si}} x_{P_{si}} \leq c_l \end{aligned} \tag{4.1}$$

where $x_{P_{si}}$ is the total rate obtained by the $N_{P_{si}}$ flows on path P_{si} (we have assumed that this rate is identically distributed among the $N_{P_{si}}$ flows) and $U_{P_{si}}(x)$ is a non-decreasing, concave and continuous function. A typical example of $U_{P_{si}}(x)$ is the utility function that leads to the so-called α-fairness [66]:

$$U_{P_{si}}(x) = \begin{cases} (1-\alpha)^{-1} x^{1-\alpha}, & \alpha \neq 1 \\ \log x, & \alpha = 1 \end{cases}$$

The parameter α sets the fairness of the optimum. For $\alpha = 0$ it maximizes the weighted sum of $x_{P_{si}}/N_{P_{si}}$, $\alpha = 1$ results in proportional fairness [67] and for $\alpha \to \infty$ it results in max-min fairness [16].

Possible adaptations of TCP to the multi-path case (MP-TCP) have been extensively studied. The idea is that each flow may use all the paths in $\mathcal{P}_s$, thus instead of associating each end-user to a path, we associate them with an OD pair. Moreover, the utility each user perceives is now a function of the total rate it obtains from all its paths. This results in the following problem:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{s=1}^{S} N_s U_s \left(\frac{\sum_{i=1}^{n_s} x_{P_{si}}}{N_s} \right) \\ \text{s.t.} \quad & \sum_{s} \sum_{i:l \in P_{si}} x_{P_{si}} \leq c_l \end{aligned} \tag{4.2}$$

The objective in MP-TCP is the same as in load-balancing: use the numerous paths between any two points in a network to increase resiliency and performance. As mentioned above, the difference lies in that for load-balancing the network operator exploits this multitude of paths, and in congestion control it is the end-user.

Several mechanisms that solve (4.2) have been proposed. For instance, in [68, 69, 70] the user is responsible of calculating its total rate and how much it should send along each path. This kind of solution is very problematic since the network operator should provide the end-users with information of the network topology. To avoid this complication, in the mechanism proposed in [71] the user only calculates the total sending rate, and the routers distribute traffic among paths. However, it is not clear what happens when only a fraction of the end-hosts implement this mechanism and the rest still uses legacy TCP.

A different but related problem is a user downloading the same file from different sites or hosts (as in Bittorrent). Currently, greedy policies are used where users change a path only if they obtain a better performance on the new one. The authors of [72] show that if current TCP flavors are used in such schemes, the resulting allocation can be both unfair and inefficient, and that a mechanism similar to MP-TCP should be used instead.

Finally, the works presented in [63, 73] could be considered as a combination of congestion control and load-balancing. In it, users have several paths to choose from, and the objective is to adapt the sending rates to maximize users utility minus a network cost. Although at first look it could seem similar to the TCP+AQM analysis [65], this cost function could be the same as in MATE, and the idea is that sources should also take into account the utilization of the links and leave a margin for future arrivals. We believe this is not the best objective. Congestion control should enable users to consume *all* their fair-share of the path. Saving a little bandwidth for future arrivals is, in our opinion, a waste of resources.

Chapter 5

Utility Maximization Load-Balancing

5.1 The New Objective Function

In this section we present a new objective function for DLB. As discussed in Sec. 4.3, previously proposed objective functions take a *network-centric* approach, in the sense that they minimize either congestion or utilization. Since the network operator is interested in the communication between the OD nodes, in this section we state the load-balancing problem in their terms (i.e. we take a *user-centric* approach). Assuming that the majority of traffic is elastic, and inspired on the ideas we presented in Sec. 4.4, the objective is to further maximize the flows' total utility by means of the multiple paths available. However, we do not propose to change the congestion control mechanism. Instead, this maximization is achieved by changing the amount of traffic routed along each path, thus changing the flows' obtained mean rate. Before actually presenting the new objective function, we will introduce the notation (valid here and in the following chapters), which we partially introduced in Sec. 4.4.

The network is defined as a graph $G = (V, E)$. In it there are a number of so-called *commodities* (i.e. OD pairs), indexed by $s = 1, \ldots, S$, specified in terms of the triplet o_s, q_s and d_s; i.e. origin node, destination node and a certain fixed demand of traffic from the former to the latter. Commodity s can use any path from set $\mathcal{P}_s$, where each of its elements (noted as P_{si} with $i = 1, \ldots, n_s$) connects o_s to q_s. All commodities can distribute their total demand arbitrarily along their paths. In particular, commodity s sends an amount $d_{P_{si}}$ of its traffic along path P_{si}, where $d_{P_{si}} \geq 0$ and $\sum d_{P_{si}} = d_s$. This distribution of traffic induces the demand vector $d = (d_{P_{si}})$. Given the demand vector, the total load on link l is then $\rho_l = \sum_s \sum_{i:l \in P_{si}} d_{P_{si}}$. It should be noted that in the framework described above the destination for a commodity is not necessarily a single node (e.g. two gateways to the internet may be seen as a single destination).

As in Sec. 4.4, if we assume that demands are constituted of N_s elastic flows, of which $N_{P_{si}}$ are in path P_{si}, the congestion control problem can be written as follows:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{s=1}^{S}\sum_{i=1}^{n_s} N_{P_{si}} U_{P_{si}}\left(\frac{x_{P_{si}}}{N_{P_{si}}}\right) \\ \text{s.t.} \quad & \sum_{s}\sum_{i:l\in P_{si}} x_{P_{si}} \leq c_l \end{aligned} \tag{5.1}$$

where $x_{P_{si}}$ is the total rate obtained by the $N_{P_{si}}$ flows on path P_{si} and $U_{P_{si}}(x)$ is a non-decreasing, concave and continuous function. Note that since we assumed that $x_{P_{si}}$ is identically distributed among the N_{si} flows, the argument of $U_{P_{si}}$ in (5.1) is the rate obtained by each flow.

The above problem optimizes in the obtained rate, considering the $N_{P_{si}}$'s (i.e. routing) as given. However, to improve performance and resiliency, we could jointly maximize in both N and x (adding the constraints $\sum N_{P_{si}} = N_s$ and $N_{P_{si}} \geq 0$). If we approximate N by a real vector (reasonable if, for instance, the number of flows is high) the problem in both N and x is still convex ($yU(x/y)$ is known as the *perspective* of $U(x)$, which is convex in both x and y as long as $U(x)$ is convex [74]), a fact that makes this possibility even more appealing.

To implement such maximization, a first idea is that end-users decide their path. However, as mentioned in Sec. 4.4, network operators are very reluctant to share routing information with the end-users for security reasons. Moreover, allowing end-hosts to choose their paths, or even making them aware that several possibilities exist, presents several technical difficulties in current Internet architectures. However, it is possible to exploit path diversity within a carrier network through (dynamic) load-balancing. We thus propose to keep the separation between end-to-end congestion control (maximization on x performed by end-users) and routing (maximizing on N performed by routers) but still try to solve (5.1). However, two important issues should be considered, which we shall now discuss.

The first obvious problem is that routers do not know $U_{P_{si}}(x)$, and even if they knew it, we as network operators may not "like" it[1]. For this reason, we shall use an arbitrary $U(x)$ function which we believe convenient to operate the network (in our simulations we used $U(x) = \log(x)$). The second issue has to do with time-scales. Congestion control acts at the RTT timescale (generally in the order of ms) while routing does it at the seconds or minutes timescale. This means that we cannot consider the instantaneous values of $x_{P_{si}}$, but its temporal mean. Moreover, TCP flows have a certain finite lifetime, thus N_s cannot be considered as static or given and a substitute should be used too. We will first introduce the dynamic traffic model we use for this context, and then we will discuss how to address these issues.

The model is very similar to the one used in Sec. 2.3.1. Each origin node o_s generates flows as a Poisson process of intensity λ_s. Each of these elastic flows consists

[1] For instance, we may consider that the bias against RTT of TCP [75] should not be re-enforced by routing.

of a random arbitrarily distributed workload with mean b_s. After this workload is transmitted, the flow disappears. Each flow is routed along path P_{si} with probability p_{si}, where it remains throughout its lifetime. Under these conditions, the demand of the corresponding commodity can be written as $d_s = \lambda_s b_s$, and the demand vector as $d = (d_s p_{si})$.

The above described system has been extensively studied in the past. The mean throughput obtained by flows traversing path P can be roughly approximated by the path's Available Bandwidth (ABW):

$$\mathbb{E}\left\{\frac{x_P}{N_P}\right\} \approx ABW_P = \min_{l\in P}\{ABW_l\} = \min_{l\in P}\{c_l - \rho_l\}$$

where $\mathbb{E}$ represents the temporal mean. A possible justification of the above approximation is the following. The authors of [13] proved that balanced fairness [76] may be used to approximate other fair sharing notions, such as proportional fairness or max-min fairness (cf. Sec. 4.4). Moreover, they have proved in [14] that a possible upper bound to the flows obtained rate under this model is precisely the path's available bandwidth. Similarly to [77], we shall then approximate $x_{P_{si}}/N_{P_{si}}$ by $ABW_{P_{si}}$, and use this approximation in (5.1).

The other variable we needed to approximate in (5.1) was $N_{P_{si}}$. Its natural substitute in this dynamic context is $d_{P_{si}}$, who plays the role of the amount of traffic using path P_{si}. The load-balancing approximation to (5.1) results finally in:

$$\begin{aligned} &\underset{d}{\text{maximize}} \sum_{s=1}^{S}\sum_{i=1}^{n_s} d_{P_{si}} U\left(\min_{l\in P_{si}}\{c_l - \rho_l\}\right) \qquad (5.2)\\ &\text{s.t.} \quad d_P \geq 0 \; \forall P \in \mathcal{P}_s \text{ and } \sum_{P\in\mathcal{P}_s} d_P = d_s \quad \forall s = 1,\ldots,S \end{aligned}$$

Note that (5.2) is the maximization of a continuous function over a compact set, meaning that there always exists at least one solution to the problem. However, it does not present the convexity property of (5.1). In the following section we shall derive certain characteristics of this optimum that will help us design distributed algorithms to achieve it.

When reading Sec. 4.4 and the problem we just stated, one may wonder why we did not consider an objective function similar to that of MP-TCP. That is to say, a utility maximization problem where the argument of the utility function is the mean rate obtained from all paths of the OD pair (i.e. $d_s U(\sum_i p_{si} ABW_{P_{si}})$ instead of $\sum_i d_{P_{si}} U(ABW_{P_{si}})$). After all, the authors of [72] proved that MP-TCP obtains a better performance than the solution of (5.1). In fact, we have also studied possible adaptations of the MP-TCP problem to the load-balancing time-scale, as we just did. The first problem we encountered is that the resulting problem is not solvable exactly, and several approximations have to be made. Secondly, the simulations we performed indicate that the performance obtained by both problems are very similar. Because of this, we have considered it somewhat redundant to include both versions of the problem in the thesis, and the interested reader may consult reference [78] for more details.

5.2 Characterization of the Optimum

To characterize the optimum of (5.2), we will begin by transforming it into a minimization problem. After some elemental transformations (which mainly take into account that $U(x)$ is non-decreasing) we may obtain the following equivalent problem:

$$\underset{d}{\text{minimize}} \sum_{s=1}^{S} \sum_{i=1}^{n_s} d_{P_{si}} \max_{l \in P_{si}} \{-U(c_l - \rho_l)\}$$

We now introduce the auxiliary variable $t_{P_{si}}$, which at optimality will be equal to $\max\limits_{l \in P_{si}} \{-U(c_l - \rho_l)\}$, resulting in the following problem (the restrictions have been shortened for the sake of clarity):

$$\begin{aligned} \underset{d,t}{\text{minimize}} \; & \sum_{s=1}^{S} \sum_{i=1}^{n_s} d_{P_{si}} t_{P_{si}} \\ \text{s.t.} \quad d_{P_{si}} \geq 0 \quad & \sum_{i=1}^{n_s} d_{P_{si}} = d_s \\ t_{P_{si}} \geq -U(c_l - \rho_l) \, & \forall s, i \;\; \forall l : l \in P_{si} \end{aligned} \tag{5.3}$$

Necessary conditions of a local minimum can be obtained from the Karush-Kuhn-Tucker (KKT) conditions [15]. Let us first write the Lagrangian function associated to problem (5.3):

$$\begin{aligned} L(d, t, \nu, \lambda, \theta) = & \sum_{s=1}^{S} \sum_{i=1}^{n_s} d_{P_{si}} t_{P_{si}} + \sum_{s=1}^{S} \nu_s \left(\sum_{i=1}^{n_s} d_{P_{si}} - d_s \right) \\ & - \sum_{s=1}^{S} \sum_{i=1}^{n_s} \lambda_{P_{si}} d_{P_{si}} - \sum_{s=1}^{S} \sum_{i=1}^{n_s} \sum_{l : l \in P_{si}} \theta_{P_{si} l} \left(t_{P_{si}} + U(c_l - \rho_l) \right) \end{aligned}$$

Let d^* be a local optimum of problem (5.3). The KKT conditions state that there exist unique Lagrange multiplier vectors ν^*, $\lambda^* \geq 0$ and $\theta^* \geq 0$ such that:

$$t^*_{P_{si}} + \nu^*_s - \lambda^*_{P_{si}} + \sum_{l : l \in P_{si}} \widehat{\theta}_l = 0 \tag{5.4}$$

$$d^*_{P_{si}} - \sum_{l : l \in P_{si}} \theta^*_{P_{si} l} = 0 \tag{5.5}$$

$$\lambda^*_{P_{si}} d^*_{P_{si}} = 0 \tag{5.6}$$

$$\theta^*_{P_{si} l} (t^*_{P_{si}} + U(c_l - \rho^*_l)) = 0 \tag{5.7}$$

where

$$\widehat{\theta}_l = \sum_{s=1}^{S} \sum_{i : l \in P_{si}} \theta^*_{P_{si} l} U'(c_l - \rho^*_l) \tag{5.8}$$

Firstly, note that (5.7) indicates that $\theta^*_{P_{si}l}$ is zero for all links that are not the bottleneck of path P_{si} (i.e. $c_l - \rho_l > ABW_{P_{si}}$). If we assume that paths have only one bottleneck, there will be only one non-zero term in the sum of (5.5). In such case, $\theta_{P_{si}l}$ is simply $d_{P_{si}}$ for the bottleneck link of P_{si}, and 0 for the rest:

$$\theta^*_{P_{si}l} = \begin{cases} d^*_{P_{si}} & \text{if } l = \underset{l \in si}{\operatorname{argmin}}\{c_l - \rho^*_l\} \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

Similarly, (5.6) means that the associated Lagrange multiplier $\lambda^*_{P_{si}}$ is positive only for those paths that are not used at optimality ($d^*_{P_{si}} = 0$), else it is equal to zero. Substituting this in (5.4) results in:

$$\begin{aligned} \phi^*_{P_{si}} &:= t_{P_{si}} + \sum_{l:l \in P_{si}} \widehat{\theta}_l = \max_{l \in P_{si}} \{-U(c_l - \rho^*_l)\} + \sum_{l:l \in P_{si}} \widehat{\theta}_l = \\ &= -U\left(\min_{l \in P_{si}}\{c_l - \rho^*_l\}\right) + \sum_{l:l \in P_{si}} \widehat{\theta}_l = \begin{cases} -\nu^*_s & \text{if } d^*_{P_{si}} > 0. \\ -\nu^*_s + \lambda^*_{P_{si}} & \text{if } d^*_{P_{si}} = 0 \end{cases} \end{aligned} \tag{5.10}$$

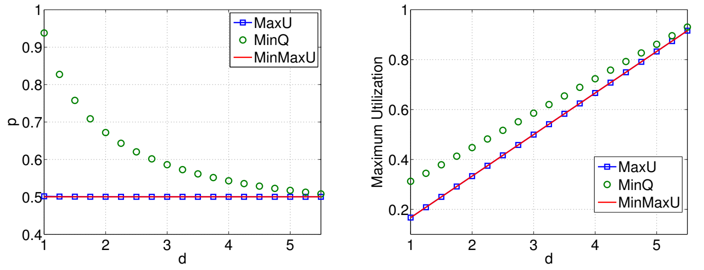
Then, for a given commodity s and by defining the path cost $\phi_{P_{si}}$ as above, all paths that are used at optimality have the same optimum $\phi^*_{P_{si}}$ ($-\nu_s$ does not change over paths), which is smaller or equal than that of paths that are not used (remember that $\lambda_{si} \geq 0$). In Ch. 7 we will present a distributed algorithm to achieve a demand vector that fulfills this condition.

Finally, note that the above condition is only necessary. This means that the solution of (5.2) verifies the condition. However, not all points that verify this condition are necessarily a minimum. For this last aspect, sufficient conditions may be derived that although difficult to prove in the general case, are verified in all the examples we evaluated (see Proposition 3.3.2 in [15]).

5.3 Illustrative Examples

So far, we have mentioned three objective functions that may be used for DLB: minimum congestion (cf. MATE in Sec. 4.3), minimum maximum link utilization (cf. TeXCP or REPLEX in Sec. 4.3, RR in Sec. 4.2.1) and maximum utility (the one we just presented). In this section we will overview three examples that illustrate the differences between these three objective functions and help us gain some insight and intuition. The simplicity of the considered examples allowed an easy offline calculation of the optimum for each objective function. In particular, in the case of minimum congestion we will assume that the link congestion function $f_l(\rho_l)$ is given by the M/M/1 mean queue size (i.e. $f_l(\rho_l) = \rho_l/(c_l - \rho_l)$). Finally, and for the sake of clarity of the exposition, we shall note each objective function as MinQ (as in Minimum Queue), MinMaxU (as in Minimum Maximum Utilization) and MaxU (as in Maximum Utility) respectively.

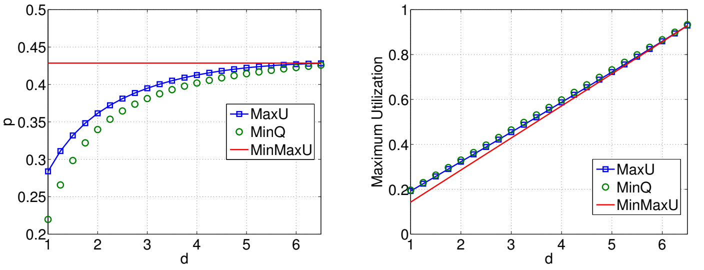
The first example we will consider is the simplest: a single commodity with two paths of the same capacity. However, one of the paths is constituted of two links, while the other of only one. In particular, link capacities will be 3.0. In Fig. 5.1(a) we can see p (the optimum traffic portion routed along the shortest path) as a function of d (the demand generated by the commodity) for the three objective functions. Naturally, for both MaxU and MinMaxU the optimum is always 0.5. However, in light loads MinQ uses the shortest path almost exclusively, and as load increases it will need to use the longest one to avoid congesting links. This behavior results in a significantly bigger maximum link utilization, as can be seen in Fig. 5.1(b).



(a) Optimum probability (b) Maximum link utilization

Figure 5.1: The single-commodity two-paths case (a longer path subcase).

The next example also has two paths, except that now both paths have the same length (one hop) and a capacity of 3.0 and 4.0 respectively. In Fig. 5.2(a) we show p (the optimum traffic portion routed along the narrowest path) as a function of d for the three different objective functions. Clearly p is always 3/7 for MinMaxU. However, p changes with d for both the other two objective functions this time. If the total demand d is small enough, the narrowest path is left unused since the obtained performance is inferior. As it can be seen in Fig. 5.2(b) the difference in the maximum utilization is not as significant as before.

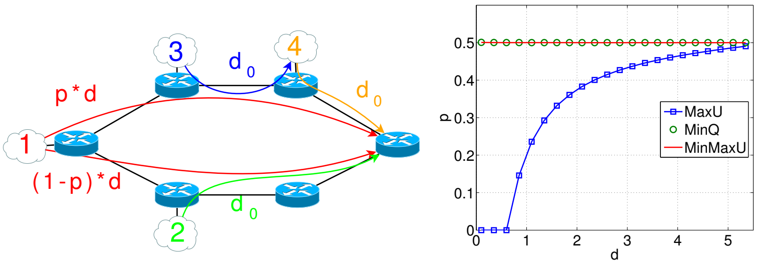


(a) Optimum probability (b) Maximum link utilization

Figure 5.2: The single-commodity two-paths case (a narrower path subcase).

The third example illustrates some fairness issues that are important to highlight. In Fig. 5.3(a) we can see the considered network. In it, all link capacities are equal and all commodities have the same demand d_0, except for commodity 1 that generates

d and is the only one to have more than one path to choose from. We will consider $c_l = 5.0 \forall l$, $d_0 = 2$ and we will study the optimum p as we vary d. It is relatively simple to verify that the optimum for MinQ and MinMaxU is $p = 0.5$ independently of d. On the other hand, MaxU enforces fairness at a path level. This means that commodity 1 takes into account that the upper path "disturbs" two other paths while the lower one disturbs only one, resulting in more traffic being sent along the latter (see Fig. 5.3(b)). So, while d is relatively small and the ABW is enough, commodity 1 uses only the lower path. If any of these conditions is not true, p will rapidly go to 0.5, but always privileging better conditions on the upper path.



(a) The network (b) Optimum p as a function of d

Figure 5.3: An example illustrating the fairness issue.

The above presented examples are relatively simple, but nevertheless give some intuition on the results that should be expected in more realistic and complicated scenarios. They will be helpful to interpret the results we will present in Sec. 8.2, where we shall compare the three objective functions over two real network topologies with real traffic demands.

Chapter 6

Minimum Congestion Load-Balancing: Learning the Cost Function

6.1 Introduction

In this section we will interest ourselves to the objective function used in MATE [4] (and probably first introduced in [16]) that, as we mentioned, is arguably the most popular one in TE. Let us recall that in this case we define a certain convex, increasing and continuous function $f_l(\rho_l)$, and the objective is to minimize $\sum_l f_l(\rho_l)$. The idea is that this function should measure the congestion on the link, for which the mean queue size is generally used. This choice is justified by two aspects. Firstly, its algebra is relatively simple since, as we shall discuss further on, the aggregation over all links of this function as a sum is only natural. Secondly, it is a very versatile indicator. A big mean queue size means more delay and jitter for streaming traffic. Moreover, a link with an important queue size is traversed by several bottlenecked flows, meaning that elastic traffic may obtain better throughput in other, less loaded, links. The mean queue size then gives a numerical value to the congestion on the link.

However, most DLB schemes require an analytical expression of this queue size, for which classic and oversimplistic models (e.g. *M/M/1*) are used [4]. It should be clear that optimizing with such a simple model results in an actual total congestion ($\sum_l f_l(\rho_l)$) that is significantly bigger than the optimum. We will now present and study a framework to avoid this arbitrary (and thus probably sub-optimal) choice of the queue size function. Except for some natural hypothesis on its shape (e.g. monotonicity) we will only assume that the mean queue size on link l is of the form $f_l(\rho_l)$ (i.e. depends only on the mean load of the link). The actual $f_l(\rho_l)$ will be obtained (or learned) from past measurements. This way, the congestion function will reflect reality independently of any additional assumption on the traffic. Moreover, the obtained total congestion will be an excellent approximation of the actual minimum.

To achieve this we will present two different regression methods. The first one is a variation of the nonparametric regression method presented in [17], and finds the regression function that fits best the measurements. However, it presents scalability issues as the number of available measurements increases. We consider then the algorithm presented in [18]. This heuristic finds a parametric function that reasonably adjusts the measurements in a very short time, although its precision is not as good as the one obtained by the first method.

Before presenting the regression methods, we will derive a characterization of the optimum for this case, analogously to Sec. 5.2. This analysis will help us identify certain characteristics of $f_l(\rho_l)$ that are required in order to converge to the optimum in a distributed fashion. Moreover, the information we need to know of $f_l(\rho_l)$ to achieve this convergence will also result from this analysis.

6.2 Characterizing the Optimum

Let us add certain elements to the notation presented in Sec. 5.1 that are required in this and the following chapters. The presence of load ρ_l on link l induces a certain mean queueing delay given by the non-decreasing function $D_l(\rho_l)$. The total delay of path P is defined as $D_P = \sum_{l:l\in P} D_l(\rho_l)$. As the measure of the network total congestion we shall use the *mean end-to-end queueing delay* (or *mean total delay*, which is the term generally used) $D(d)$, defined as:

$$D(d) = \sum_{s=1}^{S} \sum_{P\in\mathcal{P}_s} d_P D_P = \sum_{l=1}^{L} D_l(\rho_l)\rho_l := \sum_{l=1}^{L} f_l(\rho_l)$$

that is to say, a weighted mean delay, where the weight for each path is how much traffic is sent through it, or in terms of the links, the weight of each link is how much traffic is traversing it. We prefer this weighted mean to a simple total delay because it reflects more precisely performance as perceived by traffic: two situations where the total delay is the same, but in one of them most of the traffic is traversing heavily delayed links, should not be considered as equivalent. Note that, by Little's law, $f_l(\rho_l) := D_l(\rho_l)\rho_l$ is proportional to the average number of bytes in the queue of link l. Based on this observation, and as already discussed before, we will use this last value as $f_l(\rho_l)$ which (like the mean load) is readily available in most routers[1].

We can now write the problem explicitly:

$$\begin{aligned} \underset{d}{\text{minimize}} \; & \sum_{l=1}^{L} f_l(\rho_l) \\ \text{s.t.} \quad & d_{P_{si}} \geq 0 \quad \sum_{P\in\mathcal{P}_s} d_P = d_s \end{aligned} \tag{6.1}$$

[1] For instance, Cisco defines the CISCO-CLASS-BASED-QOS-MIB which includes cbQosREDMeanQSize and cbQosQueueingCurrentQDepth. If QoS is not enabled, we may use CISCO-QUEUE-MIB, which includes cQStatsDepth

Note that no explicit constraint on ρ_l was made. This is assumed to be implicitly included in the mean queue size function. For instance, if there exists a capacity constraint, $f_l(\rho_l)$ should go to infinity (or a big value) as ρ_l reaches c_l.

To characterize the solution of (6.1), we will proceed as in Sec. 5.2. The analysis in this case is very similar, and as such we will directly present the result of the KKT conditions:

$$\phi^*_{P_{si}} = \sum_{l \in P_{si}} \phi_l(\rho^*_l) = \sum_{l \in P_{si}} f'_l(\rho^*_l) = \begin{cases} -\nu^*_s & \text{if } d^*_{P_{si}} > 0. \\ -\nu^*_s + \lambda^*_{P_{si}} & \text{if } d^*_{P_{si}} = 0 \end{cases} \tag{6.2}$$

where vectors $\lambda \geq 0$ and ν are the Lagrange multipliers of the inequality and equality constraints respectively. Note that this necessary condition is true only if $f_l(\rho_l)$ is continuously differentiable (i.e. its derivative is continuous). Just like before, all paths that are used at optimality have the same cost, which is actually the minimum cost among all paths of the corresponding commodity. However, this time the path cost definition is simpler, resulting in the addition of a certain link cost function $\phi_l(\rho_l)$, defined as the derivative of $f_l(\rho_l)$ (i.e. $\phi_l(\rho_l) = f'_l(\rho_l)$).

If we assume that $f_l(\rho_l)$ is convex on ρ_l, the above condition is necessary and sufficient. Moreover, in such case the optimum is unique. As mentioned in Sec. 4.3, $f_l(\rho_l)$ is almost always assumed convex in the literature. We will not proceed differently, and assume this characteristic too. In addition to guaranteeing the existence and uniqueness of the optimum, it will help us in the design of the distributed load-balancing algorithm we will discuss in Ch. 7.

6.3 Learning the Mean Queue Size Function

6.3.1 General Considerations

In the previous section we have shown that the solution of (6.1) is a demand vector in which, for every commodity, the path cost ϕ_P (equal to the sum of the derivative of the mean queue size $f_l(\rho_l)$) is the same for all used paths, and bigger for those paths that are not used. We now address the problem of obtaining a good estimation of $\phi_l(\rho_l) = f'_l(\rho_l)$ from previous measurements on queue size and load. Actually, since it is the observable quantity, we shall first estimate $f_l(\rho_l)$ and then simply derivate this estimation. For the sake of clearness of the presentation, and since the procedure is the same for every link, in this section we shall omit the subindex l.

Assume we have a set of N measurements $\{(\rho_1, Y_1)\ (\rho_2, Y_2) \dots (\rho_N, Y_N)\}$ (also called *training set*), and assume that the response variable Y (the measured mean queue size) is related to the covariate ρ (the mean load) by the following equation:

$$Y_i = f(\rho_i) + \epsilon_i \qquad i = 1, \dots, N \tag{6.3}$$

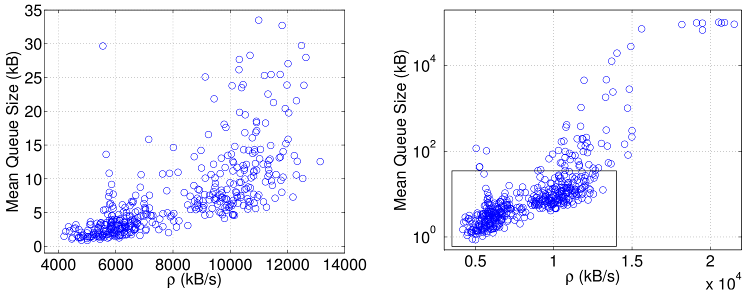
The measurement error ϵ_i is a random variable such that $\mathbb{E}\{\epsilon_i\} = 0$ and $\text{Var}\{\epsilon_i\} = \sigma_i < \infty$. The *Weighted Least Squares* (WLS) problem consists in finding the function $\widehat{f}(\rho)$ that minimizes the weighted sum of quadratic errors, assuming that $\widehat{f}(\rho)$ belongs to a given family of functions $\mathcal{F}$:

$$\min_f \sum_{i=1}^{N} w_i \left(Y_i - f(\rho_i)\right)^2 \quad \text{s.t. } f \in \mathcal{F} \tag{6.4}$$

where the weight $w_i \geq 0$ represents the relative importance of measurement point i with respect to the rest of the measurements in the training set.

There exists other criteria to choose the $\widehat{f}(\rho_l)$ that best fits the data. For instance, if the distribution of the errors ϵ_i was known, we could have used the Maximum Likelihood method [79]. However, the measurements are the result of a queueing process whose characteristic may change over time, meaning that such a prior knowledge is not available in our case. As a preview of the challenges posed by these measurements, we show in Fig. 6.1 a typical training set.

To obtain these measurements we injected a 72 hour long packet trace (obtained from [80]) to a simple queue emulator we developed[2]. The trace was measured on a trans-Pacific line with a capacity of 150 Mbps in march 2008 (we thus believe it to be a good representation of today's internet traffic). In the absence of information we assumed a relatively big buffer size of 100 MB. For each measurement, we took the mean load and queue size in a 60 second period. In Fig. 6.1 we show the measurements corresponding to two 4 hours period of the second day of the original trace: one in the early morning between 5 and 8, and the other in the afternoon between 16 and 19. Figure 6.1(a) shows a detail at relatively low loads, and in Fig. 6.1(b) we may see the whole training set in logarithmic scale (where we have highlighted the detailed area with a rectangle).



(a) Detail of the measurements at relatively low loads

(b) The complete set of measurements

Figure 6.1: An example of measurements

There are two characteristics of this training set that should be noted. Firstly, the variance of the measurements is not constant over ρ (a characteristic known as *het-*

[2] Although simple, it is based on the model described in [81], which shows an excellent accuracy.

eroscedasticity). Secondly, there are several and very important outliers. For instance, note how at $\rho \approx 6000\,\text{kB/s}$, due to a short but important burst of traffic, there are some mean queue size measurements that are almost two orders of magnitude bigger than their nearest neighbors. A good choice of the weights w_i will help us deal with these problems.

The following subsections present two different methods to estimate $\widehat{f}(\rho)$, and differ mainly on the assumed $\mathcal{F}$ in (6.4). We will start with the most general case, and based on the results obtained by it, derive the other method. How to assign the weights w_i will be left for the last subsection.

6.3.2 Convex Nonparametric Weighted Least Squares

In this subsection we present a method that keeps the assumptions on $\widehat{f}(\rho)$ to the minimum. Regarding its shape, we have only two necessary assumptions. Firstly, $\widehat{f}(\rho)$ should be non-decreasing, since more load may never mean less queue size. Secondly, $\widehat{f}(\rho)$ should be convex in order to guarantee the existence and uniqueness of the optimum demand vector, and that this optimum verifies (6.2).

We then consider $\mathcal{F}$ as the family of continuous, monotonous increasing and convex functions. We shall call Problem (6.4) with such $\mathcal{F}$ *Convex Nonparametric Weighted Least Squares* (CNWLS), a variation of the original unweighted Convex Nonparametric Least Squares (CNLS) [17]. The size of $\mathcal{F}$ makes this problem very difficult to solve in such general form. Consider instead the following alternative family of piecewise linear functions $\mathcal{G}_1(P)$:

$$\begin{aligned}\mathcal{G}_1(P) = \Big\{ g : \mathbb{R} \to \mathbb{R} \mid g(\rho) = \max_{i=1,\ldots,N} \alpha_i + \beta_i \rho;\\ \beta_i \geq 0 \;\; \forall i = 1, \ldots, N;\\ \alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \;\; \forall j, i = 1, \ldots, N \Big\}\end{aligned}$$

It is clear that $\mathcal{G}_1(P) \subseteq \mathcal{F}$ for all $P = \{\rho_i\}_{i=1,..,N}$. Moreover, consider the following theorem:

Theorem 1. *Let s_f^2 be the solution of problem* (6.4). *Let s_{g1}^2 also be the solution of problem* (6.4), *except that we substitute the constraint by $f \in \mathcal{G}_1(P)$. Then $s_f^2 = s_{g1}^2$.*

Proof. The proof is exactly the same as in the original CNLS [17]. Its demonstration relied on the fact that the optimization problem depends only on the value of $\widehat{f}(\rho)$ at the finite set of points $\{\rho_i\}$, which is also the case for CNWLS. □

This result allows us to transform the infinite dimensional problem (6.4) into the following standard finite dimensional Quadratic Programming (QP) problem:

$$\begin{aligned} &\min_{\epsilon,\alpha,\beta} \sum_{i=1}^{N} w_i \epsilon_i^2 \\ \text{s.t.} \quad & Y_i = \alpha_i + \beta_i \rho_i + \epsilon_i \; \forall i = 1, \ldots, N \\ & \alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \; \forall j, i = 1, \ldots, N \\ & \beta_i \geq 0 \; \forall i = 1, \ldots, N \end{aligned} \tag{6.5}$$

Although each observation has its own associated pair (α_i, β_i), as we shall illustrate later and already presented in the original CNLS, the actual number of significantly different values (which we shall note N^*) generally results in a small fraction of N. However, note that there are a total of $3N$ variables and $2N + N(N-1)$ constraints in (6.5). In particular, the second set of constraints, which are the key to enforce the convexity of $\widehat{f}(\rho)$, is quadratic in the number of observations, which will represent a problem as N increases.

6.3.3 Convex Piecewise-Linear Fitting

We have seen that problem (6.4) with the biggest possible family $\mathcal{F}$ may be solved by means of a QP problem. The resulting solution $\widehat{f}(\rho)$ is a piecewise function, where the partition of the linear segments is not fixed a priori; i.e. the number and location of the segments are endogenously determined to minimize the weighted squared residual. However, the resulting QP problem presents scalability issues as the number of observations increases. In this subsection we will try to solve this issue by fixing the number of segments to an arbitrary k, thus considering the following family of functions:

$$\begin{aligned} \mathcal{G}_2 = \Big\{ g : \mathbb{R} \rightarrow \mathbb{R} \mid g(\rho) = \max_{j=1,\ldots,k} \alpha_j + \beta_j \rho; \\ \beta_j \geq 0 \;\; \forall j = 1, \ldots, k \Big\} \end{aligned}$$

Substituting $\mathcal{F}$ by $\mathcal{G}_2$ in (6.4) results in the problem known as *Convex Piecewise-Linear Fitting* (CPLF). If k is bigger or equal than N^* (cf. previous subsection), then the optimum for CPLF and CNWLS will be the same. Considering that N^* is generally a small fraction of N, the possibility of solving the former instead of the latter seems interesting. Unfortunately, CPLF is not globally convex, meaning that, contrary to CNWLS, an exact solution cannot be found. The authors of [18] present an heuristic that approximately solves the unweighted version of the resulting problem, which may be easily adapted to solve the weighted one.

The algorithm is relatively simple, and it alternates between partitioning the measurements and performing a constrained ($\beta_j \geq 0$) linear WLS fitting to update the (α_j, β_j) pairs. Let $P_j^{(l)}$ for $j = 1 \ldots k$ be a partition of the measurements indices at iteration l, so that $\bigcup P_j^{(l)} = \{1 \dots N\}$ and $P_{j_1}^{(l)} \bigcap P_{j_2}^{(l)} = \emptyset$. The heuristic is described below.

Algorithm 1 Convex Piecewise-Linear Fitting (CPLF) Heuristic

```
Starting with an initial partition $P_1^{(0)} \dots P_k^{(0)}$
for $l = 0, \dots, l_{max}$ do
    Compute $(\alpha_j, \beta_j)^{(l+1)} = \underset{\beta_j \geq 0;\, \alpha_j}{\operatorname{argmin}} \sum_{i \in P_j^{(l)}} w_i \left(Y_i - \alpha_j - \rho_i \beta_j\right)^2 \quad \forall j = 1, \dots, k$
    $P_j^{(l+1)}$ contains $i$ if $j = \underset{s=1 \dots k}{\operatorname{argmax}} \{\alpha_s^{(l+1)} + \rho_i \beta_s^{(l+1)}\}$
    Finish if $P_j^{(l+1)}$ is equal to $P_j^{(l)}$ $\forall j = 1 \dots k$
end for
```

The idea is to try different random initial partitions $P_1^{(0)} \dots P_k^{(0)}$ and keep the best solution, which generally results in a relatively precise estimation. However, convergence of the algorithm is not guaranteed, and oscillations may occur. The only effective way of detecting this is by checking whether the maximum allowed number of iterations was reached. In such case, the obtained solution is of such poor quality, that a new repetition should be done instead. Moreover, during the iterations, certain partitions may be emptied. In such case, their (α_j, β_j) should be eliminated. This means that the input k actually indicates the *maximum* number of pairs in the solution. Finally, regarding complexity, the core of the iteration (solving a constrained linear WLS) is a much simpler problem than CNWLS. It is still a QP problem, but the number of variables and restrictions are now $3k$ and k respectively.

6.3.4 Choosing the Weights

In this subsection we discuss a possible way of choosing the values of w_i in (6.4). As we illustrated in Sec. 6.3.1, measurements present heteroscedasticity and important outliers. We will then set the weights so that (6.4) results in the *Least Absolute Deviations* problem (i.e. minimize the sum of the absolute, instead of the squared, errors), which is known to be more robust to these problems. The classic way of calculating such weights is to use the *Iteratively Reweighted Least Squares* algorithm [82] described below.

Algorithm 2 Iteratively Reweighted Least Squares Algorithm

```
Starting with the initial weights $w_i^{(0)} = 1$ $\forall i = 1 \dots N$
for $l = 0, \dots, l_{max}$ do
    Solve (6.4) with $w_i^{(l)}$ to compute $(\alpha_j, \beta_j)^{(l+1)}$, resulting in the regression function $\widehat{f}^{(l+1)}(\rho)$
    Update $w_i^{(l+1)}$ as the inverse of $\left|\widehat{f}^{(l+1)}(\rho_i) - Y_i\right|$
    Finish if convergence is reached
end for
```

To avoid numerical problems, as suggested in [82], if at any given iteration an error is less than 10^{-5} the corresponding weight should be set to 10^{5}. Anyway, note that the algorithm requires a solution of (6.4) at each iteration. This does not represent a problem for CPLF, although, as we mentioned before, the time required by CNWLS may be considerable, and repeatedly applying it may result in a prohibitive amount of time. In this case then, we shall proceed as follows. We perform an initial simple estimation $\widehat{f}^{(0)}(\rho_i)$, and calculate the corresponding weight as:

$$w_i = \frac{1}{\left|\widehat{f}^{(0)}(\rho_i) - Y_i\right|} \tag{6.6}$$

As the initial $\widehat{f}^{(0)}(\rho_i)$ we used the *m-nearest neighbors* algorithm (with $m = 10$), which simply estimates $f(\rho)$ as the median of the m measurements Y_i corresponding to the ρ_i's nearest to ρ. In this way, we seek to find a curve that fits the bulk of the data, and minimize the effect of outliers. Moreover, convergence is not guaranteed for the reweighting algorithm. We deal with this situation (which may be detected by checking if the total absolute error has increased from one iteration to the next) by falling back to these default weights.

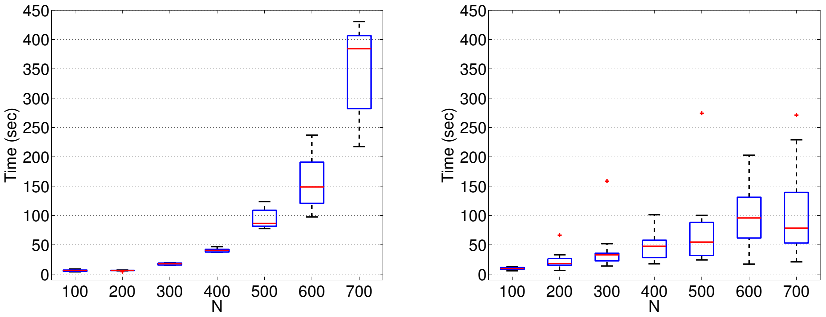
6.4 Comparison of the Regression Methods

In this section we will present a performance study of the two regression methods, so as to quantify some of their aspects that we have so far discussed based on intuition. In particular we are interested in the total time consumed by each method and the obtained precision. All calculations were performed in MATLAB (in particular, all QP solving was done using the optimization toolbox provided by MOSEK [83]) running on a laptop with a t5600 @1.83 GHz processor and 3 GB of RAM. The training sets were constructed with measurements resulting from the trace mentioned in Sec. 6.3.1 (we will note the whole set of measurements as (ρ, Y)).

6.4.1 Dependence on N

In this subsection we will analyze the performance of the algorithms as we change N (the size of the training set), in particular the computation time they require. To encompass as many operation points as possible, we separated (ρ, Y) into 5 equally sized subsets $(\rho, Y)_{m=1,..,5}$ so that $\max\{(\rho, Y)_{m_1}\} < \min\{(\rho, Y)_{m_2}\} \; \forall \, m_1 < m_2$. From each of these subsets we took $N/5$ random measurements to construct a training set, to which the two regression methods were applied. For each training set size N, the procedure was repeated 10 times. For CPLF, the maximum number of segments k was fixed at 10.

Figure 6.2 shows the boxplots of the results obtained by each method from $N = 100$ to $N = 700$. We may clearly appreciate that the computation time for CNWLS is quadratic on the number of available observations, while for CPLF it is roughly linear. This means that although at relatively low values of N the performance of the two methods may be considered as equivalent, as N increases CNWLS may become prohibitive. However, it should be noted that although in median the computation time of CPLF is small, there is a non-negligible probability that it may be very important. This is due to the oscillatory behavior of CPLF we discussed in Sec. 6.3.3. Even if we intended to try 10 random initial partitions (like in our case) we may end up actually doing many more.



(a) CNWLS

(b) CPLF

Figure 6.2: The time consumed by each method as we change N (the number of observations in the training set)

6.4.2 Dependence on k

In this subsection we will study the performance of CPLF as we change k (the maximum number of (α_j, β_j) pairs), and compare it with the optimum solution obtained by CNWLS. The procedure to construct the training set was the same as before, except that measurements were taken exclusively from the second day of the trace and we fixed N at 300.

As the precision indicator we will use the median of the absolute error obtained by applying the corresponding $\widehat{f}(\rho)$ to the rest of the measurements in the 24 hours period. We preferred the median to the mean because of the extreme outliers we already discussed. In Fig. 6.3 we can see the absolute errors obtained by the two methods in a typical training and validation set. Note how the extreme values for $10000 < \rho < 12000\,\text{kB/s}$ result in an average error that is very similar for both methods. However, the median for CNWLS is significantly smaller than for CPLF. It should be clear from the graph that CNWLS is the most precise of both methods, and the mean not reflecting this fact makes it an inadequate indicator.

A natural question that may arise from the above observations is why not minimize the median error in (6.4). First of all, such problem with the addition of shape restrictions is extremely difficult and escapes the scope of this thesis. Secondly, it is not

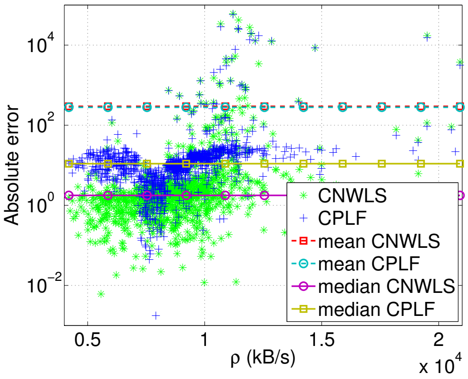
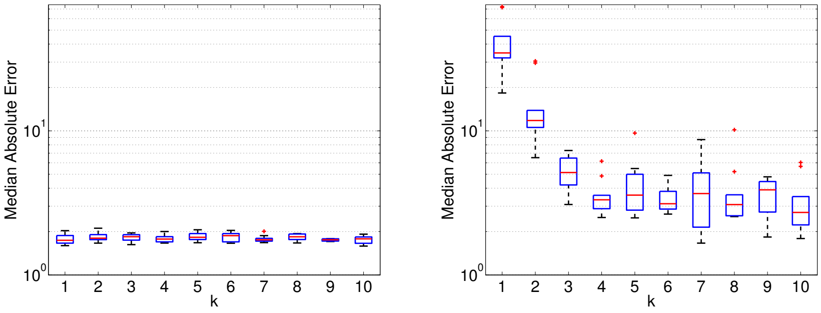


Figure 6.3: The absolute error for each measurement in the validation set for the two methods in an arbitrary example (k=2)

clear that *completely* ignoring extreme values, like the median error does, is the best solution. For instance, in Fig. 6.1 it could result in not using the measurements with $\rho_l > 15000\,\mathrm{kB/s}$.

The precision obtained by each method at different values of k is presented in Fig. 6.4. Although CNWLS is obviously not influenced by k, the graph shows its performance when applied over the same training set as CPLF. We may verify that CNWLS systematically obtains the best results. Although its weights are not adjusted optimally, the fact that it “finds” the best value of k (N^* in its case) for each particular training set, and that its corresponding optimization problem can be solved exactly, compensate this disadvantage. Note how for CPLF, after a rapid decrease, the error becomes constant for all k bigger than 4. This means that in general, setting k to a cautious value such as $k = 10$ (cautious in the sense that we are sure we are in the constant part of Fig. 6.4(b)) should be a reasonable choice.



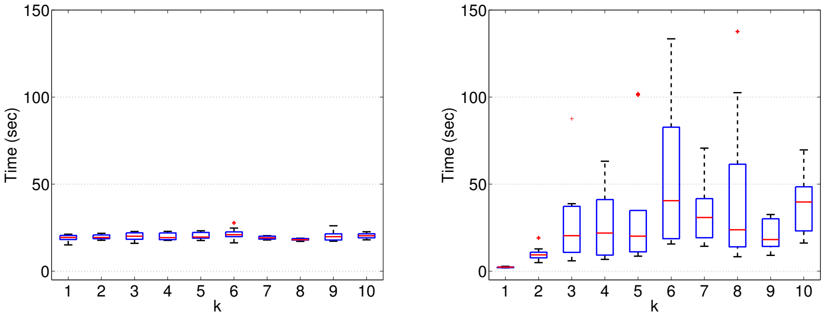
(a) CNWLS

(b) CPLF

Figure 6.4: Median absolute error obtained by each method as we change k (the maximum number of segments in CPLF)

We have already studied the time performance of each algorithm as we changed N.

We will now comment on its dependence on k. For the same cases as before, we have measured the total time consumed by each method. The results may be seen in Fig. 6.5. We have again included the results for CNWLS only as a reference. Surprisingly, the time consumed by CPLF is almost independent of k (except at its lower values). Notice that this behavior is similar to the one obtained in Fig. 6.4(b). As mentioned before, for CPLF k is only the *maximum* number of pairs. If during an iteration one of the partitions is emptied, it is discarded, and does not influence future computations any longer. This explains why in Fig. 6.4(b) after $k \approx 4$ the error is constant (for the heuristic, more segments are unnecessary) and the same happens for the computation time (except for some few iterations, the computation is the same for $k \geq 4$).



(a) CNWLS

(b) CPLF

Figure 6.5: The time consumed by each method as we change k (the maximum number of segments in CPLF)

6.5 Some Regression Examples

Function Estimation Accuracy

We finish this chapter by presenting two illustrative regression examples. The first one bears on the precision of the regression methods, specially in terms of the derivative $\widehat{\phi}(\rho)$. In this first example we will use a training set obtained through the ns-2 simulator. A single commodity with a single two-link path generates TCP flows of a random size (exponentially distributed with mean 20 kB). Flows are generated as a Poisson process with an intensity that changes over time. The first link has a capacity of 2 Mbps and the second 1 Mbps. We concentrate on the queue from the second node to the destination. The training set together with the real function $f(\rho)$ may be seen in Fig. 6.6. The measurements (235 in total) were obtained by averaging over a two minute period, while the real function was obtained by averaging over long periods with the same traffic intensity. Observe that the precision of both methods in this case is remarkably good (in particular, for CPLF we used $k = 10$). Except for a small underestimation at the higher loads (due to the fewer number of observations) the real function and the estimations are almost identical.

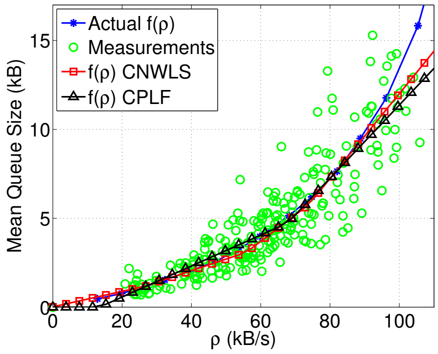


Figure 6.6: Noisy measurements, the real function $f(\rho)$, and the regression $\widehat{f}(\rho)$ for CNWLS and CPLF in the first example

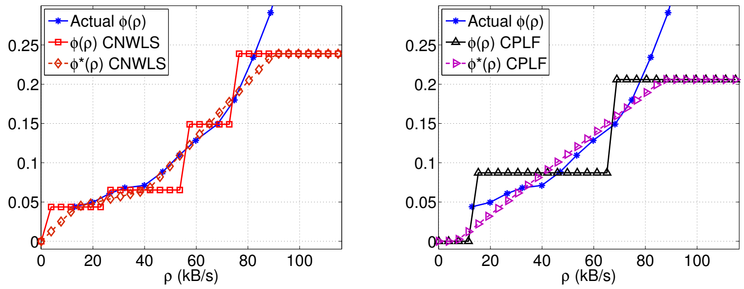
More interesting for our purposes is Fig. 6.7 which shows the real derivative (obtained by a simple difference quotient of the real function $f(\rho)$) and the regression for both methods. The first aspect that should be noted is that, since $\widehat{f}(\rho)$ is estimated as a piecewise linear function, the estimated $\widehat{\phi}(\rho)$ is piecewise constant. As we mentioned in Sec. 6.2, $f_l(\rho_l)$ should be continuously differentiable, meaning that $\phi_l(\rho_l)$ should be continuous, a condition that our estimation clearly does not fulfill. To address this issue we took a very simple approach, and approximate this piecewise constant function by a continuous piecewise linear one. The idea is the following. The piecewise constant function $\widehat{\phi}(\rho)$ may be characterized by the intervals in which its value is constant, and the corresponding value. If we have N^* pairs, we will have N^* of these intervals. We will assume that the first of these intervals starts at 0 and the last ends at the last available observation in the training set. Let us note the center of these intervals as ρ_j, where we will assume that $\rho_{j_1} < \rho_{j_2}\ \forall j_1 < j_2$. If we have to estimate the cost at a load ρ between ρ_j and ρ_{j+1}, then our continuous estimation will be:

$$\widehat{\phi}^*(\rho) = \frac{\rho_{j+1} - \rho}{\rho_{j+1} - \rho_j}\widehat{\phi}(\rho_j) + \frac{\rho - \rho_j}{\rho_{j+1} - \rho_j}\widehat{\phi}(\rho_{j+1}) \tag{6.7}$$

If we have to estimate the cost for $\rho < \rho_1$, we will then assume that there is a $\rho_0 = 0$ and that its cost is $\widehat{\phi}(\rho_0) = 0$. If we are on the other end and have to estimate the cost for $\rho > \rho_{N^*}$, we will simply return $\widehat{\phi}(\rho_{N^*})$. To characterize our continuous approximation we then require only the interval centers $\{\rho_j\}_{j=1,..,N^*}$ and their associated cost $\widehat{\phi}(\rho_j)$.

To clarify the explanation, we included in Fig. 6.7 this continuous approximation. Interestingly enough, $\widehat{\phi}^*(\rho)$ is actually a better approximation of $\phi(\rho)$ than the original $\widehat{\phi}(\rho)$. It is only natural that $\phi(\rho)$ is softer than a piecewise constant function. Finally, note that the resulting estimation for both methods becomes constant after, at the latest, no more observations are available. This aspect, which is due to the shape of the regression function, could be problematic and shall be further discussed later.

Let us finally mention that this continuous requirement is not necessary if we were to implement a centralized optimization scheme. Actually, minimization of a piecewise



(a) The real derivative $\phi(\rho)$, and the regression $\widehat{\phi}(\rho)$ for CNWLS

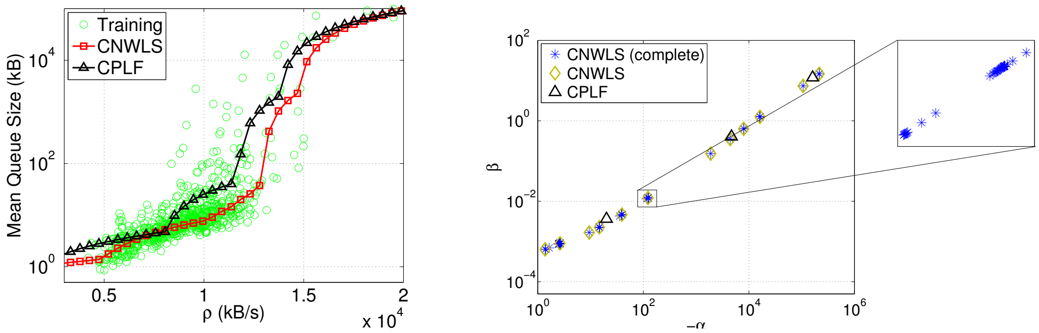
(b) The real derivative $\phi(\rho)$, and the regression $\widehat{\phi}(\rho)$ for CPLF

Figure 6.7: The derivative estimation for the first example

linear function can be easily transformed into a linear optimization problem, for which very fast solvers exist. However, our final objective is to design distributed schemes to solve the problem. As we shall see in the next chapter, the distributed optimization algorithm we propose (and probably *all* distributed optimization methods) requires a continuously differentiable $f_l(\rho_l)$ function.

Outliers Impact

We finish this section by presenting an example where we may appreciate the impact of the outliers on the two methods. We have used 720 random observations from the second day of the trace we used in the previous sections. The resulting approximative function $\widehat{f}(\rho)$ for the two methods may be seen in Fig. 6.8(a) (for CPLF we used $k = 10$). Note that, as already discussed in the previous section, CNWLS follows the bulk of the observations more closely than CPLF. The latter is greatly influenced by the relatively few (but significantly bigger than the rest) mean queue size measurements around $\rho = 10.000\,\mathrm{kB/s}$.



(a) The estimation of the mean queue size function

(b) The (α_j, β_j) pairs in the plane

Figure 6.8: The second regression example

Finally, we will use this example to illustrate the fact that the actual number of different (α_j, β_j) pairs for CNWLS is very small when compared to N. Figure 6.8(b) shows these pairs in the plane for another random 720 measurements long training set. Notice that, while N is equal to 720, N^* (the number of significantly different pairs) is only 14. A very simple clustering algorithm may be used to estimate the final set of pairs. For instance, consider A as the set of N^* (α_j, β_j) pairs that we shall finally consider in CNWLS. We may initialize this set with any pair of the complete set, and then iterate through the whole set, adding to A only those pairs whose relative difference in α or β with all the elements in A is bigger than a certain threshold. Concerning CPLF, the final number of pairs was reduced from 10 to 3.

Finally, let us highlight that Sec. 8.1.1 provides, after discussing some of the implementation issues associated with this framework, further considerations on which of the two regression methods should be preferred.

6.6 Related Work

Non-Parametric Convex Regression

In addition to CNWLS and CPLF, there exist other methods that make *no* assumptions on the regression function $\widehat{f}(\rho)$ and allow one to obtain its derivative. Probably the most prominent alternative is *Local Polynomial Regression* (for a good overview of this and other regression methods see [84]). This kernel-type regression method allows one to estimate any order derivative of the regression function through a standard weighted least square problem. However, it presents several shortcomings. First of all, evaluating the regression function at any point is as costly as the regression itself, i.e. a weighted least square problem has to be solved every time the function or its derivative is evaluated. In a way, the functional representation of $\widehat{f}(\rho)$ is the whole training set, which can be relatively big (compare this to CNWLS or CPLF, where $\widehat{f}(\rho)$ is represented by N^* and k parameters respectively). Secondly, all kernel-type methods suffer from the so-called bias-variance tradeoff, controlled by the bandwidth parameter, which can be very tricky to assign and on which the quality of the estimation depends heavily. Finally, in order to enforce shape constraints, such as monotonicity and convexity, "indirect" methods have to be used. For instance, [85] describes a method to transform the training set so that when the local polynomial method is applied, shape restrictions are assured. Anyway, the intrinsic problems of kernel-type methods we already mentioned are still present.

Non-Parametric Minimum Congestion Load-Balancing

To the best of our knowledge, there is only one paper similar to the ideas presented in this chapter. The authors of [86] present a dynamic load-balancing method that also strives to find the minimum congestion routing configuration. Differently to our work, they do not assume *any* model for the mean queue size function (i.e. this function is now $f_l(W)$, where the argument is the whole packet input process during the time period). Their approach to characterize the optimum is however relatively similar, in the sense that, based on the results discussed in Sec. 6.2, they use an algorithm similar to MATE on the path cost $\phi_P = \sum_{l \in P} \frac{\partial f_l}{\partial \rho_l}$. To estimate this derivative in such general case, they use the so-called *Perturbation Analysis*, which requires measuring the moment each packet enters and leaves the queue of link l.

The first disadvantage of this method is the measurements it requires. At the time of the proposal (late eighties), the assumption that the moment at which each packet of each link enters and leaves the queue was measurable, was somewhat reasonable. Nowadays, it simply cannot be done. Its second disadvantage, which at first may seem like the contrary, is the level of generality of their model. The optimization problem they wish to solve assumes a mean queue size function that depends solely on ρ_l (just like ours). However, their estimation of the derivative does not make such supposition, but depends on the whole packet process. This means that f_l depends on many more and unknown variables than just ρ_l. Note that since commodities may only change the portion of traffic sent through each path, these new variables are not controllable by them. This translates into oscillations, as presented in the original paper.

All in all, although our framework does not use the most general model, it does assume a “controllable” one. We can then guarantee convergence and stability, and expect that the mean total delay (or total congestion) obtained by the resulting demand vector is a good approximation of the absolute minimum. Moreover, the measurements required by our framework are available in most routers and need not be extremely precise.

Chapter 7

Achieving the Optimum: Routing Games and No-Regret Algorithms

7.1 Greedy Load-Balancing

In this section we present and discuss how to solve (5.2) and (6.1) in a distributed fashion. The solution to these problems have been characterized in Sec. 5.2 and 6.2 respectively. In both cases, having defined a certain path cost function ϕ_P, the optimum demand vector is the one in which, for each commodity, paths that are used present the minimum ϕ_P. As we shall now present, this situation may be regarded as the equilibrium resulting from OD pairs that use greedy load-balancing mechanisms.

In this kind of schemes, each commodity greedily minimizes the cost it obtains from each of its paths. This context constitutes an ideal case study for game theory, and is known as *Routing Game* in its lingo [19]. In a routing game like ours, where the traffic generated by a commodity may be arbitrarily distributed among paths, commodities are assumed to be constituted of an infinite number of agents. These agents control an infinitesimal amount of traffic, and decide along which path to send their traffic ($d_{P_{si}}/d_s$ represents then the fraction that have P_{si} as their choice for commodity s). If each of these agents acts selfishly, then the system will be at equilibrium when no agent may decrease its cost by unilaterally changing its path decision. This situation constitutes what is known as a *Wardrop Equilibrium* (WE) [20], which is formally defined as follows:

Definition 1. *A demand vector is a Wardrop Equilibrium if for each commodity $s = 1 \dots S$ and for each path P_{si} with $d_{P_{si}} > 0$ it holds that $\phi_{P_{si}} \leq \phi_{P_{sj}}$ for all $P_{sj} \in \mathcal{P}_s$.*

It is easy to see that in a WE, and for any given commodity, all paths with $d_P > 0$ have the same cost ϕ_P, namely the minimum among all paths of the corresponding commodity. It should be clear then that the optimum demand vector and the WE (for the corresponding path cost ϕ_P) are the same. We will then study how to achieve the WE given ϕ_P. However, let us first review the two most important kinds of games: *Congestion Routing Games* and *Bottleneck Routing Games*. We will also give a new interpretation to the utility maximization load-balancing scheme we presented in Ch. 5.

Congestion Routing Games

In both congestion and routing games, we first define a certain link cost function $\phi_l(\rho_l)$. The defining characteristic of a congestion game is that the path cost function is the addition of this link cost; i.e. $\phi_P = \sum_{l \in P} \phi_l(\rho_l)$. The idea is that $\phi_l(\rho_l)$, called in this context the *latency function*, represents the delay incurred by traffic traversing link l (hence the name *congestion*). In fact, the concept of a WE was introduced in the context of transportation, where the objective was to characterize the equilibrium of car users that greedily strive to minimize their travel time.

A well-known result first observed in [87] is that, if each user acts selfishly and $\phi_l(\rho_l)$ is nonnegative, nondecreasing and continuous, the resulting WE will not minimize the mean delay, but will result in a local minimum of the so-called *potential function*:

$$\Phi(d) = \sum_{l=1}^{L} \int_0^{\rho_l} \phi_l(x)dx \tag{7.1}$$

This result may be regarded as equivalent to the one we obtained in Sec. 6.2 (and is obtained almost in the same way): if we substitute $\phi_l(\rho_l)$ by $f_l'(\rho_l)$ in the above equation, we may easily verify that the WE coincides with the unique global minimum of $\sum_l f_l(\rho_l)$. Note that game theory is mostly interested in characterizing the resulting equilibrium of greedy users. For instance, there are several papers that study the so-called *Price of Anarchy* [88] (the ratio between the *worst* WE and the optimum of a certain objective function) or the *Price of Equilibrium* [89] (the ratio between the *best* WE and the optimum). We will proceed inversely. That is to say, given that the solution to our problem is a WE, we will study mechanisms that converge to this equilibrium.

Bottleneck Routing Games

Differently to congestion games, the path cost function in a bottleneck routing game is defined as the maximum among links of $\phi_l(\rho_l)$; i.e. $\phi_P = \max_{l \in P} \phi_l(\rho_l)$. This time, motivation comes from networking, particularly in the context of minimizing the maximum link utilization, as it is the case for Robust Routing (cf. Sec. 4.2.1) or REPLEX (cf. Sec. 4.3).

The authors of [90] studied general bottleneck games where the objective is to minimize the biggest link cost in the network, and showed that although the price of anarchy is unbounded, the price of stability is 1 under the so-called *efficiency condition* [90] (i.e. every WE that fulfills this condition is optimum):

Definition 2. *Let $N(P)$ denote the number of network bottlenecks over P; that is to say $N(P) = \left|\left\{l \in P : \phi_l(\rho_l) = \max_{m \in E, \rho_m > 0} \{\phi_m(\rho_m)\}\right\}\right|$. Then, WE d is said to satisfy the efficiency condition if all commodities route their traffic along paths with a minimum number of network bottlenecks; i.e. for all $P_{si} \in \mathcal{P}_s$ with $d_{P_{si}} > 0$ it holds that $N(P_{si}) \leq N(P_{sj})$.*

This definition may be interpreted as a second level path cost function that measures the number of bottlenecks in the given path. This result, which is relatively new, was not applied in the design of neither TeXCP or REPLEX, both of which strive to minimize the maximum link utilization by means of a greedy algorithm in the path utilization. It could then be the case that these algorithms converge to a sub-optimal WE. Towards the end of the Evaluation chapter (Ch. 8) we will discuss the possible consequences on the obtained performance of ignoring this result, and propose a possible alternative path cost function that strives to converge to the optimum WE.

Utility Maximization Routing Game

The utility maximization load-balancing framework we proposed in Ch. 5 may be interpreted as a mixture between congestion and bottleneck games. Instead of starting from the optimization problem (5.2), we could have simply assumed a bottleneck game (where $\phi_l(\rho_l) = -U(c_l - \rho_l)$), but where the objective function is the weighted mean of ϕ_P (like in a congestion routing game). With this new interpretation, the path cost we proposed in Sec. 5.2 may be regarded as a necessary correction to ϕ_P so that the WE coincides with the optimum.

7.2 No-Regret Algorithms

7.2.1 Definition and Results

In this section we will present an algorithm that, given the path cost ϕ_P, converges to the corresponding WE. In particular, we will consider so-called *No-Regret* algorithms. As mentioned in Sec. 4.3, previously proposed DLB algorithms always include a parameter that controls the convergence speed, which is very tricky to assign. Although for each algorithm there exists a range for this parameter in which it is stable, these values result in unresponsiveness in certain situations. We could simply increase this parameter, but this may result in oscillations. To avoid this reactivity-stability tradeoff, we will use a variation of the *Incrementally Adaptive Weighted Majority Algorithm* [22] (a no-regret algorithm), which presents the advantage of being completely self-regulated.

Before actually presenting the algorithm, let us present the result on convergence to the WE of no-regret algorithms, for which certain definitions are required. This kind of algorithm is iteratively applied over time, and as such we shall note the demand vector at time-step t as d^t. In the context of a routing game, the *per-time-step regret* incurred by a commodity is defined as the difference between its average path cost and the cost of the best fixed path at hindsight. That is to say, for a total time T, per-time-step regret for commodity s is defined as follows:

$$\frac{1}{Td_s}\sum_{t=1}^{T}\sum_{P\in\mathcal{P}_s} d_P^t\phi_P^t - \frac{1}{T}\min_{P\in\mathcal{P}_s}\sum_{t=1}^{T}\phi_P^t \tag{7.2}$$

The term *regret* comes from the fact that (7.2) measures the mean extra cost we incurred by not using exclusively the cheapest path (in the considered time interval). No-Regret algorithms are those for which the per-time-step regret may be upper bounded by zero as T goes to infinity. The authors of [21] proved that if all commodities apply no-regret algorithms, the resulting demand vector will converge towards the WE. Let us formally present this result, for which we will first define an ϵ-Wardrop Equilibrium (ϵ-WE) [21]:

Definition 3. *A demand vector d is a ϵ-WE if its total average path cost is within ϵ of the weighted mean cost of the minimum cost path available to each commodity, i.e.* $\sum_{s=1}^{S}\sum_{P\in\mathcal{P}_s} d_P\phi_P - \sum_{s=1}^{S} d_s\min_{P\in\mathcal{P}_s}\phi_P \leq \epsilon\sum_{s=1}^{S} d_s$.

Note that if $\phi_P \geq 0$ a 0-WE is simply a normal WE. Let us define as T_ϵ the number of time steps required to bound (7.2) by ϵ. We may now formally present the convergence result as obtained in [21]:

Theorem 2. *For a link-cost function $\phi_l(\rho_l)$ with maximum slope δ, for all but a ϵ' fraction of steps up to time T_ϵ, d^t is a ϵ'-WE, where $\epsilon = \Omega\left(\frac{\epsilon'^4}{\delta|E|^4+\delta^2|V|^2}\right)$. Moreover, T_ϵ depends quadratically on δ.*

Intuitively, this means that for *most* time steps, the instantaneous demand vector d^t is very near the WE, and this difference vanishes with time. However, it should be noted that the bigger the maximum derivative of $\phi_l(\rho_l)$, the slower the convergence speed. This means that, as mentioned in Sec. 6.5, $\phi_l(\rho_l)$ should be continuous for the algorithm to converge.

7.2.2 A No-Regret Algorithm: iAWM

The result presented in the previous subsection is very general, in the sense that it does not specify any algorithm in particular. Its only requirement is the use of no-regret algorithms by all OD pairs. In particular, we will consider the *Incrementally Adaptive Weighted Majority* (iAWM) algorithm [22], which originated in the context of online learning, in particular from the *online prediction using expert advice* problem [91, 92]. The algorithm does not only obtain a very good upper bound on the regret, but also presents the desirable characteristic of not requiring any parameter tuning. The complete pseudo-code for commodity s is described below.

Algorithm 3 Incrementally Adaptive Weighted Majority (iAWM) Algorithm

```
$L^t_{P_{si}} = 0 \; \forall i = 1, \ldots, n_s$
for $t = 1, \ldots, \infty$ do
    Receive the path cost $\phi^t_{P_{si}} \in [0,1] \; \forall i$
    $L^{t-1}_s \leftarrow \min_{i=1,..,n_s} L^{t-1}_{P_{si}}$
    $\varepsilon^t_s \leftarrow \min\left\{\frac{1}{4}, \sqrt{2\frac{\log n_s}{L^{t-1}_s}}\right\}$
    $\lambda^t_s \leftarrow \frac{1}{1-\varepsilon^t_s}$
    $W^t_s \leftarrow \sum_{i=1}^{n_s} (\lambda^t_s)^{-L^{t-1}_{P_{si}}}$
    $\omega^t_{P_{si}} \leftarrow (\lambda^t_s)^{-L^{t-1}_{P_{si}}} / W^t_s \; \forall i$
    Adjust the demand vector: $d^t_{P_{si}} = \omega^t_{P_{si}} d_s \; \forall i$
    Receive the outcome $y^t_s \in [0,1]$
    Update the path regret $L^t_{P_{si}} \leftarrow L^{t-1}_{P_{si}} + |y^t_s - \phi^t_{P_{si}}| \; \forall i$
end for
```

Each path has a corresponding regret $L^t_{P_{si}}$ that measures its performance up to time-step t (the bigger the path's regret, the worse its performance), which is initialized at 0 for all paths. At each time-step, the OD pair performs roughly four steps. Firstly, it receives the cost $\phi^t_{P_{si}}$ of each of its paths. Secondly, it calculates the value of λ^t_s (called the *learning rate*). Note how as the minimum regret among all n_s paths (L^{t-1}_s) increases, λ^t_s decreases accordingly.

The third step is to adjust the demand vector. The portion of traffic sent along path P_{si} (noted as $\omega^t_{P_{si}}$) depends on λ^t_s and on the corresponding path's regret $L^t_{P_{si}}$. Naturally, those paths with a smaller regret will get more traffic. Moreover, note that for the same set of regret values, the portion of traffic sent along the best path is bigger as λ^t_s increases. In a sense, λ^t_s controls the speed of the algorithm: the bigger the learning rate, the more reactive the demand vector adjustment. The interesting aspect of iAWM is that λ^t_s is automatically tuned. As mentioned before, as the regret of the best path increases, the algorithm's speed decreases.

The fourth and final step is to update the path's regret $L^t_{P_{si}}$. This is performed by simply accumulating the absolute difference between the path cost $\phi^t_{P_{si}}$ and a certain value y^t_s (called *outcome*). Let us define $\widehat{y}^t_s = \sum_{i=1}^{n_s} \omega^t_{P_{si}} \phi^t_{P_{si}}$. It may be proved that for any arbitrary sequence y^t_s of length T, the following inequality is verified when T goes to infinity [22]:

$$\sum_{t=1}^{T} \left|\widehat{y}^t_s - y^t_s\right| - L^T_s \leq (2.83 + o(1))\sqrt{L^T_s \log n_s} \tag{7.3}$$

The outcome y_s^t is a value that we consider as the objective. We shall now see that any value smaller than or equal to all path costs will serve our purposes. In particular, we have used:

$$y_s^t = \min_{i=1,..,n_s} \phi_{P_{si}}^t$$

The following theorem proves that iAWM is no-regret in the context of a routing game.

Theorem 3. *The algorithm iAWM using $y_s^t = \min_{i=1,..,n_s} \phi_{P_{si}}^t$ is no-regret in the context of a routing game.*

Proof. We need to prove that (7.2) is bounded, and that this bound goes to zero with T. In this case, it may be written as:

$$\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i=1}^{n_s}\frac{d_{P_{si}}^t}{d_s}\phi_{P_{si}}^t - \min_{i=1,..,n_s}\sum_{t=1}^{T}\phi_{P_{si}}^t\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i=1}^{n_s}\omega_{P_{si}}^t\phi_{P_{si}}^t - \sum_{t=1}^{T}(y_s^t - y_s^t) - \min_{i=1,..,n_s}\sum_{t=1}^{T}\phi_{P_{si}}^t\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\left(\widehat{y}_s^t - y_s^t\right) - \min_{i=1,..,n_s}\sum_{t=1}^{T}\left(\phi_{P_{si}}^t - y_s^t\right)\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\left|\widehat{y}_s^t - y_s^t\right| - \min_{i=1,..,n_s}\sum_{t=1}^{T}\left|\phi_{P_{si}}^t - y_s^t\right|\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\left|\widehat{y}_s^t - y_s^t\right| - L_s^T\right) \leq \frac{1}{T}(2.83 + o(1))\sqrt{L_s^T \log n_s} \leq$$

$$\frac{1}{T}(2.83 + o(1))\sqrt{T \log n_s} \underset{T}{\rightarrow} 0$$

□

As mentioned above, it is important that $y_s^t \leq \phi_{P_{si}}^t \, \forall i = 1, .., n_s$. This condition allows us to take the absolute value in the fourth step in the proof above. Using $y_s^t = \min_{i=1,..,n_s} \phi_{P_{si}}^t$ means that when the algorithm converges the regret of the best path does not increase. This fact will prove useful, as we will discuss later, when the algorithm is used with real, time-changing demands.

Finally, note that iAWM requires the path costs to be in the interval $[0, 1]$. This justifies the last inequality in the proof, since the regret of any path may never increase more than one unit per time-step (i.e. $L_s^T \leq T$). However, the path cost discussed in Ch. 6 is always positive, but its bound is not known a priori. In this case we can address this issue by using the alternative path cost $\widehat{\phi}_{P_{si}}^t = \frac{\phi_{P_{si}}^t}{\max_{P \in \mathcal{P}_s} \phi_P^t}$. In the general case, where ϕ_P may be any arbitrary real (as in the path cost discussed in Ch. 5), we can use the alternative path cost $\widehat{\phi}^t_{P_{si}} = \frac{\phi^t_{P_{si}} - K^t_s}{\max\limits_{P \in \mathcal{P}_s} \phi^t_P - K^t_s + C_0}$, where C_0 is a small constant to avoid numerical problems and K^t_s is a lower bound to $\phi^t_P \, \forall P \in \mathcal{P}_s$ (for instance, the smallest cost seen so far). Note that a routing game that uses any of the two $\widehat{\phi}^t_{P_{si}}$ instead of the original $\phi^t_{P_{si}}$ still has the same WE.

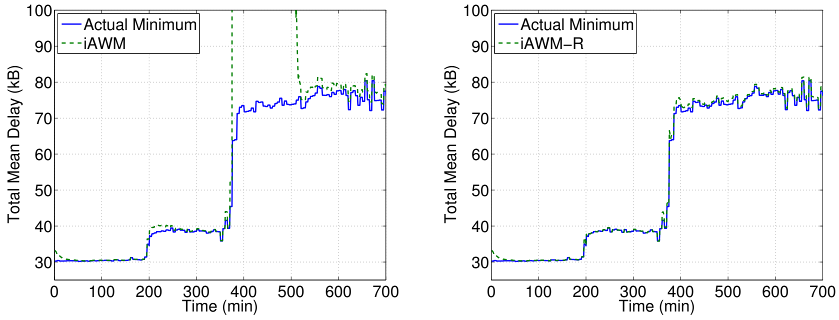
7.3 Some Preliminary Simulations: iAWM-R

In this subsection we shall present some first simulations that will help us gain insight into the framework and highlight one shortcoming of iAWM as it faces unforeseen and abrupt changes in the traffic demand. This will justify the final version of the load-balancing algorithm. Before, we will discuss how we performed these simulations.

As the reference network we used Abilene [93], an Internet2 backbone network. Abilene consists of 12 router-level nodes connected by 30 links (only intra-domain links were considered). The used topology and traffic demands are available at [94]. Traffic data consists of 6 months of traffic matrices collected every 5 minutes via Netflow. As measured demands do not significantly load the network, we re-scaled them by multiplying all their entries by a constant. Paths were constructed as discussed in [23], a method we will present with detail in Sec. 8.5.1. In particular, in this section we will consider a congestion routing game where $\widehat{f}_l(\rho_l)$ is the same for all links in the network (i.e. MinQ in Sec. 5.3), namely the one obtained in Sec. 6.5 for CNWLS and shown in Fig. 6.8(a).

For the present simulation we will consider 200 TMs from dataset X06 in [94], whose main characteristic is the anomalous increase of the traffic demand for two commodities. The simulation will be performed as follows. The TMs are fed to the mechanism in consecutive temporal order. The demand vector d is initiated at an arbitrary value d^0, which will be updated using iAWM as new link load measurements arrive. We will assume that each OD pair receives these measurements every minute, meaning that for each new TM the OD pairs will perform five updates of the portions of traffic sent along each of their paths. We will then calculate the total mean delay ($D(d) = \sum_l f_l(\rho_l)$ where $f_l(\rho_l)$ indicates the mean queue size, cf. Sec. 6.2) corresponding to each of these minutes. As a reference, we also computed the optimum value of $D(d)$ for every TM on the dataset. The difference between the obtained $D(d)$ and the optimum measures how far is d from the WE.

In Fig. 7.1(a) we show the results corresponding to this simulation. Note how iAWM rapidly converges to the optimum, and even when at $t = 200$ the first anomaly starts, there is only a small difference between the obtained total mean delay and the optimum. However, the behavior is quite the opposite for the second anomaly, which starts at $t \approx 400$. In this case, the overshoot is very important, and the convergence time is more than 100 minutes (or iterations in our case).



(a) iAWM (b) iAWM-R

Figure 7.1: The total mean delay as a function of time for the two load-balancing mechanisms

The reason behind this slow convergence is relatively simple. At the moment of the anomaly, for the commodity in question, the path through which some of the traffic is routed at optimality has a very bad history (i.e. a very big $L^t_{P_{si}}$). This means that for iAWM, this path has to be the cheapest for several iterations to revert its bad image and start being able to route traffic. When an anomaly occurs, conditions severely change and history should not be so relevant. If we consider that we are in such a situation, we could for instance completely ignore history and restart iAWM by setting $L^t_{P_{si}} = 0 \ \forall i$ for the commodity in question.

Regarding the decision of restarting iAWM, one may imagine several possibilities. In particular, we will consider that if $|y^t_s - \widehat{y}^t_s|$ is abnormally big, it constitutes a "suspicious" situation. By abnormally big, we mean that this difference exceeds the mean in previous iterations by a certain threshold L^{th}_s; in particular, we used $L^{th}_s = 0.1 \ \forall s$ (remember that y^t_s and $\widehat{y}^t_s$ are in the interval $[0, 1]$). Such a situation should indicate to us that the current situation does not correspond to what we have seen so far. However, it could also be due to noisy measurements. We will then require that this "suspicious" situation is repeated a certain amount m^{th}_s of consecutive times (in our case we used $m^{th}_s = 5 \ \forall s$). The complete pseudo-code of this new algorithm, which we called iAWM with Restart (iAWM-R) is presented below. The results obtained by iAWM-R may be seen in Fig. 7.1(b). Note how the overshoot has now disappeared almost completely. The temporal performance of iAWM-R will be further analyzed in the following chapter.

Before finishing this section, let us further justify the choice we made on y^t_s. With $y^t_s = \min_{i=1,\ldots,n_s} \phi^t_{P_{si}}$ we have two important advantages. Firstly, as the algorithm converges, $|y^t_s - \widehat{y}^t_s|$ goes to zero. If we would have used, for instance, $y^t_s = 0$, the "suspicious" situation would manifest by a decrease in this difference, complicating its detection. Secondly, not increasing the best regret when convergence is reached, improves the algorithm reactiveness (λ^t_s is not changed).

Algorithm 4 Incrementally Adaptive Weighted Majority with Restart (iAWM-R) Algorithm

```
Initialize $\tau_s^t \leftarrow 0$, $\widehat{L}_s^t \leftarrow 0$ and $m_s^t \leftarrow 0$
for $t = 1, \ldots, \infty$ do
  Perform a normal iteration of iAWM.
  $\tau_s^t \leftarrow \tau_s^{t-1} + 1$
  $\widehat{L}_s^t \leftarrow \widehat{L}_s^{t-1} + |y_s^t - \widehat{y}_s^t|$
  if $|y_s^t - \widehat{y}_s^t| > \widehat{L}_s^t / \tau_s^t + \widehat{L}_s^{th}$ then
    $m_s^t \leftarrow m_s^{t-1} + 1$
  else
    $m_s^t \leftarrow 0$
  end if
  if $m_s^t > m_s^{th}$ then
    $L_{P_{si}}^t \leftarrow 0 \ \forall i = 1, .., n_s$
    $m_s^t \leftarrow 0$
    $\tau_s^t \leftarrow 0$
  end if
end for
```

7.4 Related Work

No-Regret in the Context of On-Line Learning

The no-regret framework has been discovered and re-discovered in many research areas. For instance, in information theory [95] or computer science [96] (where it is called *Competitive Analysis*). As mentioned before, in this chapter we have considered an algorithm that comes from the context of online learning, in particular from the *online prediction using expert advice* problem. In this problem, in a sequence of time-steps, a learning algorithm is required to predict a value y^t. To produce its prediction, this algorithm takes into account the advice of N experts, where the advice of each expert is simply its prediction of y^t. After deciding its prediction $\widehat{y}^t$, the algorithm incurs a loss, measured by the loss function $L(\widehat{y}^t, y^t)$. Since it is impossible to guarantee an overall quality of the algorithm, the objective is to design an algorithm that predicts as well as the best expert for any sequence $\{y^t\}_{t=1,..,T}$, or at least that the difference with it is bounded.

If we note the i-th expert advice at time-step t as ξ_i^t, the loss of this expert is defined as:

$$L_i = \sum_{t=1}^{T} L(\xi_i^t, y^t)$$

The best expert in hindsight is the one that obtains the smallest loss, which we shall note as L^*. Analogously, the loss of the algorithm is:

$$L_{\text{alg}} = \sum_{t=1}^{T} L(\widehat{y}^t, y^t)$$

The regret of the algorithm in this context is the difference between L_{alg} and L^*. The idea is then to define an algorithm whose regret is bounded. In the previous section, for reasons that should be clear now, we used the absolute loss function, i.e. $L(\widehat{y}^t, y^t) = |\widehat{y}^t - y^t|$. Although it is probably the most common loss function, it has proved very challenging to design algorithms for it, and is still an active research area. In addition to the loss functions, the difference between no-regret algorithms also lies in the way of calculating the prediction $\widehat{y}^t$. In particular, and in order to associate each path with an expert, we used an algorithm whose prediction is a weighted mean of the experts' prediction.

Finally, let us highlight the recent article [97]. In it, the authors propose an online learning algorithm that, similarly to iAWM, is completely self-tuned. However, it presents the advantage of not requiring the outcome y^t or the paths cost ϕ_P^t to be in $[0, 1]$, and they may be any arbitrary real. We have only very recently found this article, and because of this we did not consider it further in the thesis. However, preliminary simulations made us realize that using $\widehat{\phi}_{P_{si}}^t = \frac{\phi_{P_{si}}^t}{\max_{P \in \mathcal{P}_s} \phi_P^t}$ presents a very important advantage: similarly to the situation that gave place to the reset in iAWM-R, not dividing may slow significantly the convergence speed of the algorithm (a single very big ϕ_P^t may take several time-steps to revert).

Tracking the Best Expert

Adapting on-line learning algorithms to non-stationary environments, as we have just discussed for iAWM, is an active research area. The alternative to restarting, is the more "continuous" framework of *tracking the best expert* [98]. The objective is the same as in the no-regret framework, except that we consider that time is divided into segments, and that each of them has a best expert. Performance of the algorithm is then compared to the performance obtained by this sequence of experts. Although it would be interesting to adapt such algorithms to be used for load-balancing, several aspects remain to be addressed. Firstly, convergence to the Wardrop Equilibrium has not been proved. Secondly, these algorithms are not self-configured which, as we discussed, is a very important property.

REPLEX: Exploration-Replication Policy

Regarding convergence to a WE, arguably the most prominent alternative to no-regret algorithms is REPLEX [6] (based on the game-theoretic algorithm presented in [64]), which we introduced in Sec. 4.3. Below, we can see the algorithm that each commodity executes in turn.

Algorithm 5 The REPLEX algorithm

```
$\omega'_{P_{si}} \leftarrow \omega_{P_{si}}\, \forall i = 1, .., n_s$
**for** every pair of paths $P_{si}, P_{sj}$ of commodity $s$ **do**
  **if** $\phi_{P_{si}} > \phi_{P_{sj}}$ **then**
    $\delta \leftarrow \lambda \left( (1-\beta)\, \omega_{P_{sj}} + \frac{\beta}{n_s} \right) \frac{\phi_{P_{si}} - \phi_{P_{sj}}}{\phi_{P_{si}} + \alpha}$
    $\omega'_{P_{si}} \leftarrow \omega'_{P_{si}} - \delta$
    $\omega'_{P_{sj}} \leftarrow \omega'_{P_{sj}} + \delta$
  **end if**
**end for**
$\omega_{P_{si}} \leftarrow \omega'_{P_{si}}\, \forall i = 1, .., n_s$
```

The algorithm converges to the WE as long as λ (the parameter that controls the speed of convergence) is smaller than k/r, where $k > 0$ is a suitable constant and r is an upper-bound to the relative slope of all $\phi_l(\rho_l)$, which is defined as follows [64]:

Definition 4. *A differentiable cost function $\phi_l(x)$ has* relative slope r *at x if $\phi'_l(x) \leq r\phi_l(x)/x$. A cost function has relative slope r if it has relative slope r over the entire range* $[0, 1]$.

Intuitively, migration from one path to the other should be slow if the cost function has abrupt changes. On the other hand, if the cost function is relatively "soft", changes may be faster. As discussed in [6], the values of α and β are not very influential, and $\beta = 0.1$, $\alpha = 0$ are generally good choices.

The most important inconvenience with REPLEX when compared with iAWM-R is the presence of the speed controlling parameter λ, specially in what respects guaranteeing convergence. Firstly, the numerical value of the constant k is not very clear. It may be interpreted as $1/32$ in the original paper (see the first paragraph in Sec. 2 of [64]), while in [6] nothing is said about it. Secondly, the definition of the relative slope only applies for cost functions whose argument is in the interval $[0, 1]$, an assumption that, at least in our case, we may not make.

Chapter 8

Evaluation

8.1 Introduction

In this chapter we will present a thorough evaluation of the three main propositions of this second part of the thesis. We will first study and compare the performance obtained by maximizing the utility (Ch. 5), minimizing the total mean delay (Ch. 6) or minimizing the maximum link utilization in the network. As in Sec. 7.3 we will use real topologies and demands, so as to study the performance of these objective functions in realistic scenarios. Secondly, we will concentrate on the minimum delay scheme, and analyze the impact of using a realistic mean queue size function $f_l(\rho_l)$, as opposed to a simplistic model (in our case, the M/M/1). Furthermore, we will analyze the temporal validity of a $f_l(\rho_l)$ learned from measurements. That is to say, how often does this function needs to be "relearned" so that the obtained performance is not affected significantly. Last but not least, we will present several simulations of the proposed load-balancing mechanism: iAWM-R (Ch. 7). We are particularly interested in its reactivity and convergence. In this sense, we will compare the mechanism with a Robust Routing scheme which, as mentioned in Sec. 4.2.1, is regarded as the alternative to DLB.

8.1.1 Implementation Issues

Before presenting the evaluation, we will discuss several implementation issues arising from each of the proposals.

Maximum Utility Load-Balancing

Let us first recall that in this case, the optimum traffic distribution d is the WE of the following path cost:

$$\begin{aligned}\phi_{P_{si}} &= -U\left(\min_{l\in P_{si}}\{c_l-\rho_l\}\right)+\sum_{l:l\in P_{si}}\widehat{\theta}_l\\ \text{where}\quad \widehat{\theta}_l &= \sum_{s=1}^{S}\sum_{i:l\in P_{si}}\theta_{P_{si}l}U'(c_l-\rho_l)\\ \theta_{P_{si}l} &= \begin{cases} d_{P_{si}} & \text{if } l=\operatorname*{argmin}_{l\in si}\{c_l-\rho_l\}\\ 0 & \text{otherwise}\end{cases}\end{aligned}$$

The first clear requirement of any load-balancing scheme is that border routers have to be able to send arbitrary portions of traffic along the different paths. Secondly, in this case in particular, and in order to measure $d_{P_{si}}$ and calculate $\theta_{P_{si}l}$, interior routers should distinguish between traffic belonging to a given path that is traversing its links.

These requirements are accomplished for instance by MPLS. Hashing can be used in order to load-balance traffic with an arbitrary distribution [99] and packets belonging to a given P_{si} can be identified by its label header. A counter for each of them should be kept by interior routers, indicating the number of routed bytes belonging to a given label. Periodically, each router calculates the corresponding $d_{P_{si}}$ by dividing its counter by the measurement interval, after which they reset it. The total load ρ_l is then calculated as the sum of all $d_{P_{si}}$ that use the link. The ABW of link l can be easily calculated as the difference between the total capacity and this value. However, in order to avoid numerical problems, the maximum between the ABW and a relatively small value (for instance $c_l/100$) should be used.

Another important aspect is the communication between links and the ingress routers (we have assumed that these routers, through which commodities inject traffic to the network, distribute this traffic). Explicit messages from the latter to the former are not necessary, since communication in that sense is simply how much traffic commodity s is sending through link l. It is true that what actually reaches the link will always be smaller or equal than originally at the source, but this approximation will not affect the algorithm's performance.

The most challenging communication in this case is from the link towards the ingress router. We will use the same approach as TeXCP [5] and use probe packets, which in our case will contain the path's ABW and total cost ($\sum_{l:l\in P_{si}}\widehat{\theta}_l$). Periodically, the ingress router sends a probe packet, initially indicating as the path's ABW and cost ∞ and 0.0 respectively. As the probe advances towards the destination node, each interior router checks the ABW indicated in it. If this value is bigger than that of the outgoing link, the router overwrites it with the link's ABW (and "remembers" it). When the destination node receives the probe packet, he sends it back to the ingress router along the same path but in the opposite direction. As it is going back, each interior router checks whether the final ABW indicated on the probe packet is the one its link had when the packet first passed. If this is the case, it means that it is the bottleneck of the particular path. It then calculates $\theta_{P_{si}l}$ accordingly, updates the link's total cost $\widehat{\theta}_l = \sum_s \sum_{i:l \in P_{si}} \theta_{P_{si}l} U'(c_l - \rho_l)$, and adds this value to the total path cost indicated on the packet. Finally, the ingress router receives the path's ABW and total cost, calculates $\phi_{P_{si}} = -U\left(ABW_{P_{si}}\right) + \sum_{l:l \in P_{si}} \widehat{\theta}_l$ and applies iAWM-R to update its load distribution (i.e. one iteration in Algorithm 4).

Regression-Based Minimum Congestion Load-Balancing

The implementation of this framework in a real-world network is relatively simple. Once all links have been characterized, each OD pair receives ρ_l from the links it uses (for this purpose, a TE-enabled routing protocol such as OSPF-TE may be used), calculates its paths' cost using (6.7) for each link, and applies iAWM-R to update the portion of traffic routed along each of them. This process is repeated regularly every few seconds. Regarding this update period, and this applies to Dynamic Load-Balancing in general, it should be long enough so that the quality of the measurements obtained is reasonable, but not too long to avoid unresponsiveness (in particular, we suggest 60 sec).

Regarding the learning phase (i.e. gathering the training set and performing the regression) we envisage several possibilities, differing in the degree of distribution of the resulting architecture. One possibility is that a central entity gathers the measurements, performs the regression and communicates the parameters obtained to all ingress routers. This first possibility presents the advantage that the new functionalities required on the router are minimal. However, as all centralized schemes, it may not be possible to implement in some network scenarios, and handling the failure of this central entity could be very complicated. An alternative is that links (or rather, the router at the origin of the link) perform the regression. Links keep the mean queue size measurements for themselves, perform the regression and communicate the result to ingress routers. The regression could be done once a day, in periods of low intensity (i.e. the night) so that normal operation is not affected. As we shall discuss further on, frequent updates in the regression function are not necessary.

An important aspect that should be analyzed is which measurements to keep for the training set. It is clear that newer measurements should be given priority over older ones. However, those corresponding to bigger loads will give us more information on $\phi_l(\rho_l)$ (and are more rare). A possible structure for the measurements database could be to partition the load into intervals, and keep the same amount of measurements per interval, each of which will work as a FIFO queue. As mentioned before, those intervals corresponding to bigger loads should be smaller, so as to have more information there.

Regarding the choice of the regression method, we have shown in Sec. 6.4 that the precision obtained by CNWLS may be significantly better than that of CPLF. However, as we shall present in Sec. 8.3.1, this translates in a difference in the obtained total mean delay that is much smaller. In any case, whether the extra computational burden incurred by CNWLS is worth the performance improvement depends on the given situation. If the size of the training set we are working with is significant, or the computation capacity available limited, we could fall back on CPLF. If not, we recommend using CNWLS. Not only does it obtain a better precision, but its solution does not rely on a heuristic and may be calculated exactly.

8.2 The Three Objective Functions: A Performance Comparison

In this section we present a comparative study between the three objective functions considered so far. We will reference each objective function the same way we did in Sec. 5.3; i.e. MinQ (as in Minimum Queue), MinMaxU (as in Minimum Maximum Utilization) and MaxU (Maximum Utility). Regarding MinMaxU, its objective is not simply minimizing the maximum link utilization ($u_{\max}$), but rather converging to the WE of a bottleneck game with $\phi_l(\rho_l) = u_l$ (just like in TeXCP or REPLEX). It should also be noted that the results we will show for MaxU were obtained by repeatedly applying iAWM-R using the path cost ϕ_P defined in (5.10). Differently to Sec. 5.3, MinQ will use a $f_l(\rho_l)$ that corresponds more to reality than the simple *M/M/1* model we used before. In particular, and as in Sec. 7.3, we shall use the estimated $\widehat{f_l}(\rho)$ obtained by CNWLS in Sec. 6.5, and that may be seen in Fig. 6.8(a).

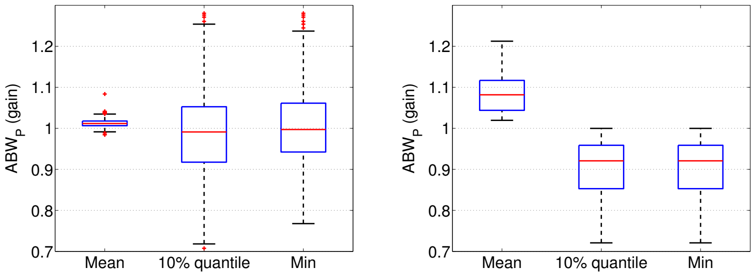
The comparison will be made in two real networks, with several real demands, calculating for each of these TMs the optimum demand vector for the three objective functions. Moreover, for each of these demand vectors and to be as fair as possible, we will calculate the three performance indicators each objective function is interested in, and whose importance we have already discussed: path's ABW (ABW_P), link utilization (u_l) and the total mean delay ($D(d) = \sum_l f_l(\rho_l)$). We could consider other performance indicators, such as path's propagation delay. However, we shall suppose that it has already been taken into account by the operator in the paths' choice.

8.2.1 Abilene Network

In this subsection we present the performance analysis for Abilene [93], a network we have already presented in Sec. 7.3. In this case, we used 613 demands (spanning a complete week) from dataset X01 of [94]. The paths are the same we used in Sec. 7.3, and were constructed as in [23].

We will first present the ABW_P results. For each demand we calculated the weighted mean ABW_P, where the weight of path P_{si} is $d_{P_{si}}$. This average provides us with a rough idea of the performance as perceived by traffic. A good value of this average indicator could however hide some pathological cases where some portions of traffic obtain a bad performance. That is why we also measured the 10% quantile and the minimum ABW_P. The comparison will be made by dividing the value obtained by MaxU by those obtained by the other objective functions.

In Fig. 8.1 we can see the boxplots of the ABW_P indicators. We first note that the weighted mean of ABW_P is almost always bigger in MaxU than in MinQ or MinMaxU. In particular, the increase with respect to MinMaxU is generally between 4% and 12%, and it may be as big as 20%. On the other hand, the difference with MinQ is much smaller, generally between 1% and 2%, and no more than 5%. Regarding the 90% quantile and the minimum ABW_P, and as expected, the best results are obtained by MinMaxU. No conclusive results may be obtained when comparing MaxU and MinQ.

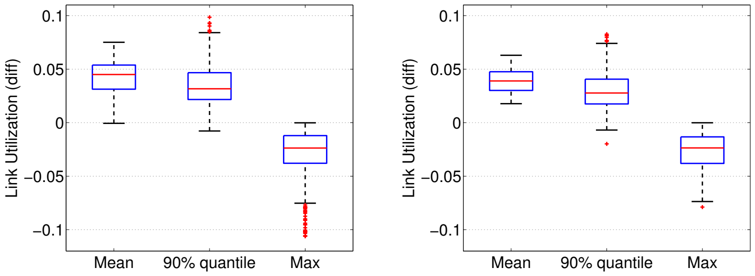


(a) MaxU vs MinQ

(b) MaxU vs MinMaxU

Figure 8.1: ABW_P for MaxU, MinQ and MinMaxU in the Abilene network

We now turn our attention to the link utilization (u_l) results. For each demand we calculated the mean, 90% quantile and maximum utilization on the network for each of the considered objective functions. The comparison will be made by making the difference between the indicator obtained by MinMaxU and the other two. Fig. 8.2 shows these results. Firstly, and as expected, the best results in terms of the maximum link utilization are obtained by MinMaxU, but with a difference that is generally no more than 5%. Quite surprisingly, the 90% quantile and the mean are bigger in MinMaxU than in the rest. The reason behind these results is that MinMaxU is so conservative in its objective, that, although it minimizes the maximum link utilization, it overlooks the less loaded links.



(a) MinQ vs MinMaxU

(b) MaxU vs MinMaxU

Figure 8.2: u_l for MaxU, MinQ and MinMaxU in the Abilene network

Let us now discuss the results on total mean delay ($D(d)$). In this case, for each TM we calculated the $D(d)$ obtained by each of the objective functions, and we present the division of that obtained by MaxU and MinMaxU, by the optimum obtained by MinQ. Figure 8.3 presents the results. Naturally, the best performance is achieved by MinQ. Differently to the other performance indicators, the difference between the objective functions may be significant this time. For instance, the increase in $D(d)$ incurred by MinMaxQ is generally between 15% and 30%, and may exceed 50%. On the other hand, MaxU obtains relatively better results, with an increase that is generally between 5% and 30% (with a median that is only 6%). However, the difference with MinQ may exceed the 60%.

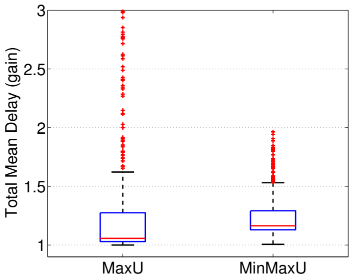


Figure 8.3: $D(d)$ for MaxU, MinQ and MinMaxU in the Abilene network

8.2.2 Géant Network

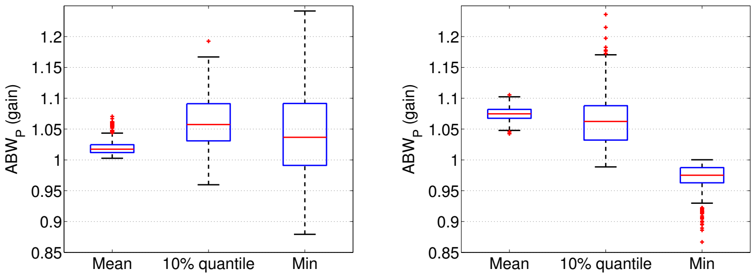
The second case scenario is Géant [100]. This European academic network connects 23 nodes using 74 unidirectional links, with capacities that range from 155 Mbps to 10 Gbps. The topology and demands (383 in total, covering a three week period) were obtained from TOTEM's webpage [101, 102, 103]. In this case paths were constructed by a shortest path algorithm, where we used the inverse of the capacity as the link's weight. For each commodity we computed two paths. The first is simply the shortest path, we then prune the network of the links this path uses, and compute the second shortest path.

Unfortunately, the great variety in link capacities forced us to use an artificial $f_l(\rho_l)$. In this case, we assumed that the mean queue size depends solely on the link utilization (as it is the case for the *M/M/1* model). We had the realistic $f_l(\rho_l)$ we used in the previous section, which corresponds to a link of capacity c_{ref}, given by the (α_j, β_j) pairs. We then calculated $f_l(\rho_l)$ for an arbitrary capacity c_l as follows:

$$f_l(\rho_l) = \max_{j=1,\dots,N} \left\{ \alpha_j + \beta_j \frac{c_{\text{ref}}}{c_l} \rho_l \right\} \tag{8.1}$$

This is equivalent to having the link characterization $(\alpha_j, \beta_j c_{\text{ref}}/c_l)$ for a link with capacity c_l. The alternative to this approximation is to re-generate the training set to obtain a new link characterization for each link capacity, for which there are two possibilities. The first one is to simulate the router using as an input any packet trace, and simply decrease/increase the link capacity. This would be equivalent to assuming that the link capacity does not influence the sources behavior, which is not true, specially as we decrease it[1]. A second alternative is to try to generate a traffic trace whose intensity depends on the link capacity (for instance, by thinnning the input process). However, transforming the packet trace would influence its statistical properties in ways we are not sure of, and designing such "invariant" transformation clearly escapes the scope of the present work. We will then use the simple approximation (8.1), but taking into account that these are artificial results, similar to using the *M/M/1* model. They should then be considered as illustrative of the general minimum-congestion framework, rather than of our regression-based one.

Results for the ABW_P in this case can be seen in Fig. 8.4. This time, the comparison between MaxU and MinQ is more favorable to the former, specially in the 10% quantile and minimum ABW_P. Results for the mean ABW_P are similar to those obtained for Abilene, with a mean that is generally between 1% and 3%, and at most 7%. On the other hand, the results of the comparison between MaxU and MinMaxU are relatively similar to the ones obtained before. Although the difference is smaller, the mean ABW_P is still always bigger in MaxU, with an increase that is generally between 7% and 8%, and as high as 11%. The most important difference is in the 90% quantile. This time MaxU obtains the best results, with a difference that may be as big as 18%. With respect to the minimum ABW_P, the results are logically better for MinMaxU, but the difference is generally not significant.



(a) MaxU vs MinQ

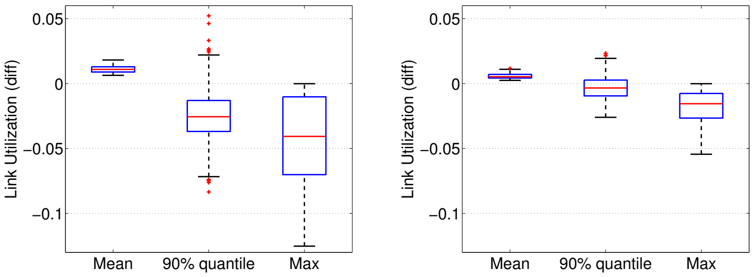
(b) MaxU vs MinMaxU

Figure 8.4: ABW_P for MaxU, MinQ and MinMaxU in the Géant network

Figure 8.5 shows that the results for the link utilization are now more favorable to MaxU than MinQ, specially in terms of the maximum link utilization. We may see that the results obtained by MinQ are somewhat worse than before, with a difference with respect to MinMaxU that is generally between 1% and 7%, and that may be as much as 12%. On the other hand, the increase incurred by MaxU is always less than 5%, with a median of only 2%.

Finally, we present the results on the total mean delay for this case in Fig. 8.6. It

[1] Unfortunately, the only publicly available traffic trace we are aware of and which significantly loads its link, is the one we used to obtain the present link characterization.



(a) MinQ vs MinMaxU (b) MaxU vs MinMaxU

Figure 8.5: u_l for MaxU, MinQ and MinMaxU in the Géant network

is interesting to note that the difference between the three objective functions is now almost insignificant. It is very difficult to measure the influence of the approximation we used to estimate $f_l(\rho_l)$ in this case, but it may seem that as link capacities are more varied across the network, the difference in $D(d)$ between the objective functions tends to be smaller.

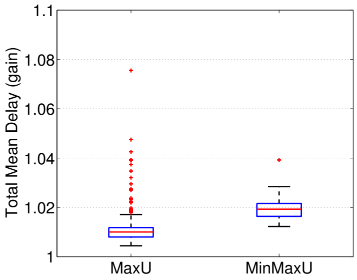


Figure 8.6: $D(d)$ for MaxU, MinQ and MinMaxU in the Géant network

8.3 Regression-Based Minimum Congestion Load-Balancing

8.3.1 Assessing the Performance Gain

In this section we interest ourselves to the potential gain in total mean delay ($D(d) = \sum_l f_l(\rho_l)$) achieved by using a good estimation of the real $f_l(\rho_l)$ instead of a simplistic model. We will also measure the loss in performance due to the relatively imprecise estimation of CPLF with respect to CNWLS. As mentioned before, the most commonly used model is *M/M/1* [4] (which results in $f_l(\rho_l) = \rho_l/(c_l - \rho_l)$), and as such we will use it as the reference. In particular, we will consider that the real mean queue size function $f_l(\rho_l)$ is the one obtained by CNWLS in Fig. 6.8(a). The resulting derivative for CNWLS as well as CPLF are shown in Fig. 8.7, where we also include the *M/M/1* model (more precisely, a piecewise linear approximation of the *M/M/1* model). It should be clear from the graph that if we assume the *M/M/1* model instead of the real $f_l(\rho_l)$ we would incur an increase of the total mean delay with respect to the optimum that may be important.

To quantify this increase more precisely, we will compare their performance in the Abilene topology, and as in Sec. 8.2.1 we will assume the same $f_l(\rho_l)$ for all links. We will also consider the same 613 TMs we used in Sec. 8.2.1. For each of these traffic demands, we calculated the optimum demand vector for each link cost function. We will measure the difference in $D(d)$ assuming the $\widehat{f}_l(\rho_l)$ obtained by CNWLS as the true $f_l(\rho_l)$. Just like before, we shall note these schemes as MinQ, but this time we will indicate the link cost function between parentheses (e.g. MinQ(CNWLS)).

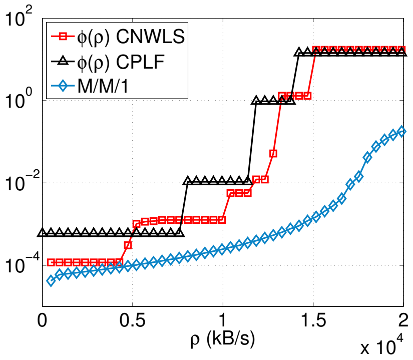
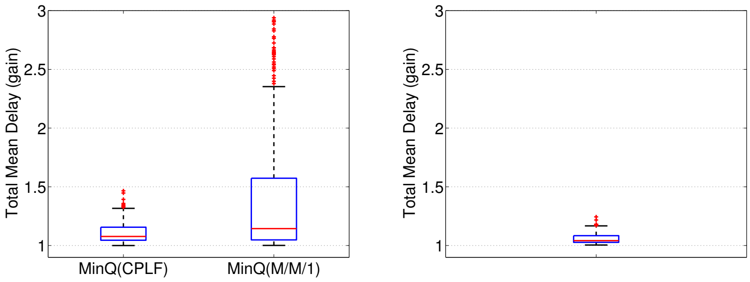


Figure 8.7: The estimated link cost used on the comparison and the *M/M/1* model

In Fig. 8.8(a) we can see the boxplot of the results. In particular, for each traffic demand we calculated $D(d)$ for the considered cost functions: CNWLS, CPLF and M/M/1. We present the division between the value obtained by each scheme and MinQ(CNWLS). The results verify that the increase in total mean delay can be important when using the *M/M/1* model. In particular, the total mean delay obtained by MinQ(M/M/1) is generally between 5 and 55% bigger than the one obtained by MinQ(CNWLS). This difference may actually go as high as a 135%, and in some cases even more (although not shown for the sake of clarity of the graph, the actual maximum was 600%). If we look carefully at Fig. 8.7 we can see that this difference originates in the fact that the *M/M/1* model underestimates $\phi_l(\rho_l)$. In particular, the abrupt increase in queue size that occurs at $\rho \approx 12\,\text{MB/s}$, is also present in the *M/M/1* estimation, but at a much higher load of $\rho \approx 17.5\,\text{MB/s}$. This leads iAWM-R to "believe" that links are operating at a low queue size load, when it is actually the opposite. Regarding MinQ(CPLF), we may verify that the loss in precision does not seriously impact the obtained performance. Except for some isolated cases where the difference may be as big as 30%, the increase in total mean delay is generally between 0% and 15%.

For the sake of completeness, we will also make the comparison in terms of the link utilization (i.e. $u_l = \rho_l/c_l$). As a reference for the comparison, and as in the previous section, we will use the results obtained by MinMaxU. We will measure the three

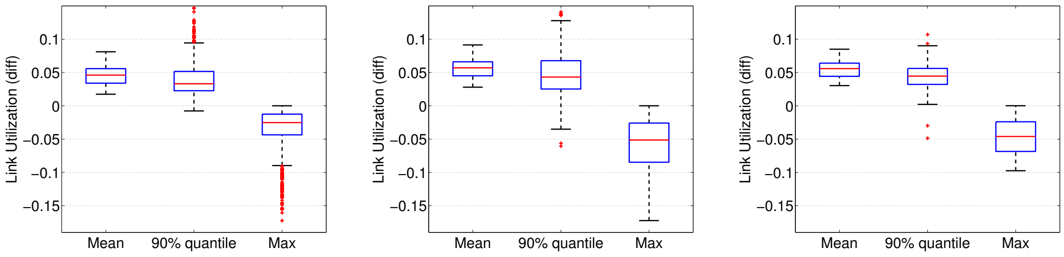


(a) MinQ(CPLF) and MinQ(M/M/1) with respect to MinQ(CNWLS)

(b) Using the $\widehat{f_l}(\rho_l)$ of the day before

Figure 8.8: Increase in Total Mean Delay in the Abilene network

same network-wide performance indicators: the mean, 90% quantile and maximum link utilization. We calculated the results obtained by all the mechanisms, and present the difference between the reference and the other schemes, which we show in Fig. 8.9. It should be noted that the results for all versions of MinQ are very similar, and as in the previous section, the difference with the MinMaxU is not significant.



(a) MinQ(CNWLS)

(b) MinQ(CPLF)

(c) MinQ(M/M/1)

Figure 8.9: Difference in link utilization between MinMaxU and the different versions of MinQ in the Abilene network

Several conclusions may be drawn from these experiments, some of which we would like to highlight. As shown in the graphs of Fig. 8.9, in terms of link utilization the exact choice of $f_l(\rho_l)$ is not crucial as long as it is increasing, convex and diverges (or has a big derivative) as it reaches c_l. However, when it comes to minimizing the total mean delay, this is no longer true. We have shown that the link queue size function obtained from measurements differs from the *M/M/1* model. Although this last statement is not a novel result, we have also shown that this difference significantly impacts on the obtained total mean delay. As shown in Fig. 8.7, the *M/M/1* model fails to predict the abrupt increase in mean queue size at relatively low loads. Whatever are the causes of this increase (e.g. long-range dependence have been observed to generate big queueing delays at relatively low loads [104]), it is probably the main reason behind this difference in $D(d)$. Finally, we have also shown that CPLF is a relatively good substitute of

CNWLS and that its obtained total mean delay is close to the real optimum. It is precisely because CPLF estimates this abrupt increase at a relatively precise link load value that it obtains such good results.

8.3.2 Temporal Behavior

A natural question that arises in our framework is how often links need to be characterized. In other words, how long can $\widehat{\phi}_l(\rho_l)$ be used as a good approximation of $\phi_l(\rho_l)$? Although more frequent updates of the links characterization will mean a more optimal or fine-tuned network, it will also mean greater computational expenses. This tradeoff between the optimality of the network and computational burden should be addressed.

Here we will give a partial answer to this question, and, as a reference, provide a lower bound to the validity of the link characterization used as a reference in the previous subsection (which we will note as $(\alpha_i, \beta_i)_{\mathrm{PREV}}$). The idea is the following. From the same 72 hours long packet trace used before, we take the same 12 hours worth of measurements, but from the next day. We will note the CNWLS characterization resulting from this new training set as $(\alpha_i, \beta_i)_{\mathrm{NEXT}}$. We now assume that the correct $f_l(\rho_l)$ for all links in Abilene is $\widehat{f}_l(\rho_l)_{\mathrm{NEXT}}$, and measure the increase in the total mean delay if we were to minimize the total mean delay using $\widehat{f}_l(\rho_l)_{\mathrm{PREV}}$ instead.

In Fig. 8.8(b) we show the results obtained in this case. We can see that although in some few cases the increase due to the misspecification may be as much as 20%, it is generally under 10%. These results are to be compared to those obtained by MinQ(M/M/1), which obtained an excess in the total mean delay of more than 15% in half of the cases, and a maximum difference of more than 100%.

Our partial answer is then that the characterization of a link obtained from the measurements of any given day, is also valid the next day. This validates our implicit assumption that $\phi_l(\rho_l)$ remains relatively stable over time. Another positive conclusion is that regression need not be performed very frequently. It should be noted that the trace used in this study contained only working days. Our conjecture is that the characterization obtained from any working day holds for the rest of the working days in the same week. The traffic mix generally changes on weekends, which will probably result in a different $f_l(\rho_l)$ than that of the working days, thus requiring its own characterization.

8.4 Packet-Level Simulations

In this section we will consider some relatively simple examples we implemented in ns-2 [39] that will verify the correct performance of iAWM-R in the presence of delayed and noisy measurements. It is important to highlight that in all the simulations, and as mentioned in Sec. 8.1.1, load balancing is performed at the granularity of flows, i.e. once a flow is routed along a path it stays there throughout its lifetime, and is random, i.e. commodity s will route new incoming flows along path P_{si} with probability $\omega^t_{P_{si}}$ (cf. Sec. 7.2.2). Finally, note that in this section we have used $\widehat{\phi}^*_l(\rho_l)$, the continuous approximation of the link-cost $\widehat{\phi}_l(\rho_l)$ (cf. (6.7) in Sec. 6.5).

8.4.1 Small Buffers and the Regression-Based Minimum Congestion Load-Balancing

In this subsection we will analyze what happens when in the regression framework described in Ch. 6 one or more links systematically present a small queue size. The network of the considered example may be seen in Fig. 8.10, and is very similar to the third example in Sec. 5.3. Its six "core" links have a capacity of 125 kB/s, while the "access" ones 250 kB/s. There are a total of four commodities, all with the same destination node q, except for commodity 3, whose destination node is origin 4. Moreover, only commodity 1 has more than one path available. We will note $\omega_{P_{11}}$ the portion of traffic of commodity 1 routes along the upper path. The traffic in the network is a mixture of elastic and streaming flows. The elastic ones (whose size is exponentially distributed with mean 20 kB) are generated as a Poisson process of intensity λ_e. The streaming traffic is constituted of CBR flows (at a bitrate of 10 kbps and an exponentially distributed duration with mean 20 s) also arriving as a Poisson process (of intensity λ_s). The corresponding intensities were calculated so that streaming represents 10% of the total traffic. Finally, the traffic distribution update and the link load measurements are performed over a 60 s period.

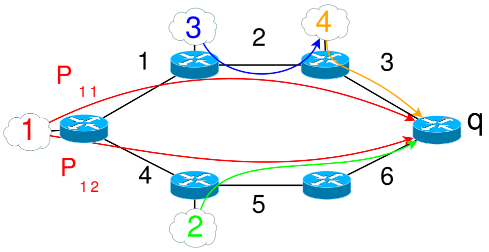
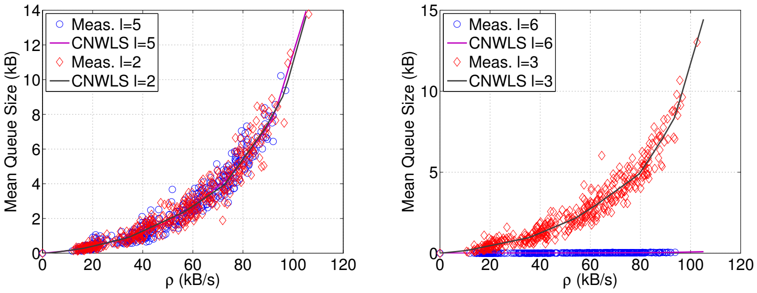


Figure 8.10: The network for the packet-level simulations.

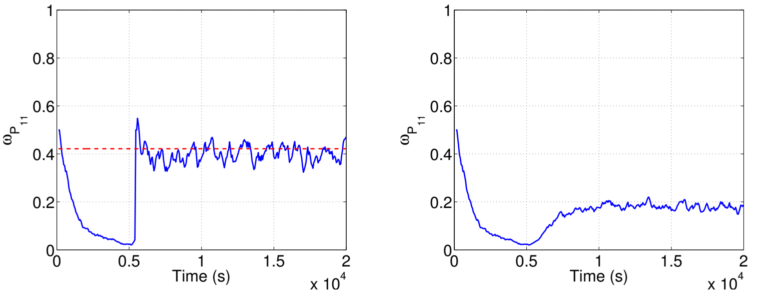
The training set has been constructed by fixing $\omega_{P_{11}}$ at 0.5 and varying the demand generated by each commodity. The training set and the corresponding regression for links 2, 3, 5 and 6 may be seen in Fig. 8.11. Note that links 2, 3 and 5 have almost the same characterization. On the contrary, link 6 presents little or no queue. This is because traffic on this link is already shaped by the queue of link 5. Links with little or no queue clearly represent a problem for our framework. This small queue may be due to the traffic characteristics as in this case, or simply because the link has a small buffer. Such links will have a constant insignificant cost, and the converged demand vector could overload them. This could also be the case for a characterization resulting from an incomplete training set. If the maximum measured load is relatively small, as observed in Sec. 6.5, the resulting cost function $\widehat{\phi}^*_l(\rho_l)$ will become a small constant.



(a) Links $l = 2$ and $l = 5$ (b) Links $l = 6$ and $l = 3$

Figure 8.11: The training sets and the corresponding regressions

To illustrate the above mentioned problems, we have conducted two simulations. In both of them, the total traffic intensity for commodities 2, 3 and 4 is 50 kB/s throughout the simulation, while for commodity 1 it changes from 12.5 kB/s to 75 kB/s at $t = 5000$ s (in a total of 20000 seconds). The difference between the two simulations lies in the characterization used for link 5. In the first one we used the complete characterization, as it appears in Fig. 8.11(a), while in the second one we have removed the (α_j, β_j) corresponding to the biggest β_j. This means that after $\rho \approx 75$ kB/s the link cost function becomes constant. In Fig. 8.12 we show the evolution of $\omega_{P_{11}}$ over time for the two simulations. Notice how the $\omega_{P_{11}}$ corresponding to the complete link characterization converges to a value very near the optimum we calculated off-line (which is marked by a horizontal line). On the other hand, the incomplete link characterization makes iAWM-R believe that link 5 is cheaper than it really is, resulting in $\omega_{P_{11}}$ converging to 0.2. In the first case, it is interesting to verify how the restart in iAWM-R is only applied when the abrupt change in demand occurs. For the second training set, the difference in cost is not enough to trigger a restart.



(a) $\omega_{P_{11}}$ as a function of time using the complete link characterization (b) $\omega_{P_{11}}$ as a function of time using the incomplete characterization of link 5

Figure 8.12: The simulation using two link characterizations

In Fig. 8.13 we present the load for all "core" links on the simulation corresponding to the incomplete link characterization. Naturally, links 5 and 6 are overloaded in the second part of the simulation. It is interesting to notice how the total demand generated by commodity 1 (approximately 120 kB/s from Fig. 8.13) exceeds the traffic intensity of 75 kB/s. This is because at overload, due to retransmissions, the mean traffic generated by each TCP flow exceeds the 20 kB. This means that, although the implicit assumption that demands do not depend on network condition or routing is not fulfilled in this simulation, the load-balancing algorithm still converges. Another interesting aspect to note is that although the level of noise in the load measurements is important, its impact on the converge of $\omega_{P_{11}}$ is relatively small.

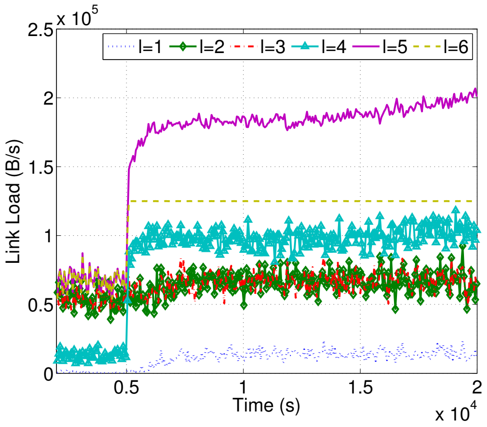


Figure 8.13: The links load as a function of time using the incomplete characterization of link 5

We will now present a possible solution to the above problems. The idea is to avoid overloading links due to unobserved link loads or little mean queue size. We propose to substitute $\widehat{\phi}_l^*(\rho_l)$ by a certain function $\psi_l(\rho_l)$ for $\rho_l > \rho_l^{\max}$, where $\psi_l(\rho_l)$ has the following characteristics:

- The resulting cost function should be continuous: $\psi_l(\rho_l^{\max}) = \widehat{\phi}_l^*(\rho_l^{\max})$.
- $\psi_l(\rho_l)$ should be increasing on ρ_l and convex.
- As load exceeds $\rho_l^{\max}$, all links should have a big and similar cost. Else, the algorithm would still prefer to overload the cheapest links.

Let us temporarily define $\psi_l(\rho_l)$ as:

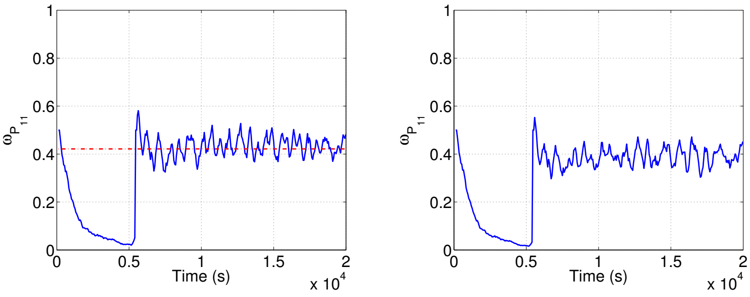
$$\begin{aligned} \psi_l(\rho_l) &= e^{b_l\rho_l/c_l} - 1 \\ \text{with } b_l &= \frac{c_l}{\rho_l^{\max}} \log\left(1 + \widehat{\phi}_l^*(\rho_l^{\max})\right) \end{aligned} \tag{8.2}$$

We suggest an exponential because, in our experience, it is a rough approximation of the shape of the $\widehat{\phi}_l^*(\rho_l)$ obtained from measurements. Function (8.2) goes through the origin and at $\rho_l^{\max}$ is exactly $\widehat{\phi}_l^*(\rho_l^{\max})$. For each link, $\psi_l(\rho_l)$ then fulfills the two first characteristics of the above list. To fulfill the third one, we will take the maximum b_l among all links ($b_{\max}$), and use the resulting function as the definitive substitute cost $\psi_l(\rho_l)$. The final link cost is then:

$$\phi_l(\rho_l) = \begin{cases} \widehat{\phi}_l^*(\rho_l) & \text{if } \rho_l < \rho_l^{\max}, \\ e^{b_{\max}\rho_l/c_l} - e^{b_{\max}\rho_l^{\max}/c_l} + \widehat{\phi}_l^*(\rho_l^{\max}) & \text{else} \end{cases} \tag{8.3}$$

As the threshold $\rho_l^{\max}$ we could use the biggest ρ_l in the training set, a certain fraction of c_l, or the minimum of both. There are many possibilities. In particular, in our simulations we used the last option, with a fraction of 80%. It could happen that the link capacity was not known, in which case we should then substitute it by a load that we consider critical for the operation of the link. The exact value of c_l in (8.3) is not so important and an order-of-magnitude value should be enough. Finally, note that as long as the load at all links is smaller than the corresponding $\rho_l^{\max}$, the presence of $\psi_l(\rho_l)$ has no influence.

We will now repeat the two previous simulations, but using the corrected link cost function (8.3). In Fig. 8.14(a) we may see that the influence of $\psi_l(\rho_l)$ when using the complete characterization is negligible. The differences with Fig. 8.12(a) are due to a relatively incomplete training set on links 1 and 4. In Fig. 8.14(b) we may see how the presence of $\psi_l(\rho_l)$ when using the incomplete link characterization plays a fundamental role in the convergence of $\omega_{P_{11}}$. In Fig. 8.15 we may verify that no link is overloaded now.



(a) $\omega_{P_{11}}$ as a function of time using the complete link characterization

(b) $\omega_{P_{11}}$ as a function of time using the incomplete characterization of link 5

Figure 8.14: The simulations using the corrective function $\psi_l(\rho_l)$

8.4.2 Two iAWM-R Commodities

We will now present a simulation where two iAWM-R routers interact. In particular, we will present an example where we strive to maximize the total utility (i.e. MaxU in the previous section). In Fig. 8.16 we can see the considered case scenario. There are two commodities and each of them can use two paths, one of which is shared. As before,

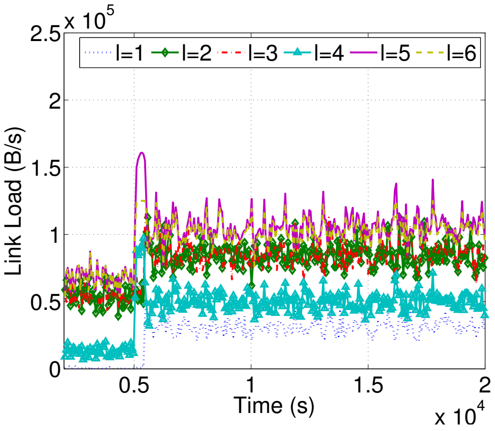


Figure 8.15: The links load as a function of time for the incomplete characterization of link 5

all links have a capacity of 125 kB/s, except for the "access" ones which have 250 kB/s. Traffic consists of elastic flows with an exponentially distributed size with mean 20 kB, arriving as a Poisson process of the corresponding intensity so as to generate a certain traffic intensity which we shall change with time.

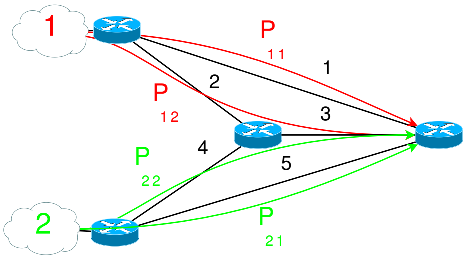
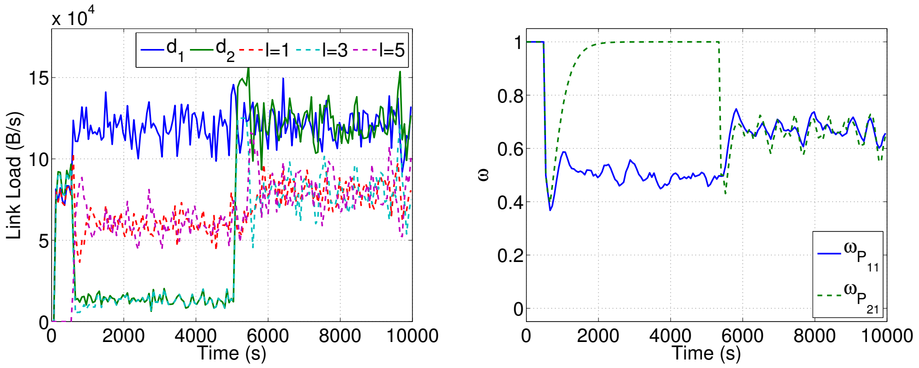


Figure 8.16: The example topology

Just like before, the demand vector is updated every 60 seconds, although commodities are not coordinated and update their probabilities at different moments (with a gap of 25 s). The amount of traffic sent along path P will be noted as ω_P. At the beginning of the simulation we fix the demand vectors, so as to reach a steady state, after which we start updating the ω_P's. The simulation is divided in two parts. Firstly, commodity 2 sends very little traffic (approximately 10 kB/s), while commodity 2 sends ten times this amount. In the second half of the simulation, commodity 2 abruptly starts sending the same amount of traffic as commodity 1 (see Fig. 8.17(a)).

In Fig. 8.17(b) we may see the evolution of the ω_P's with time. Firstly, both $\omega_{P_{11}}$ and $\omega_{P_{21}}$ are initiated at 0.5 automatically by iAWM-R. In the first iterations, both commodities see that the direct path is much less loaded than the indirect one, and increase the corresponding ω_P. However, commodity 1 rapidly realizes that it cannot use its direct path exclusively, and since the cost of the indirect one has decrease due to



(a) Links load as a function of time

(b) ω_P as a function of time

Figure 8.17: The example topology, and the traffic distribution and links' load as a function of time

the reaction of commodity 2, its $\omega_{P_{11}}$ converges to 0.5. In the mean-time, commodity 2 has converged to a situation in which it uses the direct path exclusively. However, in the second part, and analogously to commodity 1 in the first one, this path only is not enough for its traffic demand. Furthermore, the sudden increase triggers a restart in the iAWM-R of commodity 2, after which both $\omega_{P_{11}}$ and $\omega_{P_{21}}$ rapidly converge to 2/3. Notice that, like in the previous example, load measurements need not be very precise, and that the algorithm supports some noise.

8.5 Robust Routing vs Dynamic Load-Balancing

8.5.1 An Introduction to Robust Routing

In this section we interest ourselves to the performance of Dynamic Load-Balancing (DLB, in this case iAWM-R) when compared with a stable routing configuration, in particular Robust Routing (RR), which we presented in Sec. 4.2.1. As mentioned before, DLB and RR may be seen as antagonist approaches to face the ever increasing uncertainty and unpredictability of the traffic injected into the network. On the one hand, Robust Routing copes with traffic uncertainty in an off-line preemptive fashion, computing a stable routing configuration that is optimized for a large set of possible traffic demands. On the other hand, Dynamic Load-Balancing balances traffic among multiple paths in an on-line reactive fashion, adapting to traffic variations in order to optimize a certain cost-function.

Much has been said and discussed about the advantages and drawbacks of each approach, but very few works have tried to compare the performance of both mechanisms, particularly in the same network and traffic scenarios. In fact, to the best of our knowledge the only paper that performs a similar analysis is [49]. In this work, authors compare the performance of their proposal, COPE, with a dynamic approach which they claim models the behavior of mechanisms such as MATE [4] and TeXCP [5] (cf. Sec. 4.3). Given a time-series of traffic demands, this dynamic approach consists of computing an optimal routing for each traffic demand i and evaluate its performance with the following traffic demand $i+1$. There are two important shortcomings of this DLB simulation. Firstly, adaptation in DLB is iterative and never instantaneous. Secondly, in all DLB mechanisms paths are set a priori and remain unchanged during operation. This is not the case in their dynamic approach, where each new routing optimization may change not only traffic portions but paths themselves. For these reasons, we believe that the comparison provided in [49] is unrealistic with respect to DLB, and thus biased against dynamic schemes.

Before presenting our comparative study, we will present a more detailed description of RR in general, and of the considered scheme in particular. In a robust perspective of TE, demand uncertainty is preemptively taken into account within the routing optimization, computing a single routing configuration for all demands within some *uncertainty set*. In particular, we will consider a polyhedral uncertainty set $\mathbb{X}$, more precisely a *polytope* as in [8], based on the intersection of several half-spaces that result from linear constraints imposed to traffic demand. Before presenting an example, let us introduce the last necessary piece of notation. As before, we will note as $\omega_{P_{si}}$ the portion of d_s sent along path P_{si}. Moreover, the *Routing Matrix* $R = (r_{ls})$ is defined as $r_{ls} = \sum_{P \in \mathcal{P}_s} \omega_P \mathbf{1}_{\{l \in P\}}$ (i.e. r_{ls} indicates the fraction of traffic from commodity s routed through link l). This matrix simplifies the calculation of the load on link l, since $\rho = RX$ (where $\rho = (\rho_l)$ and $X = (d_s)$).

As an example of an uncertainty set, let us define $\mathbb{X}$ based on a given routing matrix R_o and the peak-hour links traffic load ρ^{peak} obtained with this routing matrix:

$$\mathbb{X} \;\; = \;\; \left\{ X \in \mathbb{R}^S, R_o.X \leqslant \rho^{peak}, \; X \geqslant 0 \right\}$$

This definition of the uncertainty set has a major advantage: routing optimization can be performed from easily available links traffic load ρ without even knowing the actual value of traffic demand X.

The now traditional *Robust Routing Optimization Problem* (RROP) [8] consists on minimizing the maximum link utilization $u_{\max}$, considering all demands within polytope $\mathbb{X}$ (see the explicit problem below). The solution to the problem is twofold; on the one hand, a routing configuration $R_{robust} = \underset{R}{\operatorname{argmin}} \max_{X \in \mathbb{X}} u_{\max}(X,R)$ and on the other hand, a worst-case performance threshold $u_{\max}^{robust} = \max_{X \in \mathbb{X}} u_{\max}(X,R)$. Given a proper definition of the uncertainty set, the obtained robust routing configuration is applied during long-term periods of time; in this sense, we refer to robust routing as *Stable Robust Routing* (SRR).

$$\underset{R}{\text{minimize}} \; u_{\max} \tag{8.4}$$

$$\text{s.t.} \;\; \sum_{s=1}^{S} \sum_{P \in \mathcal{P}_s} \mathbf{1}_{\{l \in P\}} \omega_P d_s \leq u_{\max} c_l \quad \forall l = 1, \ldots, L \;\; \forall X \in \mathbb{X};$$

$$\sum_{P \in \mathcal{P}_s} \omega_P = 1 \quad \forall s = 1, \dots, S; \qquad \omega_P \geq 0 \quad \forall P \in \mathcal{P}_s \ \forall s = 1, \dots, S$$

The authors of [8] have shown that RROP can be efficiently solved by linear programming techniques, applying a combined columns and constraints generation method. This method iteratively solves the problem, progressively adding new constraints and new columns to the problem. The new constraints are the extreme points of the uncertainty set $\mathbb{X}$, and the new columns represent new paths added to reduce the objective function value. Only extreme points of $\mathbb{X}$ are added as new constraints, as it is easy to see that every traffic demand $X \in \mathbb{X}$ can be expressed as a linear combination of these extreme demands.

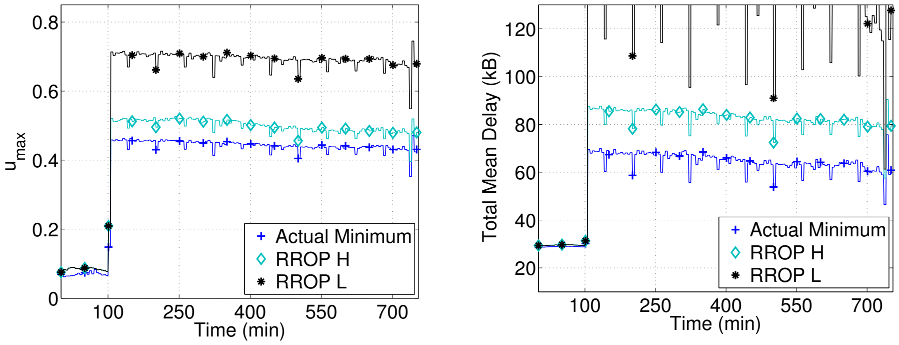
Regarding new added paths, the algorithm in [8] may not be the best choice from a practical point of view since the number of paths for each OD pair is not a priori restricted and the characteristics of added paths are not controlled. For example, and in order to improve resilience, it would be interesting to have disjoint paths to route traffic from each single OD pair. For this reason, the authors of [23] suggested a modification to the path selecting algorithm, both limiting the maximum number of paths in $\mathcal{P}_s$ and taking as new candidates the shortest paths with respect to link weights $w_l(i) = (\epsilon + (1 - r_{ls}(i)))^{-1}$. In this case, $r_{ls}(i)$ corresponds to the fraction of traffic d_s that traverses link l after iteration i and ϵ is a small constant that avoids numerical problems. If OD pair s uses a single path P, $r_{ls} = 1$ for every link $l \in P$, and so this path is removed from the graph where new shortest paths are computed ($w_l \to \infty$, $\forall l \in P$). While this may result in a sub-optimal performance, it allows a real and practical implementation. In case there are no disjoint paths for OD pair s, we use the column constraint generation method used in [8] to add new paths for OD pair s.

8.5.2 Improving Stable Robust Routing

An illustrative Example

In this subsection we will present a first simulation that will help us gain insight into RR and highlight some of its shortcomings. The topology is exactly the same as in Sec. 8.2.1, i.e. Abilene with the demands obtained from [94]. The polytope $\mathbb{X}$ is computed based on the historical routing matrix of Abilene R_o: $\mathbb{X} = \left\{ X \in \mathbb{R}^S, R_o.X \leqslant \rho^{max},\ X \geqslant 0 \right\}$, where ρ^{max} contains the maximum link load values observed at normal operation during the evaluation period. R_o is available at [94].

Figure 8.18(a) shows the results for the maximum link utilization $u_{\max}$. The considered demands are the TMs with indexes between 1050 and 1200 from dataset X23 in [94]. The evaluation starts with a normal traffic pattern, where $u_{\max}$ does not even exceed 0.10. At the 100th minute one of the OD pairs abruptly starts generating an anomalous amount of traffic. We consider two different definitions for polytope $\mathbb{X}$: a first polytope adapted to low load traffic, i.e. traffic in normal operation before the 100th minute, and a second polytope adapted to high load traffic, i.e. traffic after the occurrence of the anomaly. In the sequel we refer to SRR as RROP, recalling that the optimization problem is (8.4). In this sense, we shall use the term RROP L for the SRR adapted to normal operation traffic and RROP H for the SRR adapted to high load anomalous traffic. As may be seen in Fig. 8.18(a), the difference between RROP H and RROP L in the obtained results is significant. In the first case, the anomalous traffic belongs to the uncertainty set and results are only 0.05 higher than the optimum. On the other hand, RROP L performs quite bad during the anomalous event, a result somehow expected given the definition of the polytope. Moreover, and as it may be seen in Fig. 8.18(b), RROP L obtains disastrous results in terms of the total mean delay ($D(d) = \sum f_l(\rho_l)$). Although results for RROP H are somewhat better, its difference with respect to the optimum is still significant (approximately 25%).



(a) Maximum link utilization (b) Total Mean Delay

Figure 8.18: Maximum link utilization and total mean delay over time.

Robust Routing AOF Problem

The SRR approach presents two important problems, one related to the objective function it intends to minimize and the other one as an inherent consequence of using a single static routing configuration. Let us begin by the objective function problem. As we stated in Sec. 4.2.1 and verified in Sec. 8.2, the minimization of $u_{\max}$ may often lead to a worse distribution of traffic, adversely affecting for instance the total mean delay ($D(d)$). This is exactly the case in Fig. 8.18(b) for RROP H. Using $D(d)$ as the objective function in RROP results in a difficult optimization problem, as $f_l(\rho_l)$ is not a linear function. Instead, we could use the total network link utilization $u_{\text{tot}} = \sum_l u_l$ as the objective function. However, simply minimizing u_{tot} results in an unbounded value of $u_{\max}$, which is not practical from an operational point of view.

An alternative approach is to minimize both the value of $u_{\max}$ and u_{tot} at the same time, which constitutes a multi-objective optimization problem. The difficulty that this kind of problem presents is that traditional single-objective optimization techniques cannot be directly applied. An intuitive and easy approach to address this issue is to construct a single aggregated objective function (AOF) that combines both objective functions. We define a weighted linear combination of $u_{\max}$ and u_{tot} as the new objective function $u_{\text{aof}} = \alpha \,.\, u_{\max} + (1-\alpha) \,.\, u_{\text{tot}}$, where $0 \leqslant \alpha \leqslant 1$ is the combination fraction. Equation (8.5) presents the resulting optimization problem, which we shall note as Robust Routing AOF Problem (RRAP). The problem is solved using the same recursive algorithm as in RROP. As we show in the results from the following section, the optimum of this problem achieves better global performance.

$$\begin{aligned} &\underset{R}{\text{minimize}}\; u_{\text{aof}} = \alpha u_{\max} + (1-\alpha) u_{\text{tot}} \\ &\text{s.t. } \sum_{l=1}^{L}\sum_{s=1}^{S}\sum_{P\in\mathcal{P}_s} \mathbf{1}_{\{l\in P\}} \omega_P \frac{d_s}{c_l} \le u_{\text{tot}} \quad \forall X \in \mathbb{X}; \\ &\sum_{s=1}^{S}\sum_{P\in\mathcal{P}_s} \mathbf{1}_{\{l\in P\}} \omega_P d_s \le u_{\max} c_l \quad \forall l = 1,\dots,L \;\; \forall X \in \mathbb{X}; \\ &\sum_{P\in\mathcal{P}_s} \omega_P = 1 \quad \forall s = 1,\dots,S; \qquad \omega_P \ge 0 \quad \forall P \in \mathcal{P}_s \;\; \forall s = 1,\dots,S \end{aligned} \tag{8.5}$$

As we illustrated in Fig. 8.18, the other important drawback of SRR is its inherent dependence on the definition of the uncertainty set. The authors of [23] proposed an adaptive version of SRR, known as Reactive Robust Routing (RRR). The basic idea in RRR consists of computing a set of paths $\mathcal{P}_s$ and a robust routing configuration R^o_{robust} for expected traffic in nominal operation. Additionally, and using this set of paths, a routing configuration R^j_{robust} is computed for every single anomalous traffic event A_s, $s = 1, .., S$, where A_s represents a large volume anomaly in traffic from OD pair s. Using a single anomaly detection/localization sequential algorithm, RRR balances traffic among paths from $\mathcal{P}_s$ according to R^s_{robust} when an anomaly of type A_s is detected and localized. This way they avoid to define an uncertainty set that encompasses all possible anomalous traffic. We refer the reader to [23] for details on the implementation of RRR.

8.5.3 Evaluation and Discussion

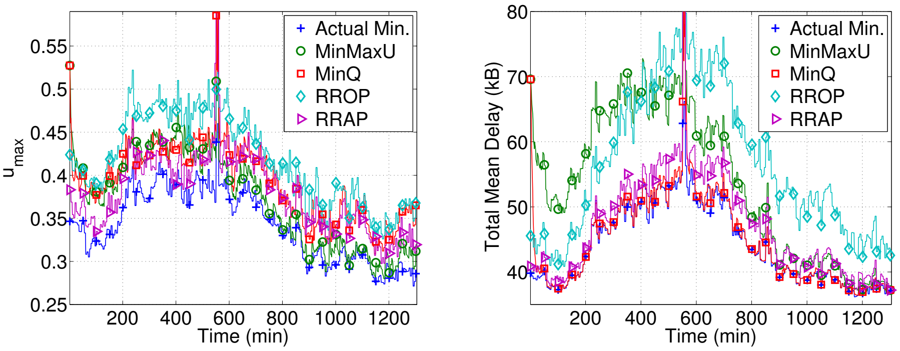
In this section we evaluate the performance of iAWM-R and the different RR algorithms presented before (in particular, for RRAP we used $\alpha = 0.5$), considering both normal operation and anomalous traffic situations. Regarding iAWM-R, we will concentrate on MinQ and MinMaxU, and use this evaluation to present some of their aspects we have not discussed yet. For MinQ, we are interested in the effects on the convergence of the approximations we used for the derivative function $\widehat{\phi}^*_l(\rho_l)$: the linear approximation (cf. (6.7) in Sec. 6.5), and the addition of the corrective $\psi_l(\rho_l)$ (cf. (8.3) in Sec. 8.4.1). For MinMaxU we will present an example in which the WE is suboptimal, and analyze a possible solution to this issue. To be as fair as possible, all mechanisms use the same set of paths, namely those calculated by SRR as discussed in Sec. 8.5.1.

We present and discuss three simulation case-scenarios: starting from a normal traffic variation scenario, we increase the number of OD pairs that present anomalous traffic variations. This allows for performance comparison at different levels of traffic variability. The considered network is again Abilene, and the traffic demands were obtained from [94]. Finally, the methodology for DLB is the same as in Sec. 7.3 (i.e. TMs are fed to the commodities in order, and they perform five iterations of iAWM-R per TM).

Normal Operation

The first case-scenario is the simplest one, which corresponds to traffic in normal operation. The only variability is due to typical daily fluctuations. Fig. 8.19 shows the evolution over time of $u_{\max}$ and $D(d)$ for the different mechanisms, when fed with 260 TMs from set X01 in [94]. All algorithms perform similarly as regards maximum link utilization, depicted in Fig. 8.19(a). This may be further appreciated in Fig. 8.20(a), where we present a boxplot of the difference in $u_{\max}$ with respect to the optimum for all the mechanisms. Naturally, the algorithm that performs best is MinMaxU, although its improvement over the rest is not significant.

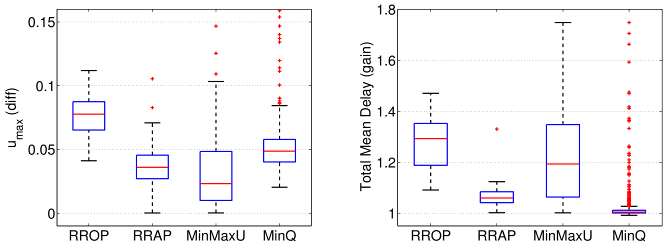
Results with respect to $D(d)$ are quiet different, as it can be seen in Fig. 8.19(b) and 8.20(b); the latter shows the $D(d)$ obtained by each mechanism divided by the corresponding actual minimum. We may verify that the best results are obtained by MinQ, followed closely by both RRAP and MinMaxU (note that the poor results obtained by MinMaxU are mainly from the beginning of the simulation, where it has not yet converged). However, RROP systematically obtains a significant difference with respect to the optimum, generally of about 30%. These results further highlight the limitations of RROP we previously discussed: using only $u_{\max}$ as a performance objective results in a relatively low maximum utilization, but neglects the rest of the links, impacting global performance. Finally, we may verify that in this case the influence of the approximations of $\widehat{\phi}_l(\rho_l)$ on $D(d)$ is insignificant.



(a) Maximum link utilization

(b) Total mean delay

Figure 8.19: Maximum link utilization and total mean delay under normal operation.

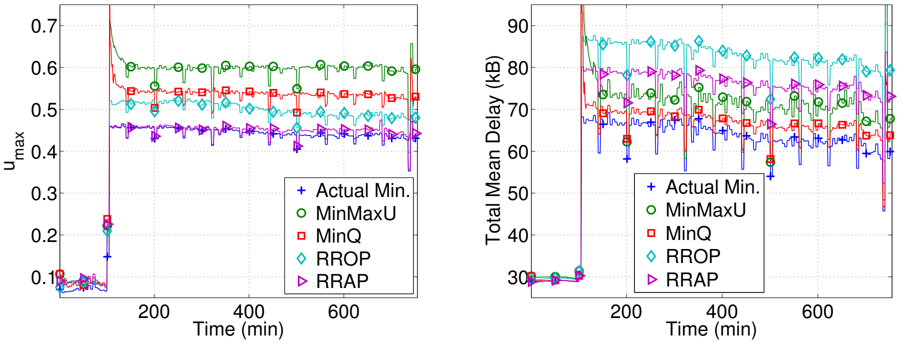


(a) Maximum link utilization (b) Total mean delay

Figure 8.20: Results overview for the first simulation case-scenario - normal operation.

One Anomalous OD Pair

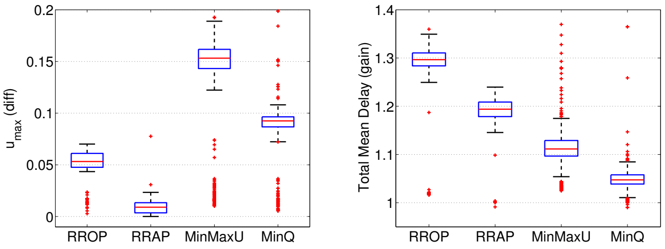
The second case-scenario is the one considered in Sec. 8.5.2, whose main characteristic was the sudden and abrupt increase of traffic generated by one OD pair. To be fair with the DLB mechanisms, both RRAP and RROP use the RRR mechanism described in Sec. 8.5.2 to adapt traffic balancing after the detection of the anomalous traffic variation. Regarding $u_{\max}$, Fig. 8.21(a) and 8.22(a) show that the results obtained by RRAP are quite good and close to the optimum. Surprisingly enough, the results for RROP are somewhat worse, although the difference is not significant. This difference is due to the fact that both RROP and RRAP are solved numerically and not exactly in order to reduce computation time. Results for $D(d)$ follow the same tendency as in the normal operation case, although RRAP performs worse than before. Moreover, although more important than in the previous case, the difference between MinQ using the approximations on $\widehat{\phi}_l(\rho_l)$ and the actual optimum is relatively small (generally less than 5%).



(a) Maximum link utilization (b) Total mean delay

Figure 8.21: Maximum link utilization and total mean delay under abrupt and large traffic variations.

The most interesting aspect of this example are the results obtained by MinMaxU, which does not converge to the optimum, and obtains the worst performance of all the



(a) Maximum link utilization (b) Total mean delay

Figure 8.22: Results overview for the second simulation case-scenario - abrupt traffic variation.

considered schemes (a difference of 15% with respect to the optimum). The reason behind this poor performance is simply that MinMaxU does not take into account the result regarding the optimality of the WE and the *efficiency condition* discussed in Sec. 7.1 and originally presented in [90]. This result states that if at a WE the commodities send their traffic along paths with a minimum number of network bottleneck links (those with the maximum utilization in the whole network), the WE is optimal. The problem we analyze now is how to design a path cost function ϕ_P that takes into account this condition, so that when using it, the load-balancing algorithm converges, when possible, to the correct WE. Note that the condition is only sufficient, meaning that a WE that fulfills the efficiency condition may not exist. A simple example of such case is a single commodity with two paths with different lengths, where all links have the same capacity. Anyhow, the two main difficulties in the design of such path cost are the following. Firstly, the number of bottleneck links in a path is an integer (thus not continuous on $d_{P_{si}}$). Secondly, the probability of two links having exactly the same utilization is zero, and as such we should consider the number of links that have an utilization *similar* to the network bottleneck.

The objective is then to find a cost function that penalizes paths in which several links have similar utilizations (and that this utilization is the maximum in all the network), and that it does not switch between values to avoid oscillations. A candidate ϕ_P that fulfills these two conditions is the so-called *log-sum-exp* function. Consider a set of arbitrary numbers $A = \{a_i\}$, the log-sum-exp function $g(A)$ is defined as follows:

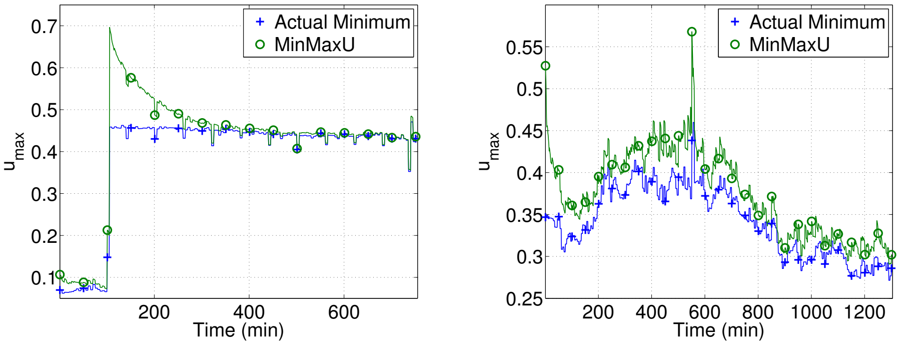
$$g(A) = \frac{1}{\gamma_A} \log\left(\sum_{i=1,\dots,|A|} e^{\gamma_A a_i} \right) = a_{i^*} + \frac{1}{\gamma_A} \log\left(1 + \sum_{\substack{i=1,\dots,|A| \\ i \neq i^*}} e^{\gamma_A (a_i - a_{i^*})} \right) \tag{8.6}$$

Consider the special case in which $a_{i^*} = \max A$. It should be clear that if a_{i^*} is significantly bigger than the rest of the elements in A, the above convex and non-decreasing function constitutes an excellent approximation of a_{i^*}. In fact, it easy to prove that $a_{i^*} \leq g(A) \leq a_{i^*} + \log(|A|)/\gamma_A$, meaning that we may control the precision of the approximation through the parameter γ_A (the bigger this parameter, the more precise the resulting approximation). Moreover, as more elements in A are similar to the maximum, $g(A)$ approaches the upper bound, reaching it when all elements are the same.

We will then use the second term of (8.6) as a penalty to those paths with several links whose utilization is similar to $u_{\max}$ (the maximum utilization in the network). More precisely, given a path P, let $U_P = \{u_l\}_{l \in P}$ be the utilizations in the path, and $l^* \in P$ be the link with the biggest utilization in P. We will then use the penalty function with the alternative set U_P^*, which has the same elements as U_P, but substitutes u_{l^*} by $u_{\max}$. This results in the following cost function:

$$\phi_P = u_{l^*} + \frac{1}{\gamma_P} \log \left(1 + \sum_{\substack{l \in P \\ l \neq l^*}} e^{\gamma_P (u_l - u_{\max})} \right) \tag{8.7}$$

Even if this new cost function penalizes paths with several network bottleneck links, it also penalizes longer paths, which although it may seem as a positive effect, it was not our original objective. A good choice of γ_P will alleviate this side-effect. For instance, we used $\gamma_P = \log(|P|)/\max\{0.01, u_{l^*}/10\}$. This way, we try to minimize the effect of $log(|P|)$ and relativize the penalization to u_{l^*}.



(a) Second example

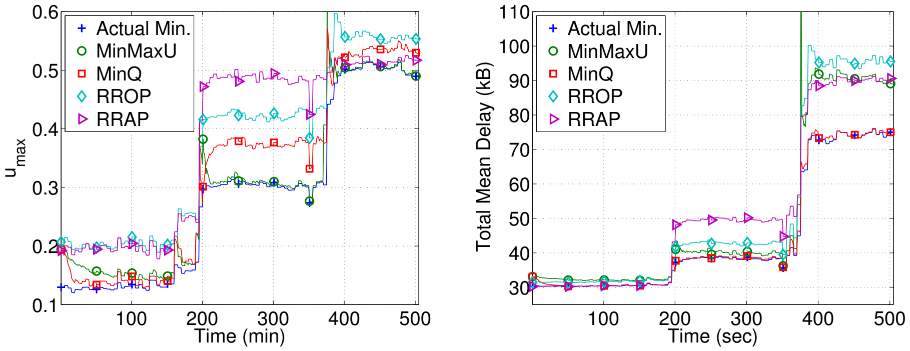
(b) First example

Figure 8.23: Maximum Link Utilization for the first and second simulation case-scenario using the alternative path cost function.

Figure 8.23 shows the results for this new cost function in this and the previous example. As shown in Fig. 8.23(a), the results on the second example are exactly as we intended to. Notice how iAWM-R now converges to the correct WE, effectively minimizing $u_{\max}$. Results for the first example, shown in Fig. 8.23(b), illustrate how the new cost function may influence negatively the obtained maximum utilization. Due to the condition only being sufficient, and the penalization of longer paths, we obtain a slight difference (less than 3% in most of the cases) with respect to the actual minimum $u_{\max}$. We believe that such small loss is largely justified by the gain obtained in the first example.

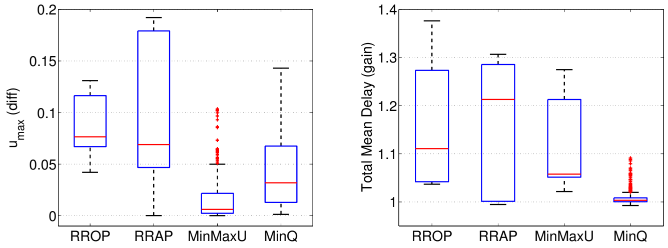
Two Anomalous OD Pairs

In this case-scenario, obtained from dataset X06 of [94], two OD pairs largely increase their traffic demand, one at approximately the 150th minute and the other at the 320th. They both present this anomalous traffic until the end of the simulation. We shall then separate the simulation in three parts: the first third where traffic is normal, the second third were only one OD pair is anomalous, and the last third were both OD pairs are anomalous. The anomaly localization algorithm of RRR was designed for the case of one single anomalous OD pair. Because of this, we will further illustrate the tradeoff between size of the considered uncertainty set and efficiency of the obtained routing, and chose the uncertainty polytope by the traffic loads seen after the second anomaly.



(a) Maximum link utilization (b) Total mean delay

Figure 8.24: Maximum link utilization and total mean delay under gradual and large traffic variations.



(a) Maximum link utilization (b) Total mean delay

Figure 8.25: Results overview for the third simulation case-scenario - gradual traffic variation.

In Fig. 8.24(a) we may see that, as expected, the $u_{\max}$ obtained by both RROP and RRAP in the last third of the simulation are very close to the optimum. However, in the rest of the simulation the difference may be important, specially in the second part where the difference for RRAP is almost 0.2. It is important to highlight the results obtained by MinQ and MinMaxU. Notice that the overshoot this time is much smaller than in the previous example (a maximum of 0.1 in $u_{\max}$ for both MinMaxU and MinQ) and the settling time is negligible. In this case, the increase in traffic of the anomalous OD pairs is more gradual than before, which clearly favors dynamic schemes in their performance. In this case we used the new cost function for MinMaxU, which again has a negligible effect on the obtained $u_{\max}$. The same can be said about the approximative $\widehat{\phi_l^*}(\rho_l)$ used for MinQ and the obtained $D(d)$.

8.5.4 Conclusions

Several conclusions may be drawn from the simulations we just presented, some of which we will now highlight. Firstly, and most importantly, we haven shown that using a single static routing configuration is not a satisfactory solution when the traffic presents a certain level of uncertainty. It could either lead to wasted resources when too many TMs are considered (cf. Fig. 8.24), or a very poor performance when considering too few and facing an unforeseen TM (cf. Fig. 8.18).

It is clear then that some way of adaptability should be implemented in the network, which could either by Reactive Robust Routing (RRR) or DLB. In any case, note that if the anomaly may not be detected correctly (as in the third example in Sec. 8.5.3), the only viable solution is DLB. In this sense, we have verified that iAWM-R is not only stable, but shows a remarkable reactivity to adapt to sudden changes in the demand vector (specially when considering MinQ).

Regarding MinMaxU in particular, we have presented an example in which not considering the efficiency condition discussed in Sec. 7.1 leads to a very poor performance. We have then proposed an alternative path cost function that penalizes those paths that have several network bottlenecks. Preliminary results presented in the previous section are very promising and encourage us to further study this new cost function (e.g. characterize the resulting WE or consider other alternative penalization functions).

Chapter 9

Conclusions and Future Work

In this thesis we have studied new and effective Traffic Engineering (TE) techniques. In particular, we have made special emphasis in what is known as *Dynamic Load-Balancing* (DLB). Let us recall that the idea in DLB is to exploit path diversity to increase resilience and obtain a better performance out of the available resources. More to the point, in DLB each Origin-Destination (OD) pair is connected by several paths, and the objective is to dynamically distribute traffic among these paths depending on the current demand and network condition.

Motivation to study this kind of mechanisms came mainly from two problems network operators are facing today, and that risk to worsen with time. Firstly, and due to service convergence, traffic injected to the network is increasingly dynamic and unpredictable. Indeed, we have shown real traffic demands in which abrupt increases of an order of magnitude are not rare. The possibility of unexpected events such as link/node failures, network attacks, flash crowds or external routing modifications only exacerbate this problem. Secondly, simple overprovisioning is being reconsidered as a valid solution for all problems. This is due to the ever increasing access rates (which has an influence on the dynamic nature of traffic too) and to the emergence of new network architectures in which link capacities are intrinsically limited (e.g. Wireless Networks).

Our study was separated in two parts, depending on the assumed architecture. The first part concentrated on the case in which resources are reserved for paths. The typical way of managing the tunnels in this kind of architectures is based on token buckets. This *a priori* descriptor of traffic has proved inadequate in the past, leading to a systematic overestimation of its parameters to obtain a certain conformity level. We proposed instead to use *Cross-Protect*, a flow-aware networking TE mechanism. In cross-protect, streaming flows obtain head-of-the-queue priority, while elastic flows fairly share the remaining capacity. This way, each kind of traffic obtain the QoS they are interested in. The advantage of Cross-Protect with respect to, for instance, DiffServ, is that no packet marking is required. Instead, flows are classified implicitly based on their rate. A flow transmitting at less than the current fair-rate (which is detected by the flow not being backlogged) is classified as streaming, and viceversa. Moreover, to guarantee a minimum QoS to ongoing flows and to assure the scalability of the scheduler, an admission control mechanism is used. If the amount of streaming traffic (called Priority Load) is more than a certain threshold, or if the fair-rate is less than another one, no more flows are accepted. Note that Cross-Protect provides precise and predictable QoS guarantees, which are easily mapped to the only two parameters of the mechanism. Moreover, and since resources are reserved for each tunnel, the mechanism needs only be implemented in the border routers.

We have modeled a cross-protect router, and derived a close formula for the only remaining QoS parameter: the flow blocking probability generated by the admission control. Moreover, we have studied how to define a dynamic load-balancing scheme in this case. Since the priority load and the fair rate are already measured for admission control purposes, it was only natural to use them to choose the path along which new flows are to be routed. In particular, we have defined *Simple Greedy Policy* (SGP), a DLB mechanism where new flows are routed along the path with the biggest fair-rate. This way, we maximize performance for the majority of traffic, i.e. elastic flows. Moreover, the blocking probability of this scheme is orders of magnitude smaller than the one obtained by classical load-balancing schemes. We have modeled SGP, and could derive tight upper-bounds to its blocking probability. The whole analysis has been verified by means of flow and packet-level simulations.

In the second part of the thesis we have studied the more general "shared resources" case. The implementation of Cross-Protect in such case was somewhat problematic, and as such we did not consider any scheduling mechanism in particular. Instead, we analyze possible DLB schemes that rely on measurements that are available in most current routers (e.g. link load). In particular, we have concentrated on two objective functions: Utility Maximization Load-Balancing (MaxU, as in Maximum Utility, introduced by us in this thesis) and Minimum Congestion Load-Balancing (MinQ, as in Minimum Queue).

MaxU was designed with elastic flows in mind, with the idea that TCP is in charge of the path's resource sharing, while load-balancing further maximizes utility by indirectly changing flow's obtained rate through their path choice. The objective in MinQ is to minimize the total congestion in the network, measured by $\sum_l f_l(\rho_l)$. The idea is that $f_l(\rho_l)$ measures the congestion on link l, for which we have chosen the mean queue size. However, most DLB mechanisms, and ours is not the exception, require an analytical expression for $f_l(\rho_l)$. Instead of assuming an arbitrary model, like most prior works do, we have studied the possibility of *learning* this function from measurements, thus reflecting the actual congestion on the link. Moreover, certain shape restrictions (e.g. increasing) were to be imposed to $f_l(\rho_l)$. To obtain a function that is as general as possible, but still enforces the restrictions, *Convex Nonparametric Weighted Least Squares* (CNWLS) was presented. This regression method obtains an excellent precision, but presents scalability issues as the number of measurements available increases.

The alternative *Convex Piecewise Linear Fitting* (CPLF) addressed this issue, but at the cost of precision.

The optimum of both objective functions has been characterized as the *Wardrop Equilibrium* (WE) of a certain path cost ϕ_P. That is to say, the equilibrium resulting from OD pairs that greedily strive to minimize their path cost. To achieve the WE in a distributed fashion, we have analyzed the possibility of using so-called *no-regret* algorithms, in particular *Incrementally Adaptive Weighted Majority with Restart* Algorithm (iAWM-R). The advantage of iAWM-R with respect to prior distributed optimization algorithms is that it is completely self-regulated. For instance, its convergence speed is automatically set depending on previously observed path costs. The stability and convergence speed of the algorithm have been verified by means of flow and packet-level simulations. Dynamic approaches as DLB are generally met with reluctancy due to their transient behavior under strong traffic variations. However, we have shown that this transient behavior can be effectively controlled, or at least alleviated, by simple mechanisms.

We have performed a thorough performance comparison of three objective functions. In addition to MaxU and MinQ, we have also considered a WE where the path cost ϕ_P is defined as the maximum link utilization (we have noted this scheme as MinMaxU, as in Minimum Maximum Utilization). From our study, conducted over two real networks along with several real traffic demands, some conclusions can be drawn. Firstly, in what respects the path available bandwidth (ABW), it is always better in MaxU than both MinQ and MinMaxU. More specifically, the improvement over MinQ is generally not very big, although it does obtain significantly better results than MinMaxU. Secondly, results on link utilization are very similar between MaxU and MinMaxU. MinQ obtains similar results in the mean and quantile link utilization. However, due to its reticence to use longer paths, the maximum link utilization may be somewhat bigger in MinQ. In terms of the total mean end-to-end queueing delay $D(d) = \sum_l f_l(\rho_l)$ (which we have called total mean delay), the difference may be substantial. The increase in $D(d)$ incurred by MinMaxU or MaxU may easily exceed the 50%. Moreover, although MaxU generally obtains a smaller difference than MinMaxU, its maximum difference is the worst (more than 100%).

Regarding the regression-based minimum congestion framework above, we have studied its potential gain in terms of the total mean delay when compared with assuming a simplistic model for $f_l(\rho_l)$. DLB (and TE in general) is usually defined in terms of a $f_l(\rho_l)$ that is arbitrarily chosen, as long as it is convex and goes to infinity when load reaches the link capacity. In terms of link utilization and path available bandwidth, we have shown that the exact choice of $f_l(\rho_l)$ may be considered as somewhat unimportant. However, when the objective is minimizing queue sizes, which is a very important performance indicator for both elastic (a big mean queue size indicates several bottlenecked flows) as well as streaming traffic (it indicates bigger delays and jitter), this choice is crucial, and a misspecification can result in significant increases of total mean delay with respect to the optimum. We have also shown that the loss in precision incurred by CPLF in its estimation of $f_l(\rho_l)$ does not impact significantly the obtained total mean delay.

All in all, it seems like MinQ is the most balanced objective function, in the sense that it generally outperforms the rest, and when it does not the difference is not very important. Moreover, once we have characterized the links (i.e. learned $f_l(\rho_l)$), the load-balancing algorithm is exactly as complicated as MinMaxU, considered the most simple one. On the other hand, this is exactly the problem with MaxU. It path-cost function ϕ_P, through which the optimum is obtained, requires to distinguish and measure the traffic generated by each commodity. Moreover, a mechanism so that a link can learn whether it constitutes a path's bottleneck has to be designed. Although such mechanisms may be difficult to implement in a packet switched environment, they may be implemented in an MPLS network (where, for instance, measuring $d_{P_{si}}$ is straightforward).

We have also conducted a comparative study between DLB and so-called *Robust Routing* (RR), from which we may reach several conclusions. The most important is probably that we have shown that using a single and unique routing configuration is not a viable solution when traffic is relatively dynamic. It obtains a very poor performance either when faced with unforseen TMs or when its design tries to consider as many TMs as possible. It is clear from our study that some form of dynamism is necessary, which could be either *Reactive Robust Routing* (RRR) or DLB.

RRR computes a nominal operation routing configuration, and has an alternative routing (using the same paths than in normal operation) for certain possible anomalous situations. In order to detect these anomalous situations, link load measurements have to be gathered. On the other hand, DLB gathers these same measurements but also requires updating load-balancing in a relatively small time-scale. The added complexity is then to distribute these measurements to all ingress routers (instead of a central entity) and updating the load-balancing in real-time. Our results show that the additional complexity involved in DLB may not be justified when the variability (or the anomalies) are not very significant. However, the use of DLB under highly dynamic traffic is very appealing and generally provides better results than RRR. Moreover, if the anomalies may not be correctly detected, the only effective solution is DLB. It should also be noted that RR only considers uncertainty in traffic, and that DLB considers traffic variability and link/node failures, thus making it much more flexible.

Finally, regarding RR in particular and similarly to DLB, we have proved that a local performance criterium such as $u_{\max}$ (maximum link utilization), widely used in current network optimization problems, does not represent a suitable objective function as regards global network performance. The use of $u_{\max}$ together with other performance indicators (such as u_{tot}, the sum of links utilization) through the framework of Multi-Objective Optimization provides interesting results and deserves further analysis.

9.1 Future Work

The presented results encourage us to continue on this line of research. However, much remains to be done. A first, and very important problem, is that the result on convergence to the WE of no-regret algorithms proved in [21] was specifically for congestion routing games. For instance, we have used iAWM-R to converge to the WE of a bottleneck routing game[1]. Although convergence was verified in all the simulations we tried, an actual formal proof is still, to the best of our knowledge, an open and very interesting problem. More so, as it has been proved that in general games, agents repeatedly applying no-regret strategies do not converge to the Nash Equilibrium.

Related to the considerations above, and in order to converge to the optimum in MaxU, we have proposed the path cost ϕ_P defined in (5.10). We have proved that any optimum of the problem was necessary a WE with this path cost. However, we were unable to prove that this condition was also sufficient. Moreover, in this case ϕ_P is not continuous in the demands, a fact that could translate into oscillations. However, we wanted to analyze the performance of the objective function before tackling these complicated problems.

Regarding the regression framework, a possible improvement has to do with the model used when defining $f_l(\rho_l)$. Although, as we saw in Sec. 6.5, the mean queue size can be reasonably modeled with such a function in wired mediums, this is not necessarily true in a wireless medium. Actually, as discussed for instance in [105], the MAC-layer interactions between routers play a significant role in determining the capacity of a link (and thus its queue size). This means that the f of any given link should include the load of all links in its collision domain, and not only itself. A deeper analysis of this non-local model represents interesting future work. It would also be very interesting to perform the regression method over real traces, and not over emulation or simulations. Although we have tried to obtain such data from the routers in our own local network, for reasons more political than technical, we have not yet succeeded. Finally, the two regression methods we presented gave as a result a piecewise linear function. However, iAWM-R requires a continuous differentiable $f_l(\rho_l)$. We addressed this issue by approximating the piecewise constant derivative by a piecewise linear function, and verified that this approximation did not affect the obtained performance significantly. However, it would be interesting to design a regression method that shares the optimality properties of CNWLS, but whose output is a continuous differentiable function.

Let us also highlight that the DLB framework we presented was limited to architectures in which paths can be established (e.g. MPLS). An interesting extension would be to consider pure-IP networks. There are two ideas in this direction that we would like to highlight here. Firstly, in multi-topology routing (see for instance [106]) there are several logical topologies in the same network. Routers have an associated rout-

[1] It should be noted that the authors of [6] apply REPLEX to a bottleneck routing game, even when their algorithm was also designed for congestion routing games.

ing table for each of these topologies, and packets are marked at the ingress so as to identify to which logical topology they belong to. For instance, the implementation of *Selective Load-Balancing* (cf. Sec. 4.2.3) with this technique is relatively straightforward. Adaptation of the DLB problem to this framework would be an important extension of our work. Another interesting possibility is the one presented in [63]. Instead of ECMP (Equal Cost Multi-Path) as the only possible multi-path routing in IP networks, they propose to split traffic with an exponential penalization to longer paths. Even if they require this addition to legacy IP routing, they keep the simplicity of hop-by-hop routing and do not require paths to be established, while still minimizing $\sum_l f_l(\rho_l)$.

We have also limited ourselves to intradomain routing. How to define a Dynamic Load-Balancing scheme in the interdomain case, in which packets may traverse several Autonomous Systems (ASs), is an important, but very complicated, problem. Even the establishment of paths in this case poses several technical and political challenges. However, multi-homing and the advances in the development of, for instance, the Locator/ID Separator Protocol (LISP) [107] could simplify and accelerate the development of such interdomain DLB schemes.

Appendix A

List of Publications

International Conferences

- Pedro Casas, Federico Larroca, Jean-Louis Rougier, and Sandrine Vaton, "Robust Routing vs Dynamic Load-Balancing A Comprehensive Study and New Directions," in proceedings of *7th International Workshop on the Design of Reliable Communication Networks (DRCN 2009).* Washington D.C., USA, October 2009.
- Pedro Casas, Federico Larroca, and Sandrine Vaton, "Robust Routing Mechanisms for Intradomain Traffic Engineering in Dynamic Networks," in proceedings of *IEEE/IFIP 6th Latin American Network Operations and Management Symposium (LANOMS 2009).* Punta del Este, Uruguay, October 2009. Awarded with the **Best Conference Paper Award**
- Federico Larroca and Jean-Louis Rougier, "Robust Regression for Minimum-Delay Load-Balancing," in proceedings of *21st International Teletraffic Congress (ITC 21).* Paris, France, September 2009.
- Federico Larroca and Jean-Louis Rougier, "Routing Games for Traffic Engineering," in proceedings of *IEEE International Conference on Communications (ICC 2009).* Dresden, Germany, June 2009.
- Federico Larroca and Jean-Louis Rougier, "Minimum-Delay Load-Balancing Through Non-Parametric Regression," in proceedings of *IFIP/TC6 NETWORKING 2009.* Aachen, Germany, May 2009.
- Federico Larroca and Jean-Louis Rougier, "A Fair and Dynamic Load-Balancing Mechanism," in proceedings of *International Workshop on Traffic Management and Traffic Engineering for the Future Internet (FITraMEn 08).* Porto, Portugal, December 2008. Selected to be published in *Traffic Management and Traffic Engineering for the Future Internet, First Euro-NF International Workshop,*

FITraMEn 2008, Porto, Portugal, December 11-12, 2008, Revised Selected Papers edited by LNCS, Springer.

- Andrés Ferragut, Daniel Kofman, Federico Larroca, and Sara Oueslati, "Design and analysis of flow aware load balancing mechanisms for multi-service networks," in proceedings of *4th EURO-NGI Conference on Next Generation Internet Networks (NGI 2008)*. Krakow, Poland, April 2008.

- Andrés Ferragut, Daniel Kofman, Federico Larroca, and Sara Oueslati, "Design and analysis of flow aware load balancing mechanisms for multi-service networks - Extended Abstract," presented in *EuroFGI Workshop on IP QoS and Traffic Control*. Lisbon, Portugal, December 2007.

International Journals

- Pedro Casas, Federico Larroca, Jean-Louis Rougier and Sandrine Vaton, "Taming Traffic Dynamics: Analysis and Improvements" submitted for fast-tracking in *Computer Communications (COMCOM) Journal* (Elsevier).

- Federico Larroca and Jean-Louis Rougier, "Minimum delay load-balancing via nonparametric regression and no-regret algorithms," submitted to *IEEE/ACM Transactions on Networking*.

Appendix B

Notation Index

Notation	Meaning
γ_s	Threshold for the streaming queue in Cross-Protect
γ_e	Threshold for the elastic queue in Cross-Protect
C	Capacity of a tunnel or LSP
x_s	Number of active streaming flows in Cross-Protect
λ_s	Streaming flows arrival intensity
r	Rate of each streaming flow
τ_s	Streaming flows' mean duration
d_s	Demand generated by the streaming flows ($d_s = \lambda_s \tau_s r$)
x_e	Number of active elastic flows in Cross-Protect
b_e	Mean elastic flows' size
d_e	Demand generated by the elastic flows ($d_e = \lambda_e b_e$)
N_e^{max}	Maximum possible number of elastic flows in the Cross-Protect queue
N_s^{max}	Maximum possible number of streaming flows in the Cross-Protect queue
B_e	Blocking probability of the elastic queue in Cross-Protect
π_s	Stationary distribution of the streaming queue in Cross-Protect
B	Blocking probability of the Cross-Protect router
B_{obp}	Blocking probability of an optimal balanced policy router
$G = (V, E)$	The graph defining the network
l	The index of a link
ρ_l	Load on link l
ρ	A vector containing the ρ_l's
c_l	Capacity of link l
u_l	The link utilization of l ($u_l = \rho_l / c_l$)
s	The index of an OD pair or commodity
S	The number of OD pairs in the network
o_s	Origin node of OD pair s
q_s	Destination node of OD pair s
P	A generic path

$\mathcal{P}_s$	Set of paths belonging to OD pair s
P_{si}	Any path belonging to OD pair s
n_s	The number of paths available to OD pair s ($n_s = \|\mathcal{P}_s\|$)
ϕ_P	Path cost function of P
$\phi_l(\rho_l)$	Link cost function of l
N_P	Number of flows on path P
x_P	Total rate obtained by the N_P flows on path P
d_s	Traffic demand generated by OD pair s
d_P	Amount of traffic sent along path P
d	The demand vector $d = (d_{P_{si}})$
$U(x)$	The utility function
ABW_l	Link available bandwidth ($ABW_l = c_l - \rho_l$)
ABW_P	Path available bandwidth ($ABW_P = \min\limits_{l \in P} ABW_l$)
$D_l(\rho_l)$	Mean queueing delay on link l
D_P	End-to-End path queueing delay ($D_P = \sum_{l:l \in P} D_l(\rho_l)$)
$D(d)$	Mean end-to-end queueing delay
$f_l(\rho_l)$	Mean queue size on link l
$\{(\rho_i, Y_i)\}_{i=1,..,N}$	Set of N measurements (the training set)
w_i	Weight for the measurement i in the regression problem
(α_i, β_i)	Parameters specifying the line segment i in the piecewise linear approximation
k	Maximum number of segments in CPLF
$N(P)$	Number of network bottlenecks over P
y_s	Outcome in iAWM
$L_{P_{si}}$	Path regret in iAWM
λ_s	Learning rate in iAWM
$\omega_{P_{si}}$	Portion of d_s sent along path P_{si} ($\omega_{P_{si}} = d_{P_{si}}/d_s$)
$\mathbb{X}$	The uncertainty set in Robust Routing
r_{ls}	Fraction of traffic from commodity s routed through link l
R	Routing Matrix ($R = (r_{ls})$)
X	A vector representation of the TM ($X = (d_s)$)
$u_{\max}$	The maximum link utilization
u_{tot}	The total network utilization ($u_{\text{tot}} = \sum_l u_l$)

Bibliography

[1] A. E. I. W. D. Awduche, A. Chiu and X. Xiao, “Overview and Principles of Internet Traffic Engineering,” RFC 3273 (Informational), May 2002. [Online]. Available: http://www.faqs.org/rfcs/rfc3272.html

[2] Cisco Systems, “Global ip traffic forecast and methodology 2006-2011,” White Paper, 2007 - updated 2008.

[3] ——, “The exabyte era,” White Paper, 2007 - updated 2008.

[4] A. Elwalid, C. Jin, S. Low, and I. Widjaja, “MATE: MPLS adaptive traffic engineering,” in *IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOM 2001)*, vol. 3, Anchorage, USA, April 2001, pp. 1300–1309.

[5] S. Kandula, D. Katabi, B. Davie, and A. Charny, “Walking the tightrope: responsive yet stable traffic engineering,” in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM ’05)*, Philadelphia, USA, August 2005, pp. 253–264.

[6] S. Fischer, N. Kammenhuber, and A. Feldmann, “Replex: dynamic traffic engineering based on wardrop routing policies,” in *Proceedings of the 2006 ACM CoNEXT conference (CoNEXT ’06)*, Lisboa, Portugal, December 2006, pp. 1–12.

[7] A. Khanna and J. Zinky, “The revised arpanet routing metric,” *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 45–56, 1989.

[8] W. Ben-Ameur and H. Kerivin, “Routing of uncertain traffic demands,” *Opt. and Eng.*, vol. 6, no. 3, pp. 283–313, September 2005.

[9] D. Applegate and E. Cohen, “Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM ’03)*, Karlsruhe, Germany, August 2003, pp. 313–324.

[10] J. W. Roberts, “Internet Traffic, QoS and Pricing,” *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1389– 1399, September 2004.

[11] S. Oueslati and J. W. Roberts, “A new direction for quality of service: flow-aware networking,” in *Next Generation Internet Networks (NGI 2005)*, Rome, Italy, April 2005, pp. 226– 232.

[12] A. Kortebi, S. Oueslati, and J. W. Roberts, “Cross-protect: implicit service differentiation and admission control,” in *Workshop on High Performance Switching and Routing (HPSR 2004)*, Phoenix, USA, 2004, pp. 56–60.

[13] T. Bonald, L. Massoulié, A. Proutière, and J. Virtamo, “A queueing analysis of max-min fairness, proportional fairness and balanced fairness,” *Queueing Syst. Theory Appl.*, vol. 53, no. 1-2, pp. 65–84, 2006.

[14] T. Bonald and A. Proutière, “On performance bounds for balanced fairness,” *Perform. Eval.*, vol. 55, no. 1-2, pp. 25–50, 2004.

[15] D. P. Bertsekas, *Nonlinear Programming.* Athena Scientific, 1999.

[16] D. Bertsekas and R. Gallager, *Data Networks.* Prentice-Hall, 1992.

[17] T. Kuosmanen, “Representation theorem for convex nonparametric least squares,” *Econometrics Journal*, vol. 11, no. 2, pp. 308–325, July 2008.

[18] A. Magnani and S. P. Boyd, “Convex piecewise-linear fitting,” *Optimization and Engineering*, vol. 10, no. 1, pp. 1–17, March 2009.

[19] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, “A survey on networking games in telecommunications,” *Comput. Oper. Res.*, vol. 33, no. 2, pp. 286–311, 2006.

[20] J. Wardrop, “Some theoretical aspects of road traffic research,” *Proceedings of the Institution of Civil Engineers, Part II*, vol. 1, no. 36, pp. 352–362, 1952.

[21] A. Blum, E. Even-Dar, and K. Ligett, “Routing without regret: on convergence to nash equilibria of regret-minimizing algorithms in routing games,” in *Twenty-Fifth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2006)*, Denver, USA, July 2006, pp. 45–52.

[22] P. Auer, C. Gentile, and A.-G. Austria, “Adaptive and self-confident on-line learning algorithms,” *Journal of Computer and System Sciences*, vol. 64, no. 1, pp. 48–75, February 2002.

[23] P. Casas, L. Fillatre, and S. Vaton, “Robust and reactive traffic engineering for dynamic traffic demands,” in *Next Generation Internet Networks (NGI 2008)*, Krakow, Poland, April 2008, pp. 69–76.

[24] Global Action Plan, “An inefficient truth.” [Online]. Available: http://www.globalactionplan.org.uk/upload/resource/Full-report.pdf

[25] M. Gupta and S. Singh, "Greening of the internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)*, Karlsruhe, Germany, August 2003, pp. 19–26.

[26] L. Kleinrock, *Queueing Systems.* Wiley-Interscience, 1975.

[27] "Metro Ethernet Forum." [Online]. Available: http://www.metroethernetforum.org/

[28] "Provider Bridges." [Online]. Available: http://www.ieee802.org/1/pages/802.1ad.html

[29] "Provider Backbone Bridges." [Online]. Available: http://www.ieee802.org/1/pages/802.1ah.html

[30] "VPLS - Virtual Private LAN Service." [Online]. Available: http://www.vpls.org

[31] G. Carofiglio and L. Muscariello, "On the impact of tcp and per-flow scheduling on internet performance," in *Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM 2010)*, San Diego, USA, March 2010.

[32] T. Bonald, A. Proutiere, and J. W. Roberts, "Statistical Performance Guarantees for Streaming Flows using Expedited Forwarding," in *IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, vol. 2, Anchorage, USA, 2001, pp. 1104–1112.

[33] T. Bonald and J. W. Roberts, "Congestion at flow level and the impact of user behaviour," *Computer Networks*, vol. 42, no. 4, pp. 521–536, 2003.

[34] S. B. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J. W. Roberts, "Statistical bandwidth sharing: A study of congestion at flow level," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01)*, San Diego, USA, 2001, pp. 111–122.

[35] P. Goyal, H. Vin, and H. Cheng, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 690–704, October 1997.

[36] M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round Robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 231–242, June 1996.

[37] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, "Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '05)*, Banff, Canada, 2005, pp. 217–228.

[38] N. Benameur, S. B. Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J. W. Roberts, "Integrated admission control for streaming and elastic traffic," in *Second COST 263 International Workshop on Quality of Future Internet Services (QofIS 2001)*, vol. 2156, Coimbra, Portugal, 2001, pp. 69–81.

[39] "The Network Simulator - ns." [Online]. Available: http://nsnam.isi.edu/nsnam/index.php/Main_Page

[40] F. Bonomi, "On job assignment for a parallel system of processor sharing queues," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 858–869, 1990.

[41] G. Koole and A. Hordijk, "On the Assignment of Customers to Parallel Queues," *Probability in the Engineering and Informational Sciences*, vol. 6, pp. 495 – 511, 1992.

[42] B. Hajek, "Optimal control of two interacting service stations," *IEEE Trans. on Automatic Control*, vol. 29, no. 6, pp. 491– 499, June 1984.

[43] T. Bonald, M. Jonckheere, and A. Proutiere, "Insensitive load balancing," in *Proceedings of the joint international conference on Measurement and modeling of computer systems (SIGMETRICS '04/Performance '04)*, New York, USA, June 2004, pp. 367–377.

[44] S. Agarwal, A. Nucci, and S. Bhattacharyya, "Controlling hot potatoes in intradomain traffic engineering," Sprint ATL, Tech. Rep. RR04-ATL-070677, July 2004.

[45] M. Roughan, M. Thorup, and Y. Zhang, "Traffic engineering with estimated traffic matrices," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC '03)*, Miami Beach, USA, October 2003, pp. 248–258.

[46] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Towsley, "On optimal routing with multiple traffic matrices," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, vol. 1, Miami Beach, USA, March 2005, pp. 607–618.

[47] M. Johansson and A. Gunnar, "Data-driven traffic engineering: techniques, experiences and challenges," in *3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS 2006)*, San José, USA, October 2006, pp. 1–10.

[48] I. Juva, "Robust load balancing," in *IEEE Global Telecommunications Conference (GLOBECOM '07)*, Washington D.C., USA, November 2007, pp. 2708–2713.

[49] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "Cope: traffic engineering in dynamic networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 99–110, 2006.

[50] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe, “Resource management with hoses: point-to-cloud services for virtual private networks,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 679–692, 2002.

[51] A. Gupta, J. M. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, “Provisioning a virtual private network: a network design problem for multicommodity flow,” in *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC ’01)*, Crete, Greece, July 2001, pp. 389–398.

[52] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, “Algorithms for provisioning virtual private networks in the hose model,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 565–578, 2002.

[53] T. Erlebach and M. Ruegg, “Optimal bandwidth reservation in hosemodel vpns with multi-path routing,” in *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, vol. 4, Hong Kong, China, March 2004, pp. 2275–2282.

[54] F. Eisenbrand, F. Grandoni, G. Oriolo, and M. Skutella, “New approaches for virtual private network design,” *SIAM J. Comput.*, vol. 37, no. 3, pp. 706–721, 2007.

[55] A. Juttner, I. Szabo, and A. Szentesi, “On bandwidth efficiency of the hose resource management model in virtual private networks,” in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, vol. 1, San Francisco, USA, March, April 2003, pp. 386–395.

[56] M. Berthelot, D. Pletsch, and H. Ravokatra, “Dimensionnement de réseau à l’aide du modèle de hose,” July 2007, end-of-course project for UE RES 381 in Télécom ParisTech.

[57] R. Zhang-Shen and N. McKeown, “Designing a predictable internet backbone network,” in *Third Workshop on Hot Topics in Networks (HotNets-III)*, San Diego, USA, November 2004.

[58] M. Kodialam, T. Lakshman, and S. Sengupta, “Efficient and robust routing of highly variable traffic,” in *Third Workshop on Hot Topics in Networks (HotNets-III)*, San Diego, USA, November 2004.

[59] R. Prasad, P. J. Winzer, S. C. Borst, and M. K. Thottan, “Queuing delays in randomized load balanced networks,” in *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, Anchorage, USA, May 2007.

[60] L. G. Valiant and G. J. Brebner, “Universal schemes for parallel communication,” in *STOC ’81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, Milwaukee, USA, 1981, pp. 263–277.

[61] R. Zhang-Shen, “Valiant load balancing,” Ph.D. dissertation, Stanford University, Stanford, California, 2007.

[62] F. B. Shepherd and P. J. Winzer, “Selective randomized load balancing and mesh networks with changing demands,” *Journal of Optical Networking*, vol. 5, pp. 320–339, 2006.

[63] D. Xu, M. Chiang, and J. Rexford, “Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering,” in *The 27th Conference on Computer Communications (INFOCOM 2008)*, Phoenix, USA, April 2008, pp. 466–474.

[64] S. Fischer, H. Räcke, and B. Vöcking, “Fast convergence to wardrop equilibria by adaptive sampling methods,” in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (STOC ’06)*, Seattle, USA, May 2006, pp. 653–662.

[65] R. Srikant, *The Mathematics of Internet Congestion Control.* Birkhäuser Boston, 2003.

[66] J. Mo and J. Walrand, “Fair end-to-end window-based congestion control,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 556–567, 2000.

[67] F. Kelly, A. Maulloo, and D. Tan, “Rate control in communication networks: shadow prices, proportional fairness and stability,” *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[68] W.-H. Wang, M. Palaniswami, and S. H. Low, “Optimal flow control and routing in multi-path networks,” *Perform. Eval.*, vol. 52, no. 2-3, pp. 119–132, 2003.

[69] F. Kelly and T. Voice, “Stability of end-to-end algorithms for joint routing and rate control,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 5–12, 2005.

[70] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, “Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1260–1271, 2006.

[71] F. Paganini, “Congestion control with adaptive multipath routing based on optimization,” in *40th Annual Conference on Information Sciences and Systems (CISS ’06)*, Princeton, USA, March 2006.

[72] P. Key, L. Massoulie, and D. Towsley, “Path selection and multipath congestion control,” in *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, Anchorage, USA, May 2007, pp. 143–151.

[73] J. He, M. Bresler, M. Chiang, and J. Rexford, “Towards robust multi-layer traffic engineering: Optimization of congestion control and routing,” *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 5, pp. 868–880, June 2007.

[74] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[75] T. V. Lakshman and U. Madhow, "The performance of tcp/ip for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Netw.*, vol. 5, no. 3, pp. 336–350, 1997.

[76] T. Bonald and A. Proutière, "Insensitive bandwidth sharing in data networks," *Queueing Syst. Theory Appl.*, vol. 44, no. 1, pp. 69–100, 2003.

[77] P. Lassila, A. Penttinen, and J. Virtamo, "Communication networks dimensioning of data networks: a flow-level perspective," *European Transactions on Telecommunications*, to appear.

[78] F. Larroca and J.-L. Rougier, "A fair and dynamic Load-Balancing mechanism," in *International Workshop on Traffic Management and Traffic Engineering for the Future Internet (FITRAMEN) 2008*, Porto, Portugal, December 2008.

[79] A. van den Bos, *Parameter Estimation for Scientists and Engineers.* Wiley-Interscience, 2007.

[80] K. Cho, "WIDE-TRANSIT 150 Megabit Ethernet Trace 2008-03-18." [Online]. Available: http://mawi.wide.ad.jp/mawi/samplepoint-F/20080318/

[81] N. Hohn, D. Veitch, K. Papagiannaki, and C. Diot, "Bridging router performance and queuing theory," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 355–366, 2004.

[82] R. C. Fair, "On the robust estimation of econometric models," *Annals of Economic and Social Measurement*, vol. 3, no. 4, pp. 117–128, October 1974.

[83] "The MOSEK Optimization Software." [Online]. Available: http://www.mosek.com/

[84] L. Wasserman, *All of Nonparametric Statistics: A Concise Course in Nonparametric Statistical Inference.* Springer, 2006.

[85] Y. Ait-Sahalia and J. Duarte, "Nonparametric option pricing under shape restrictions," *Journal of Econometrics*, vol. 116, no. 1-2, pp. 9–47, September-October 2003.

[86] C. Cassandras, M. Abidi, and D. Towsley, "Distributed routing with on-line marginal delay estimation," *IEEE Trans. Comm.*, vol. 38, no. 3, pp. 348–359, Mar 1990.

[87] M. Beckmann, C. B. McGuire, and C. B. Winsten, *Studies in the Economics of Transportation.* Yale University Press, 1956.

[88] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS '99)*, Trier, Germany, March 1999, pp. 404–413.

[89] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden, “The price of stability for network design with fair cost allocation,” in *45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*, Rome, Italy, October 2004, pp. 295–304.

[90] R. Banner and A. Orda, “Bottleneck routing games in communication networks,” *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 6, pp. 1173–1179, August 2007.

[91] V. G. Vovk, “Aggregating strategies,” in *Proceedings of the third annual workshop on Computational learning theory (COLT ’90)*, Rochester, United States, August 1990, pp. 371–386.

[92] N. Littlestone and M. K. Warmuth, “The weighted majority algorithm,” *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, 1994.

[93] “The Abilene Network.” [Online]. Available: http://www.internet2.edu/network/

[94] Yin Zhang, “Abilene Dataset.” [Online]. Available: http://www.cs.utexas.edu/~yzhang/research/AbileneTM/

[95] T. M. Cover, *Mathematical Finance*, vol. 1, no. 1, pp. 1–29, 1991.

[96] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[97] N. Cesa-Bianchi, Y. Mansour, and G. Stoltz, “Improved second-order bounds for prediction with expert advice,” *Mach. Learn.*, vol. 66, no. 2-3, pp. 321–352, 2007.

[98] M. Herbster and M. K. Warmuth, “Tracking the best expert,” *Mach. Learn.*, vol. 32, no. 2, pp. 151–178, 1998.

[99] Z. Cao, Z. Wang, and E. Zegura, “Performance of hashing-based schemes for internet load balancing,” in *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, vol. 1, Tel Aviv, Israel, March 2000, pp. 332–341.

[100] “Géant Topology Map.” [Online]. Available: http://www.geant.net

[101] G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D’Arienzo, O. Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescape, B. Quoitin, S. Romano, E. Salvadori, F. Skivee, H. T. Tran, S. Uhlig, and H. Umit, “An open source traffic engineering toolbox,” *Computer Communications*, vol. 29, no. 5, pp. 593–610, 2006.

[102] “TOTEM: TOolbox for Traffic Engineering Methods.” [Online]. Available: http://totem.info.ucl.ac.be/

[103] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 83–86, 2006.

[104] W.-B. Gong, Y. Liu, V. Misra, and D. Towsley, "Self-similarity and long range dependence on the internet: a second look at the evidence, origins and implications," *Comput. Netw.*, vol. 48, no. 3, pp. 377–399, 2005.

[105] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, vol. 2, San Francisco, USA, March, April 2003, pp. 836–843.

[106] A. Kvalbein and O. Lysne, "How can multi-topology routing be used for intradomain traffic engineering?" in *Proceedings of the 2007 SIGCOMM workshop on Internet network management (INM '07)*, Kyoto, Japan, August 2007, pp. 280–284.

[107] D. Farinacci, V. Fuller, D. Oran, D. Meyer, and S. Brim, "Locator/id separation protocol (lisp)," draft-farinacci-lisp-11, December 2008. [Online]. Available: http://tools.ietf.org/id/draft-farinacci-lisp-11.txt