Distributed resource coallocation: architectures, protocols, optimization

Antoine Pichot

Thèse

présentée pour obtenir le grade de docteur

de l'Ecole Nationale Supérieure des Télécommunications

Spécialité : Informatique et Réseaux

Antoine Pichot

Co-allocation de ressources distribuées: architectures, protocoles, optimisation

Soutenue le 7 Avril 2008 devant le jury composé de

M. Philippe d'Anfray (CEA)	
Prof. Bijan Jabbari (George Mason U.)	Rapporteurs
Dr. Franck Cappello (INRIA Futurs)	Président du Jury
M. Olivier Audouin (Alcatel-Lucent)	Examinateur
Prof. Maurice GAGNAIRE (ENST)	Directeur de thèse

Ecole Nationale Supérieure des Télécommunications

To my Parents

...

Acknowledgements

I would like to thank the Alcatel-Lucent Research&Innovation Packet Transport Infrastructure group for giving me the opportunity to conduct this research with such freedom.

I am grateful to Professor Maurice Gagnaire for having supervised this research. Althought very busy, he answered all of my questions and provided me with his experience and vast knowledge.

I also thank Olivier Audouin for being my tutor at Alcatel-Lucent, who followed this research very closely. The many discussions we had helped me clarify, summarize and move forward with my research.

I would also like to thank my colleagues at Alcatel-Lucent starting with my manager Emmanuel Dotaro with whom I had many interesting discussions. He provided me with guidance not only in technical matters but also in human relationships. I am also grateful to my team members, Agostino Chiosi, Dominique Verchere and Bela Berde for creating such a comfortable work environment. This great teamwork experience has given me the skills to confront any kind of environment.

I would like to express my sincere thanks to Doctor Steven Pickering for being the first to read (twice!), review and correct this document's English. He has done a fantastic job and greatly contributed to the clarity of the final result.

Last, but not least, I would like to deeply thank Matthieu Boulay for his friendship and support. Also my family, my sister Laurence for her support and the so many conversations we had. The authors of the SIP RFC and my brother-in-law Christophe's patience without whom those conversations would never have been possible. My sister Martine, my brother Jeremie and my parents Colette and Jean-Claude for having supported my ups and downs during the last three years.

Résumé

Les applications de calcul intensif nécessitent de plus en plus de ressources. Pour des raisons de sécurité, de consommation d'énergie et de passage à l'échelle, ces dernières ne peuvent plus être localisées au sein d'un même bâtiment. L'utilisation de ressources distribuées est une réalité à laquelle doivent faire face de nombreuses industries et centre de recherche. Par exemple la collaboration entre laboratoires exige l'utilisation de ressources hétérogènes telles des calculateurs et des espaces de stockage géographiquement distribués pour résoudre des problèmes complexes. Ces ressources distribuées bien qu'appartenant à des entités juridiques et administratives différentes doivent être associées logiquement temporairement pour constituer une infrastructure virtuelle afin de résoudre un problème scientifique ou fournir un service donné. Une telle infrastructure virtuelle et le réseau sous jacent est appelée une grille.

Les modèles actuels d'exploitation des ressources telles le modèle "overlay" où les applications exploitent le réseau comme une boite noire sans échange d'information de contrôle et de gestion ne permettent pas de réservation jointe de ces ressources à la demande: réseaux, calculateurs et espaces de stockage. C'est pourquoi une nouvelle génération de plan de contrôle et de gestion doivent être définis pour fournir un service de calcul à la demande avec un niveau de qualité contractuel garanti prenant en compte l'utilisation du réseau et des ressources de calcul.

Le but de cette thèse est de fournir une vision de l'état de l'art en matière de Co-allocation. Dans ce but, différents environnements sont considérés: les systèmes de gestion de ressources distribuées basés sur les Web Services (WS), l'architecture IP Multimedia Subsystem (IMS) et Generalized Multi-protocol Label Switching (GMPLS). Des extensions aux logiciels de gestion des grilles, aux Web Services, à l'IMS, et à GMPLS sont proposées. Elles sont étudiées afin de discuter de leur pertinence, avantages et inconvénients, puis comparées à d'autres alternatives. Une fois les architectures globales décrites, nous étudions plus en détail le coeur du système de Co-allocation: le protocole de communication entre l'ordonnanceur de la grille et les ordonnanceurs locaux. Le protocole de création de contrat de service (Service Level Agreement, SLA) WS-Agreement est étendu afin de négocier dynamiquement l'accès aux ressources. Enfin, des algorithmes modélisant l'utilisation des ressources de calcul et des ressources réseaux sont proposés pour déterminer quel est la meilleure interaction possible entre le gestionnaire de ressource réseau et celui des ressources de calcul. Un algorithme de Co-allocation est proposé pour améliorer l'efficacité du système telle qu'elle est perçue par l'utilisateur. Un modèle analytique est proposé pour prédire et comprendre les performances, des simulations ont permis de vérifier la validité du modèle et des résultats.

Même si cette thèse porte essentiellement sur les ressources de calcul et réseau, il est

possible de généraliser la plupart des résultats du chapitre architectural et protocolaire à d'autres types de ressources.

Abstract

New computing applications require nowadays a physical distribution of computing resources in order to mitigate power consumption, security policies and system scalability problems. For instance, collaborations between research institutes require the use of several heterogeneous computational and storage resources to solve complex problems. These geographically distributed resources belonging to different organizations must be associated logically in order to solve cooperatively a given problem or to provide a given service. The virtual infrastructure corresponding to the set of these distributed and remote resources and to the inherent underlying networking facilities is called a Grid. Present models where applications are overlaid onto networks without any management information exchange do not enable network and other resources such as computing or storage to be co-allocated on demand. Thus, new generation of control plane and of management plane must be defined in order to provide on-demand cooperative computational services with a service level agreement (SLA) including Quality of Service (QoS) guarantees taking into account network and computing resources.

The aim of this thesis is first to provide a review of the state of the art on co-allocation. For that purpose, various environments such as Web Services (WS) distributed resources management systems, IP Multimedia Subsystem (IMS) and Generalized Multi-protocol Label Switching architecture (GMPLS) are considered. We propose extensions to existing Grid toolkits, WS, IMS and GMPLS for dynamic resource co-allocation provisioning. The suitability of each of these approaches for Grid services provisioning is investigated and compared to the other alternatives. We then analyze a WS based protocol between a global resource coordinator (Grid Scheduler) and local resources managers (local schedulers). The WS-Agreement protocol is extended in order to permit dynamic SLA negotiation. Algorithms are proposed to model the possible interactions between the grid scheduler, the network resource manager and the local schedulers. A co-allocation algorithm is proposed to improve the efficiency as seen by the end user and the resource providers. An analytical model is proposed to predict and understand the performance; simulations are run to verify the validity of the model and the results.

Even though this thesis focuses on computational and network resources co-allocation, most of the results presented in the architecture and protocol section can be applied to any kind of resources.

Version Française

Introduction

De nombreux projets [39, 141, 121, 94] comme SETI@HOME [22, 40] nécessitent une puissance de calcul et de stockage phénoménale, allant bien au delà de ce que peuvent fournir les ordinateurs actuels. SETI@HOME analyse les signaux radios venant de toutes les directions visibles de l'espace dans le but de découvrir des signaux émis par une source intelligente extra-terrestre. La conception de super calculateurs pour résoudre de tels problèmes n'est pas une option réaliste tant pour des raisons économiques que techniques: manque d'espace, d'accès électriques, risque de sécurité (incendie). L'usage de centaines ou de milliers de ressources de calcul distribuées est donc une réalité et une nécessité. Ces architectures ainsi formées sont dites virtuelles, en ce sens elles fédèrent des ressources appartenant à des entités administratives différentes. Elles rivalisent de puissance avec les meilleurs super calculateurs au monde [39].

Les projets mentionnés ici, bien que d'une importance particulière sont hors du propos de cette thèse car ils ne permettent pas de fournir de garantie de qualité de service (QoS) aux utilisateurs. Cette thèse porte sur les infrastructures de calcul virtuelles fournissant une qualité de service prenant en compte l'utilisation du réseau et des calculateurs. Les utilisateurs de l'infrastructure peuvent bénéficier d'un contrat de service (Service Level Agreement SLA) garantissant par exemple les temps de calcul en prenant en compte le temps passé dans le réseau et le calculateur. Le mot "Grille" anglicisme issu de "Computational Grid [72, 70, 73]" lui même issu de "Power Grid", c'est-à-dire réseau électrique serait mieux traduit par réseau. Il désigne une fédération de ressources de stockage, de calcul, réseau appartenant à des entités administratives différentes établies dans le but de résoudre un même problème. Ces ressources sont caractérisées par leur hétérogénéité et leur distribution géographique. Elles sont inter connectées par un réseau longue distance. Le concept est né de la volonté de créer une infrastructure de calcul ou de stockage aussi facile à utiliser que celle du réseau électrique. Il suffit de se brancher au réseau pour pouvoir utiliser les ressources sans avoir à se soucier de leur localisation ou caractéristiques techniques à l'instar du réseau électrique: personne ne sait si l'énergie utilisée à un moment donnée est issue d'une centrale nucléaire ou à vapeur. Cette abstraction s'appelle la virtualisation des ressources.

Dans tous les domaines scientifiques et dans toutes les industries, de nombreuses applications nécessitent l'utilisation une forte puissance de calcul. L'émergence de ces besoins est concourante à l'apparition de réseau ultra haut débit et de ces infrastructures de calcul virtuelles. Toutefois de nouveaux modèles commerciaux de fourniture de service sont à

envisager. Le modèle prédominant est la location de puissance de calcul proportionnellement à l'utilisation qui en est faite, par exemple un euro par heure par CPU (Central Processing Unit). De même de nouvelles organisations sont nécessaires pour pouvoir garantir la qualité de service. Cette thèse distingue les acteurs suivants: les propriétaires exploitant la ressource de calcul, ceux exploitant la ressource réseau, les utilisateurs finaux et le fournisseur du service de calcul virtualisé. Le fournisseur du service de calcul virtualisé peut appartenir à la même entité juridique que l'opérateur réseau. L'opérateur réseau jouerait donc un rôle d'intermédiaire entre les consommateurs de ressources et les fournisseurs. Chacun des acteurs exploitera un logiciel (Grid toolkit) ou une partie du logiciel de gestion et d'accès aux ressources de la grille. Le gestionnaire du réseau et celui du ou des calculateurs devront communiquer avec le gestionnaire permettant la fourniture du service de calcul. Bien que différents fournisseurs de ressources existent, ils présentent une interface commune afin de faciliter la gestion de l'accès à ces ressources, c'est l'une des fonctions principale du Grid toolkit.

Les relations commerciales entre ces différents acteurs sont régies par des contrats de fourniture de service (SLA). La description du service, ses caractéristiques, ses contraintes, ses garanties sont négociées par les deux parties: le fournisseur de service et le consommateur. Ces contrats font l'objet d'une littérature abondante [21, 119, 80, 106, 105, 91, 101, 59].

Parmi les différents types de service qui peuvent être fournis, nous étudierons particulièrement les services de connectivité à la demande sous contrainte de bande passante ou latence et un service de calcul intégré, c'est-à-dire intégrant la qualité de service réseau dans l'offre de calcul. La bande passante et latence à la demande sont les deux services réseaux fortement demandés [134] par la communauté développant les infrastructures pour les applications fortement distribuées. La figure 1 à la page 5 illustre le schéma global de fonctionnement d'un tel service et de ses interactions. Dans ce contexte différentes approches [82, 124, 157, 86] d'optimisation cherche à maximiser l'utilisation qui est faite des ressources réseaux seules, par exemple par la minimisation du nombre de requêtes rejetées. Cette thèse porte sur l'étude d'un service de calcul intégré qui prendrait en compte l'optimisation des ressources de calcul et des ressources réseau. Dans ce cadre, de nombreux problèmes sont à résoudre: l'architecture, les protocoles, l'utilisation efficace des ressources, l'hétérogénéité des ressources, le passage à l'échelle, les mécanismes de réalisation de la garantie de QoS, la tarification, etc. Cette thèse porte sur les trois premiers problèmes: l'architecture, les protocoles et l'optimisation. Un des points clés de l'étude porte sur la nature des informations échangées entre l'opérateur réseau et le fournisseur du service de calcul intégré. Chaque gestionnaire de ressources ne souhaite pas diffuser des informations jugées parfois comme stratégiques et confidentielles: la topologie du réseau, l'état de réservation des liens, la localisation des pannes, la charge des processeurs. Ces informations permettent une bien meilleure utilisation des ressources. Nous verrons dans la suite quelles informations peuvent être communiquées et dans quel but.

La structure de la version française reflète l'organisation globale de la thèse. Dans un premier temps une revue synthétique de l'état de l'art et des projets portant sur la co-allocation de ressources est faite au chapitre 1. Les trois architectures et les contributions architecturales de cette thèse sont présentées au chapitre 2. Nous étudions plus en détail au chapitre 3 le mécanisme de co-allocation de ressource et le protocole d'établissement de

SLA. Nous proposons des extensions au protocole d'établissement de SLA afin de mieux négocier l'accès aux ressources puis nous étudions la performance de ces extensions. Enfin, la contribution majeure de cette thèse est décrite au chapitre 4, nous étudions les optimisations possibles grâce à la co-allocation dans trois scénarios d'échanges d'information entre le gestionnaire du réseau et celui de la grille. Nous proposons un modèle analytique permettant de prédire les performances de la grille et nous validons ce modèle et nos prédictions grace à des simulations. Nous concluons ce travail en donnant les pistes pour poursuivre cette recherche qui ouvre les portes d'un domaine peu étudié d'optimisation des ressources de calcul et des ressources réseau: l'optimisation croisée.

Contexte

Bien que des services commerciaux d'exploitation de ressources de calcul existent déjà [1, 2, 23, 4], ils ne permettent pas de réserver à la demande des ressources réseaux et des ressources de calcul. Le temps potentiellement gagné en sélectionnant un ordinateur puissant pour réaliser une tache complexe peut être perdu à cause de réseau qui relie l'utilisateur au calculateur. Nous allons donc décrire dans cette section et au chapitre 1 les technologies qui permettent la gestion des ressources distribuées, les grid toolkits et les projets dans lesquels ces logiciels interagissent avec le réseau pour assurer une certaine qualité de service.

Parmi ces logiciels nous devons distinguer les logiciels libres Globus [69], Unicore [65, 66], G-Lite et les logiciels commerciaux Platform LSF [18], Altair PBS [19], Sun Grid Engine [24], IBM LoadLeveler [26, 89], Datasynapse GridServer [6]. Quasiment tous exploitent fortement les technologies dites Web Services: SOAP [49] ou REST [67]. Les deux toolkits principaux, Globus et Unicore présentent à peu près les mêmes fonctionnalités. La figure 1.6 de la page 24 reprends les différents noms des modèles associés aux fonctionnalités de gestion des tâches, d'ordonnancement, de transfert de données et de sécurité. Ces logiciels ne sont pas encore complètement compatibles et c'est pour l'heure leur principal défaut. Bien que des tentatives de standardisation sont en cours [74], leur évolution reste très lente et l'adoption de ces standards limitée. Pourtant des tests d'interopérabilité dans le cadre de la communauté libre sont réalisés [123, 122] et constituent une première étape vers l'émergence d'un standard. Le chapitre 1 décrit plus en détail les principaux toolkits et l'état de la standardisation.

Dans la section 1.5 (page 24) du chapitre 1, nous passons en revue les nombreux projets exploitant à la fois le réseau et la ressource de calcul: Network Resource Scheduling (NRS) [12], Globus Architecture for Reservation and Allocation (GARA) [7, 71], GARA based DataTAG [57], User Controlled Light Paths (UCLP) [27], Internet2 QBone Bandwidth Broker [20], EGEE Bandwidth Allocation and Reservation (BAR) [32], Dynamic Resource Allocation in Generalized Multi Protocol Label Switching GMPLS optical networks (DRAGON) [99], AkoGrimo [37], EuQoS [133], Phosphorus [104], Nortel's DRAC [16], VIOLA [28, 152], Enlightened [45], G-Lambda [146]. Tous ces projets proposent des modèles et des techniques pour offrir une qualité de service réseau aux applications de la grille. La figure 1.23 de la page 43 reprends les caractéristiques de chacun de ces projets. Les éléments différentiateurs sont les suivants: méthode pour offrir la QoS

(orienté paquet ou circuit), niveau de communication entre le réseau et les applications (niveau applicatif ou réseau), prise en compte de plusieurs domaines réseaux ou non. Nous avons regroupés ces projets en trois familles correspondant aux technologies sous jacentes les plus utilisées: Web Service, GMPLS et IMS (IP Multimedia Subsystem). La section suivante correspondant au chapitre 2 va décrire plus en détail ces architectures et les contributions de cette thèse correspondant pour chacune.

Architectures

Cette section reprends les propositions majeures de cette thèse en matière d'architecture des systèmes de co-allocation de ressources. Nous décrivons dans un premier temps les extensions aux protocoles réseaux, puis à l'architecture IMS et enfin aux architectures reposant sur les Web Services. Ces extensions ont fait l'objet de plusieurs brevets [111, 112, 114, 151] et publications [117, 118].

La fourniture d'un service de calcul offrant des garanties de qualité de service nécessite la réservation des ressources de calcul et réseau. Quand une tâche nécessite plusieurs ressources différentes, la probabilité que deux fournisseurs soit en mesure de fournir la ressource immédiatement est faible, elle diminue avec le nombre de ressources qui doivent être disponible au même instant. Il est donc indispensable de pouvoir réserver les ressources à l'avance afin de planifier leur utilisation. Avec l'émergence d'architectures comme MPLS et GMPLS, il est désormais possible de réserver la ressource réseau afin de garantir un débit ou une latence. Plus rapidement, on peut parler de réservation de bande passante ou de bande passante à la demande. Toutefois, les mécanismes actuels ne permettent pas de réserver aujourd'hui de la bande passante pour une utilisation future. Il est impossible avec GMPLS d'envoyer un message aux nœuds du réseau du type "Réserve une connexion virtuelle qui sera exploitée demain entre 3h et 6h". Cette fonctionnalité, aussi appelée planification des ressources peut être réalisée de deux façons différentes. La première consiste à ajouter cette fonctionnalité au sein du gestionnaire du réseau, la seconde consiste à modifier les protocoles de contrôle actuels, RSVP-TE et OSPF-TE. Yong [36] propose à l'Internet Engineering Task Force IETF d'ajouter au gestionnaire du réseau un module permettant de recevoir les requêtes de connexion à la demande, de les traiter, de gérer une base de donnée des requêtes et de l'état futur du réseau, de calculer les routage futurs du réseau et de gérer l'allocation des ressources le moment venu. La figure 2.3 de la page 55 décrit ce type d'architecture. Une autre approche consiste à modifier directement le plan de contrôle afin d'intégrer les fonctionnalités voulues. Il est intéressant de modifier le protocole de signalisation RSVP-TE, de routage [114] et l'architecture de calcul de route Path Computing Element PCE [112] afin que tous trois prennent en compte l'état futur du réseau et soient capable d'automatiser les demandes de connectivité futures.

La modification à apporter au protocole de signalisation consiste à rajouter les champs décrivant les informations temporelles d'utilisation de la connexion. Ainsi modifié, RSVP-TE pourrait comme faire suivre les requêtes futures comme les requêtes immédiates de nœud en nœud. Chaque nœud traiterait les nouvelles requêtes en analysant une base de données contenant l'état futur du réseau. Cette base pourrait avoir été construite à partir du gestionnaire du réseau ou grâce à un protocole de routage modifié de façon à diffuser

l'état futur du réseau décrit dans le brevet [114] ou encore à partir d'un système ad-hoc de type météo du réseau.

Aujourd'hui, les solutions pour fournir une réservation à l'avance des ressources utilisent une architecture centralisée. De plus, l'établissement d'une connexion future nécessite une intervention humaine sur le gestionnaire de réseau afin de lancer l'optimisation future du réseau et la configuration des liens. Une fois l'état futur du réseau déterminé il est téléchargé dans chacun des nœuds. Ce processus est donc complètement manuel et pas du tout automatisé. L'idée d'un protocole de routage comprenant les informations temporelles permettrait de distribuer et d'automatiser le processus de calcul des routes futures en diffusant l'état futur du réseau dans chaque nœud. Toutefois le plan de contrôle du réseau transporterait des informations liées au futur, l'impact de ce surplus d'information sur le dimensionnement du plan de contrôle est encore à étudier.

Le groupe de travail sur l'architecture d'éléments de calcul de route PCE a démarré en 2004. L'idée de départ consiste à externaliser la fonctionnalité de calcul de route gourmande en ressource de calcul des routeurs et de la mettre dans un nouvel élément du réseau: le PCE. La requête de calcul de route serait donc envoyé par un client, le Path Computation Client PCC au Path Computation Element PCE. Des calculs de routes récursifs peuvent ainsi se faire par le biais de cette communication entre éléments. Il devient alors possible de calculer des routes inter domaines dans le but de faire de l'ingénierie de trafic inter domaine. Une liste complète des avantages du PCE peut être trouvée dans le document architectural [35]. Cette thèse propose [112] d'utiliser le PCE afin de calculer les routes à l'avance dans le réseau et d'étendre les fonctionnalités du PCE à la planification des routes dans le réseau. Une telle utilisation est décrite dans la figure 2.4 de la page 57. Dans ce scénario les PCE sont utilisés pour établir une connexion inter domaine dans le futur à la suite d'une demande de l'ordonnanceur de la grille. Dans le scénario le plus simple, l'ordonnanceur n'a pas la connaissance du réseau sous jacent et ne connait rien de la topologie ni de l'état des ressources réseaux. Il demande à un fournisseur de connectivité l'établissement de la connexion entre le client et la ressource de calcul. Le fournisseur de connectivité a besoin de connaitre la route à utiliser dans le futur. Il envoie donc un message de calcul de route à son PCE. Ce message est traité par les différents PCE du réseau, chacun calculant la route propre à son domaine de responsabilité. Ce calcul est représenté par les flèches pleines sur la figure 2.4. Une fois la route future calculée, il est nécessaire de réserver les ressources réseaux sur cette route, pour cela, il est possible d'utiliser une extension du PCE comme le suggère la figure ou un protocole de signalisation future comme décrit précédemment. L'utilisation du PCE pour réserver les ressources (unsolicited mode) est possible, néanmoins des extensions sont nécessaires afin de réserver des resources dans le futur. Dans ce mode, le calcul de route et la réservation dans le futur se feraient grace aux PCE, dès que les routeurs ont besoin de recevoir l'information de réservation, le PCE ou le service de connectivité leur enverrait par le biais d'une interface verticale (traits pointillés sur la figure 2.4).

GMPLS est une architecture qui comprend deux protocoles de contrôle du réseau: Un pour diffuser l'état du réseau à l'ensemble des nœuds et un pour demander l'établissement de connexions virtuelles par le biais de réservation de ressources. L'idée de GGMPLS [132, 34] Grid GMPLS consiste à réutiliser les mécanismes de GMPLS et de les étendre à d'autres ressources comme le stockage ou le calcul. Un nouveau objet de type Opaque Link

State Advertisment (Opaque LSA) est proposé pour les ressources de calcul et de stockage. Ces objets sont des objets du protocole OSPF-TE conçus pour être transmis par tous les routeurs. Seuls les routeurs possédant la fonction de traiter les nouveaux objets sauraient interpréter ces messages. Il en serait ainsi des applications de la grille, des ressources de stockage et de calcul et des routeurs GGMPLS. Les informations contenues dans cet objet serait l'identifiant Grille, la puissance CPU maximum disponible, la puissance disponible instantanée, l'espace de stockage disponible, etc. Ces extensions permettraient donc de diffuser l'état de l'ensemble des ressources, calcul, stockage et réseau à l'ensemble des routeurs et application connectées à ces routeurs. En plus de ces modifications à OSPF-TE, des modifications seraient apportées au protocole de réservation des ressources réseau afin d'étendre sa fonctionnalité à d'autres types de ressource. Ces idées regroupées sous le titre Grid User to Network Inteface proposées en 2004 [135] mais non détaillées sont très ambitieuses. Le projet Phosphorus [104] semble vouloir matérialiser cette interface. Les fonctionnalités de l'interface sont très nombreuses: Gestion des jobs (soumission, suivi, contrôle), Gestion de la sécurité, Représentation et Manipulation des états des ressources, Notification, Allocation de bande passante flexible, établissement de connexion future, détection des erreurs, protection, restauration, propagation des événements relative aux serices, classification du trafic, etc. Les bénéfices d'étendre GMPLS à d'autres types de ressource ne sont pas très clairs en dehors d'une "Intégration globale des ressources". Un des facteurs ayant menés à la création de standards comme MPLS ou GMPLS a été que les coûts associés à la gestion de différentes technologies (IP, ATM, Frame Relay, Ethernet) étaient supportés par une même entité: l'opérateur réseau. Aujourd'hui, des entités différentes gèrent les ressources réseau et les ressource de stockage, le besoin n'est donc pas de réduire les coûts de gestion en proposant une architecture unique commune, mais de disposer de mécanismes de collaboration, de coordination et de co-allocation. De plus, GMPLS gère des objets binaires et non textuels, la communauté de développeurs ayant à maitriser GMPLS est très petite alors que celle ayant à maitriser les protocoles de gestion des Grilles est énorme. Les protocoles binaires tel GMPLS sont plus durs à concevoir, maitriser, dépanner. GMPLS est conçu par la communauté réseau qui ne comprend pas toujours la communauté informatique. GMPLS n'a pas été conçu pour gérer n'importe quel type de ressource et est prévu pour fonctionner dans un environnement protégé, des fonctionnalités importantes de sécurité lui font défaut. Enfin, ce protocole est prévu pour faire fonctionner des systèmes extrêmement stables qui doivent offrir un niveau de fiabilité et de disponibilité incomparable. Toute modification du standard même infime est très difficile à obtenir. Pour ces raisons, les développements des Grilles reposant sur des modifications de GMPLS ont peu de chance d'être un jour standardisé. De plus, des technologies alternatives existent et ne présentent pas ces défauts, ces technologies reposent sur l'XML Extensible Markup Language. Par nature, elles sont conçues pour supporter des innovations et des extensions facilement.

IP Multimedia Subsystem IMS [14] est l'architecture réseau de nouvelle génération (Next Generation Network au sens de l'Union Internationale des Télécommunication). Cette architecture est standardisée par le 3GPP [25]. Elle propose des éléments communs pour fournir n'importe quel type de services multimédia à n'importe quel type de terminal. A l'origine, les terminaux envisagés étaient des terminaux mobiles, néanmoins avec la converge fixe mobile, l'architecture a évoluée pour fournir n'importe quel type de service à

n'importe quel type de terminal fixe ou mobile. L'architecture IMS distingue trois acteurs: les utilisateurs finaux, l'opérateur réseau et les fournisseurs de services. Ces derniers sont hébergés sur des serveurs applicatifs. L'architecture a été conçue pour permettre l'ajout de nouveaux services à l'infrastructure réseau actuelle. Bien sûr, la notion de service dans le contexte IMS vient du monde des télécommunications mobiles, les premiers services envisagés sont des services de télécommunication, suivi d'appel, répondeur, vidéo, etc. Néanmoins la flexibilité de l'infrastructure, l'existence de fonctions communes comme la facturation, la gestion des utilisateurs, la sécurité en font un candidat important pour la fourniture d'un service de calcul. L'emploi de l'architecture IMS dans le but de fournir des services de calcul [117, 118] est l'une est contribution originale de cette thèse. Bien que nous n'ayons pas pu implémenter nos propositions, l'utilisation du protocole Session Initiation Protocol qui est au cœur de l'architecture IMS dans le cadre de service de calcul a été réalisé par Campi [53, 54, 52, 54].

L'architecture IMS peut être vue comme une architecture en deux strates telle décrite par le cadre NGN de l'ITU: une strate service et une strate transport (voir la figure 1.22 de la page 41). Chaque strate étant composée d'un plan usager, de contrôle et de gestion. L'architecture IMS définie les fonctions et les interfaces entre les différents plans de contrôle de façon à ce que les acteurs puissent jouer pleinement leur rôle. Les protocoles utilisés par l'IMS sont développés par l'IETF: IP, SIP, Session Description Protocol SDP [85], Diameter. L'ensemble de l'architecture repose sur un paradigme de communication en mode connecté, c'est-à-dire que les communications se font dans le cadre d'un appel ou d'une session entre participants. Il est donc nécessaire d'établir la session avant que la communication ou l'échange de données puisse commencer. Les éléments qui vont router les appels dans le réseau sont des contrôleurs de session, plus de détails sont donnés dans le chapitre 2. La base utilisateur est gérée par le Home Subscriber Server. Elle contient toutes les informations relatives aux utilisateurs, les profiles, filtres et règles de gestion. Les serveurs applicatifs hébergent les applications qui fourniront le service à l'utilisateur, ils communiquent avec le reste de l'IMS par SIP.

La figure 2.5 de la page 63 décrit les extensions nécessaires pour fournir un service de calcul. La figure 2.6 de la page 64 illustre les messages échangés et les étapes pour établir une session de calcul avec une ressource de calcul. Nous allons prendre comme exemple un utilisateur qui a besoin de traiter un énorme fichier vidéo qui réside sur le site de gauche "Application" de la figure 2.5. Nous supposerons que les applications et les ressources utilisent un grid toolkit fournissant les fonctionnalités de base de gestion de jobs, des données et de suivi. Seulement quelques fonctions des toolkits doivent être modifiées pour prendre en compte l'architecture IMS. Nous supposerons que le réseau est contrôlé par (G)MPLS. Les deux boites orange représentent les contrôleurs de session de l'architecture IMS. L'élément PDF Policy Decision Function fait le lien entre la réservation des ressources de calcul et des ressources réseaux, c'est cet élément qui décide si l'utilisateur peut ou non réserver des resources réseaux. Les différentes étapes du processus du traitement du fichier vidéo sont maintenant décrites.

Le scénario commence par l'enregistrement dans le réseau des ressources de calcul, les ressources à droite dans la figure 2.5 envoie un message SIP REGISTER multimediaprocess@serviceY.com à l'architecture IMS, ce message arrive jusqu'à l'annuaire IMS (SIP registrar) étendu de nouvelles fonctionnalités représenté par l'élément "Advance Regis-

trar". Cette étape est réalisée une seule fois lors de l'activation des ressources, elle permettra à l'architecture IMS de faire le lien entre l'adresse logique IMS des ressources et leur adresse physique IP. Une fois cette étape réalisée, les utilisateurs pourront avoir accès aux ressources de calcul enregistrées. Quand ils arrivent sur le réseau, ils s'identifient eux aussi par le biais du message SIP REGISTER qui parviendra jusqu'à l'annuaire. Ce processus est aussi représenté dans la figure 2.6 de la page 64. Dans l'IMS, l'annuaire serait relié au Home Subscriber Server.

Quand les utilisateurs souhaitent soumettre leurs jobs sur la grille, le grid toolkit cherche à établir une session SIP avec les ressources de calcul. Il envoie un message SIP INVITE à multimediaprocess@serviceY.com contenant la description du job. Ce message est ensuite transmis de serveur relai en relai pour parvenir jusqu'à un service de localisation des ressources avancé relié à l'annuaire. Ce service va alors pouvoir réaliser le processus de sélection des ressources, c'est-à-dire choisir parmi les ressources disponibles lesquelles répondent le mieux au besoin de l'utilisateur compte tenu de l'état du réseau et des ressources. Cette sélection peut prendre en compte les règles de gestion des opérateurs, les préférences de l'utilisateur et les critères de performance comme par exemple la vitesse de traitement des machines, la capacité du réseau, la latence moyenne, etc. Etant fourni par l'opérateur réseau, il accède au plus d'information possible, c'est lui qui porte la valeur ajoutée de l'opérateur. Cette thèse propose d'adapter les mécanismes de "SIP forking" existant du type "sonne tous les téléphones partageant la même adresse" à "sonne tous les téléphones partageant la même adresse et répondant aux objectifs d'optimisation et aux règles de gestion". Par exemple, un fournisseur de service souhaitant équilibrer la charge entre deux fermes de calcul pourrait installer cette règle dans le registre avancé. L'équilibrage de charge serait réalisé en prenant en compte la charge en temps réel et la bande passante disponible entre les utilisateurs et les fermes.

Une fois que le site de calcul a reçu la requête SIP INVITE demandant l'établissement de la session, les participants de la session doivent se mettre d'accord sur les ressources réellement utilisées et les paramètres d'emploi de celles-ci. Cette étape est une étape de négociation des caractéristiques de la session. Différents mécanismes SIP [129, 131, 130] existent déjà pour réaliser cette étape: le modèle de communication offre/demande utilisé par exemple dans la négociation de codec, les pré conditions de QoS. Ces mécanismes peuvent être réutilisés pour négocier la présence de certaines bibliothèques de fonctions ou d'exécutables sur la machine cible. A la fin du processus de négociation, un des serveurs SIP demande la création d'un jeton dans l'élément de décision (PDF). Ce jeton représente qu'un session SIP est sur le point de commencer que les participants sont identifiées et autorisés et que les ressources applicatives sont prêtes. A priori les ressources réseaux sont aussi disponibles à moins de n'avoir été préemptées par d'autres processus entre temps. La réservation des ressources réseaux peut donc commencer, cette étape est représentée dans la figure 2.6. Dans une première variante le grid toolkit possède une implémentation du protocole RSVP-TE, dans une seconde variante, le contrôleur de session à la périphérie du réseau (à gauche sur la figure) déclenche la signalisation réseau dans le routeur de périphérie par le biais d'une interface verticale. Dans les deux approches le message qui parvient au routeur de périphérie contient le jeton lui permettant de vérifier si la source du message à le droit de d'initier une réservation réseau. Le routeur interroge donc le PDF qui lui confirme ou rejette la validité du jeton. Dans le cas décrit

ici, le jeton est validé et la réservation de ressource réseau peut se faire. Les services multimédia prévu par l'IMS nécessitent une QoS, il en est de même pour les services d'accès aux ressources de calcul ou de stockage. Les fonctions de contrôle d'admission pour le réseau d'accès sont en cours de standardisation et doivent être étendue pour le réseau de cœur. C'est un des objectifs de l'ETSI (European Telecommunication Standardisation Institute) TISPAN (Telecommunications and Internet Converged Services and Protocols for Advanced Networking).

L'architecture IMS possède des mécanismes pour prendre en compte la facturation de chaque session. Ces mécanismes peuvent être complètement réutilisés dans le cadre de la fourniture d'autres types de services. De plus, le protocole SIP permet la gestion de message de suivi et de contrôle [127] qui sont nécessaire à la gestion d'une grille de calcul.

De nombreux développements dans l'architecture IMS et le protocole SIP constituent les fonctions de base indispensables à une grille de calcul: l'authentification des utilisateurs, l'autorisation d'accès, la facturation par utilisateur et par session, la négociation des ressources, la qualité de service et la réservation des ressources. L'intégration de ces fonctions entre les grilles et le réseau permettrait de générer de la valeur pour les opérateurs réseaux, les utilisateurs de grille et les fournisseurs de ressources. Dans le cadre des développements actuels d'un réseau pour tous les services, le développement parallèle à l'IMS de l'ensemble des fonctionnalités de base en utilisant l'architecture orienté service (SOA) pour fournir des services de calcul semble constituer un doublon inutile. Cela illustre les difficultés de compréhension et de communication des deux communautés: celle des télécommunications pour l'IMS et celle des informaticiens pour les architectures SOA. Cette thèse propose aux gestionnaires de la grille et aux fournisseurs de ressource de s'affranchir de la gestion de certaines fonctions en se reposant sur l'architecture envisagée pour les réseaux de nouvelle génération. C'est le but d'une architecture Grille intégrée aux réseaux de nouvelle génération. Toutefois des travaux supplémentaires concernant la performance de cette architecture intégrée nécessite d'être réalisés.

Les Web Services reposent sur l'architecture orientée service (SOA), cela veut dire que chaque fonction de l'architecture est réalisée par une entité cliente et une entité serveur qui communiquent selon un modèle requête/réponse. D'autres types de communication ne faisant pas partie de l'architecture SOA sont par exemple la signalisation telle qu'utilisée dans les réseaux de télécommunications, d'autres encore comme la notification d'erreurs à toutes les entités de gestion. L'avantage de cette architecture est qu'il facilite la réutilisation et la composition des fonctions de base pour construire des systèmes plus complexes fournissant des services plus avancés. Le paragraphe suivant va décrire le fonctionnement de base d'un service de calcul permettant à la co-allocation des ressources. Le projet VIOLA [28] (Vertically Integrated Optical Network for Large Application) sera pris en exemple car représentatif de l'état de l'art. Nous nous focaliserons plus particulièrement sur le fonctionnement de l'ordonnanceur de la grille appelé meta-scheduler [152] dans le projet. VIOLA utilise le Grid toolkit Unicore [66], la figure 2.1 de la page 51 présente un schéma global de l'architecture. Les éléments en violet font partie d'Unicore: Unicore client est le client Unicore (voir figure 2.2 à la page 52), Unicore Gateway est l'application qui se charge des problématiques de sécurité, NJS pour Network Job Supervisor se charge de la gestion des jobs, TSI pour Target System Interface se charge de l'adaptation (au sens des modèles de conception [76]) entre les fonctionnalités de gestion d'Unicore et celles offertes

par le gestionnaire installé sur le site de calcul. Les éléments RMS (Resource Management Systems) désignent les gestionnaires de ressources sur les sites de calcul, ils incluent les ordonnanceurs locaux. L'élément ARGON (Allocation and Reservation in Grid-enabled Optic Networks) est l'adaptateur entre le gestionnaire du réseau et Unicore. L'élément MSS (Meta-Scheduling service) est l'ordonnanceur de la grille. C'est lui qui va recevoir les demandes de traitement des clients, sélectionner les ressources et soumettre les jobs sur les ressources. Le fonctionnement global de l'architecture est simple: l'ordonnanceur de la grille coordonne l'utilisation des ressources, il se charge de sélectionner quelles sont les meilleures ressources à utiliser pour chaque job et planifie leur utilisation en les réservant. Chaque élément décrit communique avec les autres par le biais d'interfaces Web Service et de requêtes SOAP.

Cette section a décrit les principales architectures utilisées pour fournir un service de calcul réservant les ressources réseaux et de calcul. Trois architectures ont été présentées: GGMPLS reposant sur les protocoles purement réseau comme GMPLS et ses extensions, IMS et Web Service. Des extensions permettant l'automatisation de la planification du contrôle des ressources réseaux ont été proposées et constitue une contribution importante de cette thèse sous la forme de brevets [111, 112, 114]. L'utilisation de l'architecture IMS pour fournir des services grille est une autre contribution originale de la thèse a été décrite et fait l'objet de publications [117, 118]. L'avenir de l'architecture GGMPLS semble compromis quand on considère la petite communauté de développeurs maitrisant déjà les bases de cette architecture: GMPLS. De plus, la complexité des protocoles GMPLS et leur conception ne permettent d'être étendu avec la même facilité que les architectures SOA. De trop nombreuses modifications seraient nécessaire à ces protocoles. L'approche IMS bien qu'originale est tributaire du succès de l'IMS qui n'est pas encore assuré à l'heure de rédaction de ces lignes. Reste l'architecture Web Service qui malgré ses défauts possède une grande communauté d'utilisateurs. Elle offre par ailleurs un degré de flexibilité très apprécié et permet de s'intégrer avec d'autres technologies, notamment GMPLS, le PCE ou l'IMS. L'architecture du projet VIOLA présentée dans cette section est un exemple de réussite d'intégration. La section suivante et le chapitre 3 se focalisent sur cette architecture et approfondissent plus en détail le protocole de communication entre l'ordonnanceur de la grille et les gestionnaires de ressources locaux pour permettre la co-allocation.

Protocoles

Cette section étudie plus en détail le protocole de co-allocation. Dans un premier temps le rôle des SLA dynamiques est montré. Pour parvenir au degré de service décrit par ces contrats, la co-allocation de ressource est la solution étudiée dans cette thèse. Les différentes étapes permettant celle-ci sont donc étudiées. Puis en prenant l'exemple du projet VIOLA, nous étudierons et comparerons deux protocoles pour parvenir à la co-allocation. WS-Agreement est un protocole permettant la création de SLA. Les innovations portée à celui-ci ont fait l'objet des publications [119, 120].

Depuis longtemps déjà, les opérateurs de télécommunications utilisent les SLA avec leurs clients. Ces contrats ont une durée de vie très longue et les changements des paramètres régissant la qualité peuvent nécessiter plusieurs semaines de procédure. Les

applications distribuées envisagées ont des besoins en matière de ressource et de qualité de service très variable. La fourniture d'un service d'accès à des ressources de calcul nécessite de pouvoir mettre en œuvre pendant une durée limitée des quantités de ressources très variables selon les applications. Des contrats matérialisant l'utilisation temporaire d'une quantité définie à la dernière minute de ressource par exemple dans un cadre commercial de location à la demande de puissance de calcul conjointe à une bande passante réseau sont nécessaires. De tels contrats ont une durée de vie limitée et sont créés à la dernière minute: ce sont des SLA dynamiques. La création de SLA dynamiques et la négociation des paramètres de ceux ci sont indispensables à la fourniture d'un service de calcul garantissant une qualité de service. L'ordonnanceur de la grille après avoir reçu la demande de traitement d'une tâche est chargé de trouver les ressources et de négocier les SLA avec les différents fournisseurs. Les différentes étapes de traitement d'une tâche par l'ordonnanceur de la grille avant l'exécution de celle-ci sont décrites dans la figure 3.1 de la page 69. Schopf avait déjà décrit ces étapes dans [136]. A la réception de la description de la tâche l'ordonnanceur commence par éliminer les ressources (étape 1) qui ne correspondent pas aux contraintes décrites dans la tâche qu'elles soient liées à des informations statiques (Système d'exploitation, Vendeur, débit d'accès maximal...) ou dynamique (charge disponible, en maintenance...). Une fois un premier jeu de fournisseurs potentiels identifiés, la négociation peut commencer (étape 2). Le premier temps de la négociation est la négociation à proprement parlé, il s'agit de savoir quels fournisseurs peuvent réellement traiter la tâche demandée, une communication est nécessaire entre les fournisseurs de ressource et l'ordonnanceur. Une fois ces fournisseurs identifiés, un choix doit être fait selon des critères d'optimisation et les différentes règles de gestion. Une fois le choix des fournisseurs et des ressources établis, la création des SLA peut commencer (étape 3), celle-ci peut se faire en une phase ou deux phases. La dernière étape (étape 4) est la soumission de la tâche aux différentes ressources et l'association entre la tâche et les contrats établis par l'ordonnanceur au nom de l'origine de la tâche.

La négociation est un sujet de recherche à part entière [139, 50, 96, 95, 51]. La négociation de SLA dynamiques doit être automatisée et la littérature sur ce sujet est aussi très abondante [88, 140, 80, 58]. Cette thèse se focalise sur la négociation de contrats bilatéraux, c'est-à-dire entre deux parties: l'ordonnanceur et le fournisseur de ressources. Plusieurs contrats bilatéraux étant ensuite regroupés en un seul pour fournir le service désiré. Les systèmes de base de donnée distribués ont été largement étudiés par le passé, ils ont permis de mettre en évidence les mécanismes pour propager un état consistant sur plusieurs machines. Les experts [47, 92, 110] ont proposés des protocoles permettant de propager les changements d'états à l'ensemble des machines: "commit protocols". Les protocoles principaux reposent sur deux étapes: une étape de demande de changement d'état auprès de chacune des machines puis une demande de validation (commit) du nouvel état.

Dans un système de création de SLA bilatéraux avec plusieurs fournisseurs dans le but de réaliser une co-allocation, l'ensemble des SLA bilatéraux forme un tout logique. Chacun des SLA bilatéraux n'a pas de raison d'être sans l'existence des autres. Leur création doit donc être collective. Les travaux sur les systèmes distribués s'appliquent à la création de SLA pour réaliser une co-allocation. A chaque demande de changement d'état correspond une création de SLA bilatéral, la phase de validation est nécessaire

quand tous les SLA bilatéraux ont été créés. Si un SLA bilatéral n'a pu être créé, il est nécessaire d'annuler les autres SLA bilatéraux. Pourtant le protocole WS-Agreement [41] de création de SLA ne comporte pas de phase de confirmation de création de SLA. Nous avons donc proposé [120, 119, 154] d'étendre le protocole actuel afin de permettre 1) la création de SLA en deux phases, création et confirmation, et 2) de négocier l'accès aux ressources.

Le fonctionnement de l'ordonnanceur de la grille du projet VIOLA est décrit à la figure 3.3 de la page 76. Cette figure représente en bleu dans le cadre supérieur les différentes étapes du processus de co-allocation au sein de l'ordonnanceur de la grille (Grid scheduler process), dans le cadre du milieu en rose clair les étapes du processus sur les gestionnaires de ressources locaux et en jaune clair dans le cadre inférieur les ressources. Afin de permettre à l'ordonnanceur de la grille de communiquer de la même façon avec n'importe quel type de ressources, des adaptateurs ont été implémentés au dessus de chaque gestionnaire de ressource locale pour lui présenter toujours la même interface et les mêmes fonctions. Ces fonctions sont les suivantes: CouldRunAt, Submit, Cancel, State. CouldRunAt prend en paramètre une description de tâche à exécuter et une date (date de décalage) avant laquelle l'execution ne peut pas commencer. Elle renvoie la date à laquelle la tâche peut commencer son exécution au plus tôt sur la machine donnée après la date de décalage. Submit soumet une tâche et renvoie une référence vers la réservation. Cancel annule une réservation. State interroge l'adaptateur pour connaitre l'état d'une réservation comme par exemple la date à laquelle la réservation est effective. Pour co-allouer les ressources l'ordonnanceur commence d'abord par interroger (Etape Create Resource List) les différentes ressources par le biais des adaptateurs en appelant la fonction CouldRunAt avec pour date de départ la date présente. Il établit alors une liste des ressources disponibles. Il cherche alors dans cette liste si les ressources nécessaires par la tâche sont disponibles au même instant (Etape Calculate Timeslot). Si les ressources sont disponibles au même instant il envoie le message submit (Etape Schedule Resource) aux adaptateurs des ressources pour réserver, sinon il réitère l'étape "Create Resource List" en envoyant un nouveau message CouldRunAt avec comme date de décalage la plus grande des dates d'exécution au plus tôt, et ainsi de suite jusqu'à temps qu'une des ressources soit disponible ou que l'on ait dépassé les contraintes d'exécution de la tâche, auquel cas, la tâche est rejetée. Après avoir reçu la demande de réservation de l'ordonnanceur de la grille, les adaptateurs locaux doivent demander aux gestionnaires de ressources locaux la réservation des ressources pour la tâche. Ils renvoient un message décrivant simplement si la réservation a réussi ou échoué. L'ordonnanceur de la grille doit alors interroger les adaptateurs (Etape Get Reservation Properties) en envoyant un message State pour connaitre la date à laquelle les ressources ont réellement pu être réservées. Il doit ensuite s'assurer que la date de réservation de l'ensemble des ressources nécessaires est bien la même (Etape Check Reservation), si oui, la co-allocation a réussi, sinon elle a échoué, il faut alors annuler les réservations effectuées et itérer le processus. Le problème vient du fait qu'après avoir réalisé l'étape de Négociation, ici l'état Create Resource List, les ressources locales ont pu être utilisées par des utilisateurs locaux par exemple et ne sont donc plus disponible au moment où le message de réservation (Submit) est reçu. La figure 3.4 de la page 77 décrit ce processus. Ce que nous venons de décrire est une version assez ancienne de VIOLA, toutefois des versions plus récentes exploitent

les même mécanismes fournis par le protocole WS-Agreement. Ce protocole possède les mêmes défauts.

Le fonctionnement de base de WS-Agreement est décrit dans la figure 3.5 de la page 78. Le protocole a été créé pour demander la création d'un SLA entre deux parties. Une forme de négociation ultra simplifiée est possible avec WS-Agreement, c'est une négociation en une étape dans laquelle le contexte, l'objet et les contraintes du SLA sont définies. La méthode getResourceProperties renvoie un ensemble de modèle de SLA que le fournisseur du SLA est susceptible d'accepter. En aucun cas, le fournisseur du SLA ne s'engage à accepter le modèle donné en réponse, néanmoins ce modèle peut contenir par exemple les contraintes permettant d'exclure d'office certaines demandes de SLA. Par exemple, ces contraintes peuvent être le nombre maximum de CPU qu'il peut offrir, ou la durée minimum d'une exécution sur ses ressources, la mémoire disponible. L'autre interface dans WS-Agreement permet la création du SLA: createAgreement. Toutefois la réponse à cette fonction est un message disant si la réservation à réussi ou échouée. Si réussi le contrat est créé. Il sera nécessaire de vérifier dans une seconde étape si ce contrat correspond bien à ce qui avait été demandé, notamment en matière de date de début de la réservation. Et de même que précédemment il sera nécessaire d'annuler le SLA si les dates ne correspondent pas.

Compte tenu de l'absence de support natif pour négocier des SLA dans WS-Agreement, et contrevenir à ce problème de devoir annuler un contrat si la réservation n'est pas bonne, avec Wäldrich et Ziegler, cette thèse propose d'étendre WS-Agreement et d'ajouter une phase de confirmation du SLA et de réaliser la négociation en trois étapes: négociation de SLA, création de SLA, confirmation. Ces trois étapes sont représentées dans la figure 3.7 de la page 81. La suite de cette section va décrire un modèle d'évaluation de performance et comparer celle du protocole de co-allocation en deux phases, négociation et création à celle du protocole en trois phases, négociation, création et confirmation.

Une des critiques souvent formulée à l'encontre des architectures Web-Service est leur problème de performance. Le traitement des messages XML introduit une latence dans les architectures SOA. Nous avons voulu dans cette thèse étudier sommairement ce point et construire un modèle nous permettant d'évaluer la performance du protocole de co-allocation. Nous avons focalisé l'étude sur la comparais de la performance entre un protocole à 2 phases de négociation et un protocole à 3 phases. Afin de mener cette étude, nous avons construit différents réseaux de file d'attente représentant chaque protocole. Le modèle utilisé a été proposé pour la première fois par Gurbani [84] pour l'évaluation de la performance d'un serveur SIP. Dans notre contexte, nous évaluons la performance de l'ordonnanceur de la grille. Les indicateurs de performance que nous avons étudiés sont les suivants: Délais total introduit par le processus de co-allocation, nombre de demande de tâche traitées par seconde par l'ordonnanceur, besoin mémoire nécessaire par l'ordonnanceur. Nous avons supposé que l'ordonnanceur maintient en mémoire un état correspondant à chaque tâche en cours de co-allocation. Les clients du réseau de file d'attente seront les requêtes de co-allocation contenant la description des tâches et des ressources nécessaires. A chaque état au sein de l'ordonnanceur de la grille correspond une file d'attente, le passage d'un état à un autre est conditionné par la réalisation de certaines opérations qui nécessitent un certain temps. La théorie des réseaux de Jackson est utilisée et les hypothèses classiques sont faites. Ainsi les temps de service des files sont

exponentiellement distribués, les clients arrivent selon un processus poissonien, chaque file contient un seul serveur. Nous pouvons donc appliquer les formules analytiques classiques pour calculer les indicateurs de performance. Les hypothèses simplificatrices faites pour pouvoir faire des calculs ne sont pas très réalistes, mais elles suffisent pour construire un modèle cohérent, comparer les deux protocoles et avoir une intuition de la performance globale de l'ordonnanceur.

Le réseau du haut dans la figure 3.8 de la page 87 décrit le modèle utilisée pour le protocole à deux phases, alors que le protocole à trois phases est décrit par le réseau inférieur. Chaque serveur de chaque file introduit systématiquement un temps de traitement qui correspond à la réception d'un message, à son traitement par l'entité qui l'a reçu et à la génération d'une réponse et son envoi à l'entité suivante. Les délais de propagation ne sont pas pris en compte. Chaque file correspond un état dans l'ordonnanceur, parmi ces états, certains états sont des états d'attente de réponse des adaptateurs. Une file correspondant à cet état introduira donc le temps mis par l'adaptateur à recevoir le message, à le traiter et à générer la réponse. La convention suivante à été adopté, la lettre au dessus de la file décrit qu'elle entité doit recevoir le message, le nom du message est indiqué sur la file, et le taux de service μ ou l'inverse du temps moyen de traitement est dans le serveur. LS signifie Local Scheduler, MS Méta-Scheduler, A pour ARGON, c'est à dire, l'adaptateur du gestionnaire du réseau. Prenons par exemple la première file, la requête de co-allocation est dans un état initial "non créé", une fois le message "CreateAgreement" reçu par l'ordonnanceur de la grille, celui ci doit disséquer le message, identifier le type de ressource nécessaire, et envoyer un message "CouldRunAt" aux adaptateurs locaux. Le temps moyen pour réaliser ces trois opérations est $1/\mu_1$. Le client passe alors dans la deuxième file qui correspond à l'état "en attente de réponse des adaptateurs". La deuxième file introduit le temps passé dans cette état d'attente. Ce temps est égal au temps nécessaire par les adaptateurs locaux pour recevoir le message "CouldRunAt", le traiter et envoyer la réponse à l'ordonnanceur, il s'agit de $1/\mu_2$. Les probabilités de transition d'un état à un autre sont supposées constantes. Après avoir reçu la réponse des tous les adaptateurs, l'ordonnanceur doit déterminer si les ressources nécessaires sont disponible au même temps, la probabilité q est la probabilité d'échec et qu'il faille réitérer le processus. p est la probabilité de devoir envoyer un message d'erreur à l'utilisateur suite à l'impossibilité de co-allouer les ressources. m est la probabilité de trouvé un temps commun à toutes les ressources, $m + p + q = 1$. La probabilité d'itération q dépends de nombreux facteurs, la charge sur les ressources locales est un des plus important. Cette charge est agrégée dans un facteur relativement abstrait q. C'est une des faiblesses de cette modélisation. Nous la contournerons en étudiant l'ensemble des indicateurs de performance comme des fonctionnelles de q. En résolvant le système linéaire associé à ce réseau de jackson, nous pouvons calculer la charge maximale supportée par le système, les charges internes à chaque file et calculer le nombre moyen de client dans le réseau. Nous en déduisons les besoins mémoire de l'ordonnanceur. Quant au délai total moyen pour réaliser la co-allocation il est déterminé à partir du nombre moyen de client dans le système grâce à la formule de Little.

La figure 3.9 de la page 89 représente le débit maximum de nombre de requêtes pouvant être traitée par seconde par l'ordonnanceur de la grille en fonction du paramètre q. La constatation principale est que la différence entre les deux protocoles est relative-

ment faible. La figure 3.11 de la page 90 représente le nombre moyen de requête dans l'ordonnanceur en fonction du paramètre q. La première constatation est un effet de seuil, c'est à dire au delà d'une certaine probabilité de bouclage le nombre de job dans le système explose. La leçon à tirer consiste à mettre en place des mécanismes supplémentaires dans le protocole de façon à ce que les messages qui restent trop longtemps dans un état expirent et soient annulés. De plus, le seuil pour le protocole à deux phases est le même que celui à trois phases. Le nombre de requêtes présentes dans l'ordonnanceur de la grille étant proportionnel à la mémoire requise pour la co-allocation dans l'ordonnanceur. La figure montre que dans un régime de fonctionnement proche du seuil, la quantité de mémoire nécessaire par le protocole à trois phases est le double de celle du protocole à deux phases. L'ordre de grandeur du nombre de job traités à la seconde dans un régime de fonctionnement normal est de 60 requêtes par secondes. Toutefois ces grandeurs n'ont pas pu être comparées à des résultats expérimentaux et sont donc à prendre avec toutes les précautions d'usage. Enfin, la figure 3.12 de la page 91 représente la pénalité qu'introduit le protocole à 3 phases sur le nombre de client dans le système par rapport au protocole à 2 phases. Cette pénalité est étudiée pour différentes valeurs du taux d'arrivée des clients. La conclusion a tiré est que le besoin en mémoire est globalement le double et que si le débit d'arrivée est multiplié par 10, 10% de mémoire supplémentaire sont nécessaires. La sensibilité par rapport à d'autres paramètres comme par exemple les temps de service a été étudiée. Elle est relativement faible, inférieure à 10%.

Cette section a décrit brièvement les fonctionnalités pour coallouer les ressources: négocier et créer les SLA. Les SLA sont une brique de base pour l'orchestration des ressources dans les systèmes de gestion de ressources distribuées. Des innovations ont été proposées au protocole WS-Agreement [119, 120] pour négocier dynamiquement les modèles de SLA. Deux protocoles de co-allocation ont été présentés, lŠun exploitant 3 phasesă: négociation, création et confirmation, et l'autre similaire à l'existant 2 phasesă: négociation et création. L'impact de performance du protocole en 3 phases a été étudié à l'aide d'un modèle analytique. L'impact majeur est que la mémoire requise par l'ordonnanceur coallouant les ressources en 3 phases est deux fois plus importante que celui coallouant en 2 phases. Après avoir présenter les protocoles de co-allocation dans cette section, la section suivante et le chapitre 4 présente, étudie et compare les techniques d'optimisation de l'allocation des ressources. Plus particulièrement, un algorithme d'optimisation de l'allocation des ressources réseaux et de calcul est présenté et évaluer par rapport aux techniques actuelles de co-allocation.

Optimisation

Afin de fournir un service de calcul garantissant la qualité offerte, l'importance des échanges entre le gestionnaire du réseau et de l'ordonnanceur des ressources est cruciale. La co-allocation des ressources réseaux et de calcul permet de garantir cette qualité de service. Cette section et le chapitre 4 qui lui est associé étudie les différents mécanismes d'interactions possibles. A la situation actuelle où le réseau est considéré comme une boite noire sont comparés le service de bande passante à la demande et le service d'optimisation croisé. L'exemple de la bande passante à la demande est développé ici pour des raisons

pédagogiques, mais en réalité un service plus générique de connectivité à la demande sous contraintes aurait pu être aussi considéré. Ce service nécessite que le réseau puisse répondre automatiquement à des requêtes de bande passante et garantir la bande passante étant donné une topologie virtuelle donnée. Le service d'optimisation croisée *XO* (pour Cross-Optimisation) est une des contribution de cette thèse, il propose de choisir les ressources réseaux et de calcul de façon à minimiser le temps de traitement vu par l'utilisateur des ressources. Il nécessite que le gestionnaire du réseau communique l'état de bande passante disponible pour la réservation sur les liens à l'ordonnanceur de la grille. Des améliorations de ce service respectant plus la confidentialité de cette information sont imaginables mais non développées ici. Cette section présente donc dans un premier temps la méthode que nous avons retenu pour comparer les trois modes d'interactions entre le gestionnaire du réseau et l'ordonnanceur de la grille: boite noire *Legacy*, bande passante à la demande *BoD*, optimisation croisée *XO*. Puis nous développons un modèle analytique permettant de d'avoir une intuition de la performance. Enfin, nous décrivons les simulations que nous avons menées pour comparer les trois services. Ces simulations valident les résultats analytiques obtenus. Les résultats de ce chapitre font l'objet d'un article [116] non encore publié à la date de rédaction de ces lignes et d'un brevet déposé [115].

Les problèmes de co-allocation qui prennent en compte le routage du réseau et les ressources de calculs ont été très peu étudiés. Différentes formulations peuvent être développées: sous la forme de problème d'ordonnancement ou sous la forme de problème de flot dans les graphes. Le début du chapitre 4 traite de la formulation sous forme de problème de flot dans les graphes. L'annexe 4.6 décrit l'algorithme de co-allocation *XO* proposé s'il devait être implémenté dans un ordonnanceur. Nous en avons étudiez le fonctionnement dans un cadre simplifié décrit plus loin.

Dans le contexte industriel de cette thèse, la conception d'algorithmes permettant l'optimisation de l'accès aux ressources doit répondre à certains critères de performance excluant différentes techniques d'optimisation. L'algorithme doit être capable de traiter les jobs à la volée, c'est à dire trouver une allocation des ressources à chaque arrivée de job, sans remettre en question les choix fait précédemment. Par conséquent, l'optimisation ne peut être parfaite, ni complètement équitable. Les techniques produisant des solutions exactes comme la programmation linéaire ou les méta-heuristiques classiques, recuit simulé, recherche tabou, algorithmes génétiques ont été éliminée à cause de leur lenteur. Les tâches pouvant s'exécuter sur les ressources de calcul étant très différentes et ayant des profils d'utilisation des ressources hétérogènes, nous ferons une hypothèse simplificatrice et une représentation abstraite des taches. Nous supposerons que toutes les tâches peuvent se décomposer en une succession de transfert de fichiers et une exécution sur les ressources de calcul, c'est à dire, d'abord une utilisation du réseau puis d'un calculateur. Nous supposerons de plus que les tâches ne peuvent pas être retardées, elles sont acceptées immédiatement ou rejetées. Trois algorithmes seront proposés pour modéliser le comportement de chacun des trois services: *Legacy*, *BoD*, *XO*. *XO* va chercher à minimiser le temps de traitement vu par l'utilisateur, c'est à dire la somme du temps passé dans le réseau et dans de calculateur. Nous avons voulu nous affranchir d'une topologie réseau donnée et avons choisi un angle plus générique afin d'identifier les paramètres réseau pertinent. Nous avons donc étudié une famille de topologie aléatoire. L'objectif étant de mieux comprendre les mécanismes d'optimisation dans la co-allocation et compte tenu du grand

nombre de paramètres à gérer et à appréhender, nous avons préférer décomposée l'étude en trois étapes: une étude analytique, simulateur simplifié et simulation à évènements discret. Malheureusement, nous n'avons pas réalisé la simulation à événement discret, nous avons simplement spécifié l'algorithme d'optimisation croisée dans ce cadre (voire l'annexe 4.6). Le simulateur simplifié prends un nombre fixé de job comme paramètre d'entrée et détermine l'allocation pour ces jobs des ressources. Les phénomènes prenant en compte la libération des ressources et l'allocation de nouveaux jobs ne sont donc pas pris en compte. Toutefois, l'intérêt de la co-allocation sera malgré tout démontré. De plus, l'intérêt de cette démarche par étape nous permet une compréhension plus fine des phénomènes et des paramètres clés. Enfin, les résultats mettent en évidence des indicateurs comme la capacité équivalente du réseau ouvrant la porte à d'autres types d'outil d'optimisation comme les outils liés aux problèmes de capacité (sac à dos).

Nous allons maintenant décrire le modèle en détail, ensuite, nous présenterons les résultats des travaux analytiques et enfin des simulations. Nous allons décrire les choix faits pour représenter les jobs, les ressources de calcul, le réseau et les différents algorithmes d'allocation. La figure 4.3 et la figure 4.4 de la page 99 décrivent le modèle de tâche utilisé. Nous supposons que toute tâche plus complexe soit décomposée en tâche élémentaire de ce type: un transfert de fichier suivi d'un traitement. L'exemple le plus simple étant le traitement de donnée. Si les données de sorties sont de taille négligeable par rapport aux données d'entrée, le modèle correspond bien, sinon, nous devrons supposer que les taches les plus complexes comme le transfert de donnée d'entrée suivi d'un traitement et du renvoi de donnée de sortie doivent être décomposées en deux dans notre modèle. La deuxième tache étant l'envoi des données de sortie sans calcul à réaliser. Nous supposerons que les deux taches sont de plus complètement indépendantes, c'est là une hypothèse simplificatrice discutable. Chaque job j partira d'un nœud source, il est décrit par une quantité de donnée à transférer $S(j)$ et une quantité d'instructions à exécuter $D(j)$, un débit d'accès $A_r(j)$. La quantité d'instructions à exécuter peut se mesurer par une durée d'exécution sur une machine de référence, elle peut aussi être une borne supérieure de la durée réelle d'exécution. Cette information peut être obtenue pendant la phase d'installation du service de calcul ou les tâches sont lancées sur la grille pour vérifier le bon fonctionnement du système. En utilisant des benchmarks [61, 62, 63], il est possible de comparer la puissance de différentes machines. Les variables $S(j)$, $D(j)$, $A_r(j)$ sont aléatoires indépendantes et identiquement distribuées. Nous avons choisi pour $A_r(j)$ une distribution uniforme entre 10Mbit/s et 10Gbit/s pensant que cela représentent les débits d'accès futurs. Chaque job nécessite un et un seul CPU. En pratique, dans les systèmes de calcul parallèle, les jobs nécessitent plusieurs CPU, néanmoins le modèle peut s'étendre sans difficultés pour prendre en compte ce type de jobs. Les distributions de $S(j)$ et $D(j)$ sont uniformes dont les moyennes sont déterminées de façon à ce que le temps passé dans le réseau et dans le calculateur sont du même ordre de grandeur. Le réseau est modélisé par un graphe généré aléatoirement, c'est à dire un ensemble de nœuds et de liens. Les liens possèdent une capacité exprimée en Gbit/s. La capacité sur chaque lien sera distribuée aléatoirement. Elle peut être soit discrète soit uniforme. La figure 4.5 de la page 101 illustre le modèle retenu pour le réseau. La génération aléatoire de topologie qui représente bien les réseaux de télécommunications est un problème complexe qui a été largement étudié [64, 144, 48]. Les graphe de Waxman [155] forment une famille de graphe aléatoires qui modélisent

bien les réseaux à l'intérieur d'un système autonome. L'idée consiste à fixer le nombre de nœuds du réseau, puis de décider selon une loi binomiale si un lien doit être créer entre deux nœuds. C'est à dire, que l'on décide avec la probabilité p_{ij} de créer un lien entre le nœud i et le nœud j, et la probabilité $1 - p_{ij}$ de ne pas le créer. p_{ij} décroissant avec la distance et déterminé par la formule 4.1 de la page 101. C'est à dire que plus les nœuds sont éloignés les uns des autres, plus faible sera la probabilité de créer un lien entre eux. Les paramètres α et β et le nombre de nœuds détermine [150, 109] le degré moyen de chaque nœud (nombre de voisins). Ainsi il nous sera possible de générer un ensemble de topologie aléatoires ayant un commun le nombre de nœuds et le degré moyen de chaque nœud. La figure 4.6 de la page 101 donne un exemple de deux graphe de Waxman de 20 nœuds pour deux degrés moyens théoriques différents: 2,5 et 3. Plusieurs clusters de CPU sont répartis dans le réseau, chacun rattaché à un nœud du réseau. Chaque cluster possède un nombre de CPU déterminé de façon à ce que la somme des CPU sur le réseau soit égale à un paramètre d'entrée de la simulation. Le nombre de CPU par cluster est distribué uniformément. Au sein d'un même cluster tous les processeurs ont la même puissance de calcul. La puissance de calcul de chaque CPU est une variable aléatoire distribuée uniformément.

Trois algorithmes d'allocation modélisent les trois types d'interactions entre l'ordonnanceur de la grille et le gestionnaire du réseau. Le premier algorithme *Legacy* alloue les jobs aux ressources quand aucune interaction ne permet la réservation de ressource réseau. C'est la situation actuelle quand des services de calcul sont utilisés sur Internet. Les jobs sont d'abord alloués à la ressource de calcul par l'ordonnanceur puis transférer sur le réseau. L'algorithme est décrit à la page 103: algorithmes 1 et 2. Son fonctionnement est le suivant. Les jobs sont traités dans un ordre donné, les CPU classés par ordre de puissance de calcul décroissante. Le premier job est affecté au premier CPU, le second job au deuxième CPU, et ainsi de suite. C'est à dire que les jobs sont affectés au processeur disponible le plus puissant. Une fois l'ensemble des jobs affectés, il faut déterminer quelles ressources réseaux vont être exploitées et comment celle-ci va être partagée entre les jobs. Nous avons supposé un routage statique reposant soit sur une minimisation du nombre de hop soit une maximisation de la bande passante pour chaque couple (source, destination). Deux questions importantes doivent être traitées, comment la bande passante est partagée entre les différents jobs et que se passe-t-il s'il y a "congestion"? Nous avons apporté quelques réponses afin de concevoir un algorithme d'allocation modélisant le comportant du réseau tout en préservant la simplicité du modèle. De plus, il s'agit de l'algorithme modélisant le comportement existant que nous savons a priori mauvais, nous avons parfois sur évaluer son comportement. L'idée de l'algorithme une fois les CPU déterminés consiste à faire passer les jobs sur le routage statique en partageant équitablement la bande passante tout en prenant en compte les débits d'accès différents, tout en évitant de saturer les liens qui appartiennent à deux routes différentes.

L'algorithme *CCB* ou *BoD* dit de connexion à la demande modélise l'allocation des jobs aux ressources quand l'ordonnanceur de la grille peut envoyer des requêtes au gestionnaire du réseau de type bande passante à la demande. Dans la variante *CCB*, si le gestionnaire du réseau ne peut pas établir de connexion vers le site demandé, l'ordonnanceur de la grille envoie une deuxième requête de connexion vers une autre destination pour tenter de traiter le job en cours. C'est la variante "CrankBack". Dans la variante sans "CrankBack" *CnoCB*,

l'ordonnanceur de la grille ne cherche pas à traiter le job sur une autre destination que celle initialement prévu si le réseau ne peut pas établir de connexion. L'algorithme *CCB* est décrit à la page 105. Le fonctionnement de l'algorithme est le suivant. Les jobs sont traités séquentiellement, chaque job est affecté dans l'ordre au CPU le plus puissant disponible. Une fois la localisation du CPU disponible le plus puissant est connue, l'ordonnanceur de la grille demande au gestionnaire du réseau d'établir la connexion avec la bande passante correspondant au débit d'accès du job. Si il est possible d'établir une connexion, le job est alloué avec la bande passante maximum que le réseau peut établir vers cette destination. Si le réseau ne peut établir de connexion, dans la variante Crankback, l'ordonnanceur de la grille allouera le job au deuxième CPU le plus puissant et réitèrera le processus, dans la variante no crankback, le job ne sera pas alloué et l'ordonnanceur de la grille passera à l'allocation du job suivant.

Le dernier algorithme va allouer les ressources quand l'ordonnanceur de la grille accède à un maximum d'information du réseau, c'est à dire ici l'état d'utilisation de la bande passante sur chaque lien. Différentes fonctions objectives peuvent être considérées. Parmi elle, la minimisation de la somme des temps de traitement sur l'ensemble des jobs serait intéressante (voir l'équation 4.2 de la page 105). Ayant choisi de développer un algorithme qui traite les jobs à la volée, et voulant démontrer dans un premier les bénéfices de la co-allocation, nous avons choisi un objectif plus simple: minimiser le temps de traitement du job en cours comme si il était seul à devoir être alloué. L'algorithme retenu est donc égoïste et ne tiens pas en compte de l'équité entre les jobs, les premiers arrivés sont les premiers servis. Les jobs sont traités de manière séquentielle, pour chaque jobs, l'ensemble des destinations de traitement disponible sont déterminées, ensuite pour chaque destination la route de bande passante maximale est déterminée. Ensuite, le temps de traitement que le job mettrait s'il était alloué à chaque couple (cluster de destination, route de bande passante maximale) est calculé et l'ordonnanceur sélectionne le couple de ressources qui minimise le temps de traitement. Cette algorithme ne permet pas l'allocation de tous les jobs, il est possible que certains n'aient plus de ressources disponible. Pour implémenter efficacement cet algorithme, nous avons étendu le réseau tel décrit dans la figure 4.7 afin de pouvoir utiliser des algorithmes de recherche de chemin de coût minimal classique. L'implémentation réelle cherche la route de bande passante maximale, mais ce problème peut être ramené à un problème classique de recherche de route de coût additif minimal (il suffit valuer les liens par l'inverse de leur bande passante disponible). L'algorithme *XO* porte le numéro 4, il est décrit à la page 107. Le lien entre l'algorithme proposé et le routage multicritère est développé dans la thèse. L'algorithme *XO* réalise un routage multicritère et détermine l'ensemble des routes optimales au sens de Pareto, la route finale étant choisi selon la pondération donnée par chaque jobs.

L'ensemble du modèle est probabiliste, nous pouvons donc essayer de prédire la performance de la co-allocation. Les deux critères de performance étudiés sont le nombre de jobs alloués et le temps moyen de traitement d'un job. Dans un premier temps nous allons essayer de déterminer le nombre moyen de job alloué. Imaginons un réseau réduit à deux nœuds et un lien et les demandes de traitement réduites à des demandes de bande passante. Soit C la variable aléatoire décrivant la capacité du lien, $A_r(i)$ celle décrivant la quantité de bande passante demandée par la requête i. Supposons que les $A_r(i)$ sont indépendants et identiquement distribués. Le nombre de requête qui peuvent être allouée

sur le lien de capacité C est un nombre aléatoire. Si on note $E(X)$ la valeur moyenne de la variable aléatoire X. Le nombre moyen de requête acceptée sur le lien est $\lceil \frac{E(C)}{E(A_r)} \rceil$. C'est une application directe de l'identité de Wald [75] (page 233). Nous appellerons la quantité $\frac{E(C)}{E(A_r)}$ capacité du lien normalisé. Le théorème 1 de la page 108 reprends l'énoncé de l'identité de Wald appliquée au temps d'arrêt $T(\omega) = inf\{n, \sum_{i=1}^{n} A_r(i)(\omega) > C(\omega)\}$. Quand le réseau est composé de deux nœuds et de plusieurs liens, le résultat ne change pas, il suffit de remplacer C par pC où p est le nombre de liens. Quand le réseau est composé de plusieurs nœuds et de plusieurs liens, le nombre de job que l'on peut allouer n'est pas aussi facile à calculer, cela dépends beaucoup du nombre de nœuds, de liens, du degré moyen, et de l'algorithme d'allocation. Le problème se complexifie encore si plusieurs sources et destinations sont possibles. Toutefois, si nous restreignons notre étude à une source et une destination et à un algorithme d'allocation qui utiliserait le routage déterminé par un algorithme de recherche du flot maximum, nous pouvons calculer le nombre moyen de job alloué dans le réseau. En effet, Lee [98] a étudié le problème du flot maximum dans un graphe aléatoire de Waxman entre deux nœuds arbitraires. Plus précisément il a supposé que les liens du graphes avait une capacité unitaire de 1 et qu'un flot pouvait soit utiliser la totalité de la capacité du lien soit ne pas utiliser le lien. Le problème de recherche du flot maximum est alors équivalent à rechercher le nombre de chemin disjoints entre la source et la destination puisque deux chemins ne peuvent pas utiliser le même lien. Par ailleurs, le théorème Max-Flot/Min-Cut nous dit que le flot maximum est égale à la capacité de la coupe minimale, c'est à dire ici, au nombre de lien de la coupe minimale. Rappelons qu'une coupe est une partition de l'ensemble des nœuds en deux (une sous partie devant contenir la source et l'autre la destination) et que la capacité de la coupe est la somme des capacités des liens joignant les deux sous ensembles de la coupe. Les résultats de Lee sont résumé dans le théorème 2 de la page 109: "Le flot maximal est en moyenne proportionnel au minimum entre le degré de la source et de la destination". Le résultat de Lee a été prouvé expérimentalement sur les graphes de Waxman entre une source et une destination. Nous pensons que ce résultat peut se généraliser selon la conjecture 1 de la page 109 entre plusieurs sources, plusieurs destinations et étant donné un algorithme d'allocation. Cette conjecture énonce que le nombre de job que l'on peut allouer est décrit par l'équation 4.5 de la page 109, c'est à dire qu'il est proportionnel à la capacité normalisé des liens. De plus, si l'on note κ le coefficient de proportionnalité, on peut définir la capacité équivalente du réseau comme étant $\kappa E(C)$ où C est la variable aléatoire donnant la capacité d'un lien. La conjecture n'est prouvée que dans le cas où le réseau est composé dŠune source et une destination et uniquement pour l'algorithme d'allocation utilisant le routage de Max-Flot. En effet, en appliquant l'identité de Wald à la coupe minimale la capacité de celle ci est en moyenne le produit du nombre moyen de lien la composant et de la moyenne des capacités. Or la capacité de la coupe minimale permet d'obtenir le nombre de job alloué qui utiliserait le routage de Max-Flot par application de l'identité de Wald au temps d'arrêt donnant le nombre de job sur la coupe minimale. Il suffit de remplacer C dans la formule par la capacité de la coupe minimale. Ainsi, on obtient que le coefficient κ de Max-Flot de la conjecture est proportionnel au degré moyen du graphe. La validité de la conjecture 1 sera vérifiée par des simulations pour les autres algorithmes d'allocation. Le fait de ne pas avoir à traiter exactement des requêtes de bande passante à

la demande mais plutôt des demande de transfert respectant le débit d'accès n'a pas une importance significative compte tenu du fait que nous cherchons à allouer le maximum de bande passante disponible. L'autre critère de performance est le temps de traitement moyen d'un job. Si nous faisons quelques hypothèses sur les distributions des paramètres des jobs allouées, notamment en supposant quŠelles sont les mêmes que celles des jobs avant allocation, nous pouvons calculer analytiquement la valeur moyenne du temps de traitement des jobs. En prenant des distributions uniformes, la formule 4.6 de la page 110 donne le résultat. Nous verrons par la suite que cette formule est proche de la réalité quand on travaille dans des réseaux dont la capacité des liens est au moins 20 fois plus grande en moyenne que le débit d'accès moyen.

Les travaux de simulation de cette thèse portent sur la vérification de la conjecture 1 de la page 109. Nous vérifions que la capacité normalisée est bien le paramètres important à prendre en compte. L'importance du nombre de CPU est ensuite étudié. Enfin, les derniers travaux portent sur la vérification de la prédiction du temps de calcul moyen donné dans la formule 4.6 de la page 110. La figure 4.8 de la page 111 illustre l'importance de la capacité normalisé. Nous avons conduit différentes allocation de jobs en maintenant constant le rapport $E(A_r)$ sur $E(C)$, le nombre de job alloué est resté le même. La figure 4.9 de la page 112 trace le nombre de job accepté par chacun des trois algorithmes en fonction de la capacité normalisée. Comme attendu par la conjecture 1, nous trouvons une relation linéaire. Les courbes ont été obtenues pour différentes familles de réseau: 10, 20 et 30 nœuds en étudiant des degrés moyens de 2,5 et 3. Chaque point est obtenu avec un intervalle de confiance de moins de 2%. 1000 jobs ont été soumis sur 2000 CPUs. Ce graphe illustre donc ce qu'il se passe quand le réseau est le goulet d'étranglement. Des courbes similaires ont été obtenues avec 1000 jobs soumis sur 1000 CPUs et 1000 jobs soumis sur 500 CPUs. La différence entre les courbes tracée est donc la pente qui dépend de l'algorithme d'allocation, du dégrée moyen, du nombre de nœuds, de la distribution des capacité des liens, du ration nombre de job soumis sur nombre de CPU disponible. La conclusion à tirer est que par rapport à un service de bande passante à la demande, dans les réseaux de petite taille, l'algorithme d'optimisation *XO* proposé est trop simpliste et ne permet pas de réaliser une optimisation croisée des ressources. En revanche pour les réseaux plus gros, un gain d'environ 10% sur le nombre de job acceptés peut être observé par rapport au service de bande passante à la demande. La figure 4.10 de la page 113 trace le nombre de job alloué en fonction du ratio nombre de job soumis sur nombre de CPU, à nouveau pour les 6 familles de topologies, et ce pour une capacité normalisée de 16. Quand ces courbes sont plates, cela illustre que le réseau est le goulet d'étranglement, quand elles croissent avant de s'aplatir, cela illustre qu'il existe un régime (pendant la croissance) où le réseau n'est pas l'unique limitateur de performance. Les courbes pour l'algorithme *Legacy* n'ont pas été tracée car par définition cet algorithme accepte tous les jobs, l'impact de ce dernier est uniquement sur le temps de traitement qui devient dramatiquement long. La figure 4.11 de la page 114 montre le temps de traitement moyen des jobs pour les trois algorithmes en fonction de la capacité normalisée. La ligne horizontale représente la valeur théorique. La précision de nos simulations de Monte-Carlo ne permet pas d'avoir des intervalles de confiance suffisamment fin qui nous permettent de comparer entre les trois algorithmes. Ce que nous pouvons bien sûr dire est que le temps moyen de traitement des jobs alloué par l'algorithme *Legacy* est hors échelle par un

facteur 100. La dernière partie des résultats des simulations discute une hypothèse faite dans la partie analytique, l'hypothèse s'avère valide, nous ne reprenons pas la discussion ici.

Un des objectifs de ces simulations était de démontrer l'avantage de l'optimisation croisée par rapport aux systèmes de bande passante à la demande. Un des résultats principaux est que pour les réseaux de petite taille, il n'est pas nécessaire d'aller plus loin qu'un système de bande passante à la demande. Une autre conclusion serait de dire que l'algorithme proposé est trop simpliste et ne permet pas de démontrer les bénéfices de l'optimisation croisée. Un résultat majeur de cette thèse a été de montrer que le nombre de jobs alloué dans le réseau pouvait être prévisible et suit une loi simple de proportionnalité par rapport à la capacité normalisée. Une autre façon de voir ce résultat consiste à définir la capacité équivalente du réseau en nombre de job. Ce résultat est important car un tel indicateur permet de concevoir des algorithmes d'allocation en considérant directement cet indicateur de performance. Une borne inférieure du temps moyen d'exécution peut être calculée analytiquement. Quand la capacité normalisée augmente cette borne est presque atteinte par les algorithmes d'allocation de type *XO* ou *CCB*. Le nombre de job allouée est déterminé quand le réseau était initialement vide jusqu'à temps que les ressources se remplissent et saturent. Il serait intéressant d'étendre l'étude menée ici dans le cadre d'un simulateur à événement discret qui prendrait en compte la libération des ressources par les jobs et l'arrivée de nouveaux jobs. La vérification de l'hypothèse suivante serait intéressante: "Dans le cadre d'une simulation à événement discret, le système est équivalent à une file d'attente de longueur la capacité équivalente et de temps de service moyen le délai moyen de traitement calculé analytiquement".

Conclusion

La co-allocation de ressource réseau et de calcul n'est possible que si la technologie du réseau est capable de réserver les ressources. Grace notamment à la réservation de ressource, la garantie de qualité de service à été introduite progressivement dans les réseaux IP. Suite aux travaux d'Intserv, l'architecture Diffserv a été un pas supplémentaire vers la garantie de QoS, sans fournir de réservation explicite. C'est uniquement avec l'architecture MPLS et son extension GMPLS qu'une garantie de QoS peut être offerte aujourd'hui par le biais de réservation de ressource explicite.

Les applications fonctionnant sur une Grille nécessitent une virtualisation des ressources. Ces dix dernières années ont vu le nombre et la diversité des applications pouvant bénéficier de la virtualisation des ressources augmenté incroyablement. Si les premières études dans le domaine ont été confinées au monde de la recherche scientifique, les prochaines années verront sans doute de nombreuses applications commerciales. En ce sens les grilles constituent un marché à fort potentiel pour les opérateurs de télécommunications. Une nouvelle entité, le courtier en ressource, représente une opportunité commerciale importante tant pour les fournisseurs de services de télécommunication que de calcul. La virtualisation garantissant de forte QoS ne peut être réalisée sans co-allocation. L'émergence des standards comme GMPLS permettant cette QoS garantie et l'ingénierie de trafic ont été un facteur clé de la co-allocation. Aujourd'hui, de nombreux problèmes encore sont

à résoudre pour exploiter GMPLS dans les grilles. Tout d'abord, il reste à déterminer comment GMPLS peut être intégré aux systèmes de gestion des grilles. Un deuxième problème est lié à la façon dont la garantie de QoS peut être fournie dynamiquement, ce qui sort du contexte traditionnel d'utilisation de cette technologie. De plus, les grilles ont aussi besoin de gérer efficacement les ressources de stockage et de calcul distribuées dans le réseau. En dŠautres termes, les grilles nécessitent le développement d'outils permettant une optimisation globale des ressources à la fois réseau de calcul et de stockage.

Des réponses à ces questions ont été fournies dans cette thèse. Dans le chapitre 1, nous avons fourni une revue de l'état de l'art dans la matière. Nous avons décrit les principaux projets de recherche portant sur la co-allocation entre 2004 et 2007. Le chapitre 2 a présenté les principales architectures pour intégrer le réseau aux ressources de calcul. Les approches existantes ont été étudiées: Web service et Grid GMPLS. Nous avons proposée une nouvelle approche en utilisant l'architecture IMS et SIP. Nous avons proposée des extensions au plan de contrôle réseau actuel pour prendre en compte les réservations de ressources à l'avance. Deux publications [117, 118] et quatre brevets [111, 112, 114, 151] ont été acceptés comme contribution.

Le chapitre 3 a étudié plus en détail les protocoles en cours de développement et de standardisation pour fournir la co-allocation par le biais de SLA. Les protocoles de négociation et de création de SLA en cours de développement ont été étudiés. Plus particulièrement le protocole de création de SLA WS-Agreement proposé par l'Open Grid Forum a été analysé. Avec Olivier Warldrich, nous avons proposé une extension de ce protocole pour négocier les SLA. Ces travaux ont fait l'objet de deux publications [119, 120]. Cette thèse à proposé un modèle à base de file d'attente pour comparer les protocoles de négociation actuelle avec nos extensions permettant une confirmation de la réservation. Nous avons quantifié l'impact de nos extensions par rapport au protocole actuel et conclu à un besoin en mémoire au niveau de l'ordonnanceur doublé.

Le chapitre 4 a analysé jusqu'où il est nécessaire d'aller dans l'interaction entre l'ordonnanceur de la grille et le gestionnaire du réseau. Est-ce que les ressources seraient mieux utilisées si l'ordonnanceur de la grille avait accès à des informations de topologie et de bande passante disponible sur les liens en temps réel? Nous avons proposé un modèle d'optimisation croisée pour répondre à cette question. Nous avons développé un modèle analytique et des simulations pour évaluer la performance de notre algorithme comparée à ceux modélisant un service de bande passante à la demande et la situation actuelle où le réseau est une boite noire sans fournir de réservation. Un des résultats principaux de ce chapitre est que le nombre de job alloué dans le système est proportionnel à la capacité normalisée des liens. Nous avons souligné qu'il est possible de définir une capacité système de la grille en nombre de job ou d'une capacité réseau équivalente en Mbit/s. Un des autres résultats principaux est que dans les réseaux de moins de vingt nœuds, il n'est pas nécessaire d'aller plus loin qu'un service de bande passante à la demande. Toutefois dans les réseaux plus gros, une optimisation croisée augmente le nombre de jobs accepté d'environ 20%. Ces résultats de simulation ont permis de valider les résultats analytiques obtenus. La capacité équivalente de la grille calculée dans le chapitre 4 permet de concevoir d'autres types d'optimisation des ressources et d'imaginer d'autres heuristiques. Les résultats de ce chapitre font l'objet d'une publication [116] et d'un brevet [115].

Les travaux futurs sur la cross optimisation sont encore à réaliser. Pour des raisons de

temps, nous n'avons pas pu vérifier notre modèle dans un environnement plus dynamique reposant sur un simulateur à événements discrets. Cette thèse peut être considérée comme un point de départ sur les travaux d'optimisation croisée dans les grilles.

Contents

List of Figures

Acronyms

2PNP 2 Phase Negotiation Protocol

3GPP 3rd Generation Partnership Project

3PNP 3 Phase Negotiation Protocol

ARGON Allocation and Reservation in Grid Optical Network

AS Autonomous System

ASP Application Service Provider

ATM Asynchronous Transfer Mode

BGP Border Gateway Protocol

BoD Bandwidth On Demand

BR Border Router

CA Commit Agreement

CCB Connection with Crankback

CnoCB Connection without Crankback

CPU Central Processing Unit

DoIA Declaration of Intention Agreement

DOM Document Object Model

$E()$ Mathematical Expectation

ETSI European Telecommunication Standardisation Institute

FSM Finite State Machine

GGMPLS Grid GMPLS

Globus Name of a grid toolkit

GMPLS Generalized Multi Protocol Label Switching

GridEngine Name of a grid toolkit

GridFTP Name of a File transfer program

GridWay Name of a Grid scheduler

GUNI Grid User to Network Interface

I-CSCF Inter - Call/Session Control Function

IETF Internet Engineering Task Force

IMS IP Multimedia Subsystem

INPI Institut National de la Propriété Intelectuelle (French patent office)

IP Internet Protocol

ISP Internet Service Provider

IT Information Technology

LRMS Local Resource Manager

LSP Label Switched Path

MPLS Multi Protocol Label Switching

MSS Meta Scheduling System

NGN Next Generation Network

NJS Network Job Supervisor

NMS Network Management System

OEB Office Européen des Brevet (European Patent Office)

OGF Open Grid Forum

OGSA Open Grid Standard Architecture

OSPF Open Shortest Path First

OSS Operation Support Systems

PA Preparation Agreement

PCC Path Computing Client

PCE Path Computing Element

P-CSCF Proxy Call/Session Control Function

PDF Policy Decision Function

QoS Quality of Service

REST Representational State Transfer

RFC Request For Comment

RMS Resource Management System

RSVP Reservation Protocol

S-CSCF Server - Call/Session Control Function

SDP Session Description Protocol

SGE Sun Grid Engine

SIP Session Initiation Protocol

SLA Service Level Agreement

SOA Service Oriented Architecture

SOAP Simple Object Access Protocol

TE Trafic Engineering

TEDB Trafic Engineering Database

TISPAN Telecommunications and Internet Converged Services and Protocols for Advanced Networking

TSI Target System Interface

UNI User to Network Interface

Unicore Unform Interface to computing resource (it's a Grid Toolkit)

VIOLA Vertically Integrated Optical Network for Large application

WS Web Service

XML Extensible Markup Language

XO Cross Optimisation

Introduction

Several experimentations such as the SETI@HOME [22, 40] program consisting in the interpretation of cosmic radiations require huge memory space and computational resources. The design of ad-hoc super-computers able to solve such problems is today unrealistic, both for technical and economical reasons. This is why a great interest has been dedicated these last ten years to distributed computing. Depending on the nature of the problem to be solved and on the available budget of the end-users, distributed computing appears in many cases as the only solution. Distributed computing may concern the federation of tens, hundreds or even thousands computers either co-located on the same premises or distributed at the regional scale or even at the national or international scale. Distributed computing relies on the design of a virtual infrastructure to interconnect remote computers in order to provide in certain cases an alternative to the largest super-computers now available on the market [39]. The examples of a massively distributed computing infrastructure are the most popular [141, 39]. Distributed computing may require the sharing of different types of resources such as computing facilities, storage facilities [121, 94] and networking facilities. Concerning the networking facilities, distributed computing may rely on peer-to-peer connections (P2P).

All the distributed computing projects mentioned above, although of a great interest, are not in the scope of this thesis since they do not offer any quality of service (QoS) guarantees to the end-users. In this thesis, we investigate the problem of dynamic resource co-allocation in distributed systems over long distance networks. The objective of dynamic co-allocation is the provision of services requiring the satisfaction of QoS guarantees such as minimum bandwidth, upper bounded end-to-end transit delay, residual bit error rate or the respect of a given scheduling between the various elementary operations to be carried out on the remote devices. Along this thesis, we outline how some of the sub-problems inherent to distributed computing may be generalized in order to be adapted to the provision of various types of service.

The term “Grid” has been introduced to describe an infrastructure integrating several resources (computing, storage, network, ...) belonging to different administrative entities. These resources can be spread across a wide area network and used cooperatively in order to solve complex problems. The Grid concept [72, 70, 73] implies coordinated resource sharing via multi-institutional virtual organizations. In this context, resource sharing is not limited to file exchange but also concerns direct access to remote computers,

software, data, storage and networking facilities. A wide range of collaborative problems requiring very specific QoS guarantees may benefit of the Grid concept. A great variety of industrial or academic sectors are directly concerned by Grids. For instance, in genomic or in bio-technology, companies or laboratories expect that Grids will facilitate their strong reactivity in a highly competitive research environment. Indeed, the design of new molecules requires intensive computing and storage facilities. Such facilities are not in general easily affordable for start-ups or SMEs (Small or Medium Enterprizes). Since Grid services may require resources managed by different administrative entities, resource providers as well as the end-users (the consumers of Grid services) must clearly define which resources have to be shared, who is administratively responsible of resource sharing and the conditions under which this sharing occurs. For that purpose, the establishment of Virtual organizations (VO) including customers, network carriers and third parties accepting to re-sell some of their computing and/or storage facilities is necessary. In summary, a Grid is an infrastructure satisfying the following four characteristics:

- It is based on heterogeneous [1] resources.
- These resources are interconnected via a public network, either in the local loop or at the metropolitan area level or via a core network.
- The resources belong in general to different institutions.
- These institutions must work cooperatively in order to solve a specific problem or to provide a given service

From these four characteristics, we understand why a cluster of computers co-located in the same private premises cannot be considered as a Grid. A "Grid toolkit" is a software program that offers features and functionalities in order to build a virtual infrastructure characterized by the four bullet points mentioned above. From a telecommunication operator's perspective, a computing service must satisfy the following requirements:

- Reliability
- Commercial viability
- Guaranteed Quality Of Service (including Network QoS)
- Security

Multiple examples of large computational problems may be found in the research, the industrial and the financial environment. Scientific techniques like Monte-Carlo simulations or parametric sweep simulations are now being used in production, design and research environments, such as:

1. For example, computing resources with different processor architectures and vendors

- Electronic design automation
- Bio-informatics and pharmaceuticals drug design
- Mechanical computer-aided engineering
- Computational fluid dynamics and finite element analysis
- Oil, gas and Seismic data analysis
- Financial analysis, real-time risk analysis, cash-flow analysis and Monte Carlo simulations
- Software development, code design, compilation and links
- Computer animation rendering and digital content creation

The emergence of these examples of applications requiring massive computing and storage devices combined with the availability of high capacity networks requires the conception of new services accessed via a carrier's network. In practise, networks managed by different operators may be implied in the provision of a Grid service. In general, Grid services are referred as "computational services". On one hand, these various applications do not share the same characteristics. For instance, some of them require an almost permanent bandwidth guarantee, whilst others only need to transfer a file as quickly as possible. On the other hand, the common point between these applications is that for each of them, the end users have a common economical interest in sharing both remote computing and storage resources and network infrastructure costs. Doing so requires the emergence of new business models. The predominant idea in this matter consists in computing power rental, i.e. a user would rent access to a computing infrastructure and pay proportionally according to its use ($/h/CPU).

Unifying frameworks

To provide computing (or storage) services that take into account network QoS, a network operator must be involved in the service delivery process. Some operators might choose to buy a computing infrastructure, whilst others might rent or lease it. Some computing service providers may also want an operator to provide "Bandwidth on demand" services or "QoS on-demand" services, limiting the role of the operator to a connectivity provider. All these scenarios can be unified if the following framework is considered: telecommunications operators (or network operators) are composed of two departments, a network department and a computing department. In practice, either the same carrier, or two distinct administrative entities manage these two departments. The network department is in charge of maintaining the network and provides connectivity. The computing department manages computing resources and provides computing services. The network department's customers can be either end users or the computing department.

The computing department's customers are always end users. Despite the existence of several resource owners, the computing department provides a common interface to the end users. The heterogeneity of the available resources is not visible by the end users. It runs a what is called "Grid middleware" (or Grid toolkit). The term Resource Management Software (RMS) will be preferred to the expression Grid middleware in the remaining of this dissertation. Although in pratice, the network department may be a customer of the computing department; this configuration will not be considered in the following. The owners of the shared resources are the fourth and last administrative entity concerned in the Grid unifying framework. They manage their own resources and facilities. They provide an interface to their customers or partners, such as the computing department. In summary, the four main actors considered by the unifying framework are:

- The end-users, mostly customer companies that ask for the resolution of a specific computing problem.
- The network department within a telecommunication operator (or belonging to different operators in the case of multi-domain environment).
- The computing department within (or not) a telecommunication operator
- The resource providers.

Business relationships between the four actors listed at the end of the previous section is materialized by service contracts between a service user and a service provider. Each service's description, terms, constraints and guarantees is described in a Service Level Agreement (SLA) negotiated between the two parties. An SLA [21] is a formal negotiated agreement between two parties. It is a contract that exists between customers and their service provider, or between different service providers. It records the common understanding about services, priorities, responsibilities, guarantee, etc. with the main purpose to agree on the level of service. For example, it may specify the levels of availability, serviceability, performance, operation or other attributes of the service like billing and even penalties in the case of violation of the SLA. Historically, fixed line telecom operators as part of their contracts have used SLAs since the late 80's with their corporate customers. More recently, IT departments in larger companies have adopted the idea of using service level agreements with their customers, i.e. users in other departments within the same company. Such an approach facilitates the comparison between the expected QoS and the effective provided QoS. Depending on the gap between these two, the alternative of outsourcing IT services to an external company may be then decided. An SLA is generally business oriented and does not go into much technical detail. Its technical specifications are commonly described through a Service Level Specification (SLS). SLAs have been widely studied [119, **?**, 106, 105, 91, 101, 59] in the context of both telecommunication networks and Grid systems. The services that can be provided to the end user or to a computing department are the following ones:

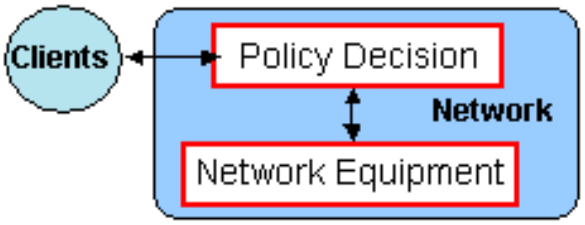
- Connectivity on Demand with guarantees like Bandwidth on demand or low latency connectivity
- Computing service on demand
- Integrated Computing Service
- Storage services on demand
- Resource orchestration services on demand

Main service types

- **On-demand connectivity with SLA:**

  This service provides a guaranteed QoS connectivity. It encompasses Bandwidth on demand or Low latency on demand that are requirements for most Grid applications [134]. The two most important features required to provide a computing service are: advance reservation and SLA provisioning. Additional features such as security, failure notification, resource monitoring can also be expected to be provided. Some telecommunication equipment vendors are already selling such devices [16] (DRAC). Diagram 1 shows the simplified architecture for such a system.

Figure 1: Connectivity on Demand



  The main research problem in matter of on-demand connectivity with SLA refers to the design of policies that maximize the utilization of network resources alone. Such policies enable to minimize the number of rejected requests because of resource shortages. Several investigations have tackled these issues [82, 124, 157, 86]. In this thesis, we extend the problem of resources optimization in considering both network and computing resources. Our objective is then to select network and computating resources in order to minimize the overall processing time to provide the required service.

- **Computing service on demand:**

A computing service can provide access to resources at three different levels. At a first level, a raw computational resource access under which the end users have to provide information about the Operating-System and the running programs. The Operating-System is loaded upon every new job request. The second level refers to computational services. The end-users have in this case to provide the source code or the binary code to be executed on a given platform. The third level refers to application services. In that case, the end-users have to provide input files or they simply interact directly with the application. The third category of users can also interact with an intermediary who will then use services provided by the computing department. In other words, operators should focus on satisfying the first two categories of users, starting with the first one. Other actors will build on top of these solutions to provide application access services. In other words, the role of the computing department is as a wholesaler. However, several aspects remain to be specified:

- Architecture
- Heterogeneity of computational resources providers
- Automatic discovery of resources
- Efficient use of resources
- Scalability with the number of resources and the number of users
- Network QoS guarantees
- SLA enforcement mechanisms
- Pricing of a bundle of multiple resources

Each of these problems is in itself quite complex. This is why this thesis focuses mainly on architectural issues, on performance issues and on one aspect of the efficient use of resources.

- **Integrated Computing Service:**

  An integrated computing service is a computing serv ice that provides QoS guarantee taking into account network and computing resources. Both network and computing resources are bundled and proposed to the user.

- **On-demand storage services :**

  Facing exceptional events as natural disasters, many companies wish to be able to restore in very short delays the status of their data bases. For that purpose, they use public storage services provided via carrier's networks. The necessity to restore in short delays huge amount of data stored at different sites via a networking infrastructure may be seen as a specific version of Grid services. Although SAN (Storage Area Network) services are comparable to Grid computing services, they do not rely on the same type of background technologies. Storage services are out of the scope of this thesis.

- **Resource orchestration services on demand:**

  This service aggregates network, storage and computational resources to provide a resource orchestration service. The idea is that the user would simply describe their resource requirement over time and the operator would have to find, select and reserve them over time, to satisfy the user's request.

This thesis deals with the problem of integrated computational service provisioning. To achieve this, it is necessary to understand which type of information is exchanged between the computing department, the network department and the resource owners.

Exchange information model

Computing service provisioning assumes an exchange of information between the various entities of the unifying framework described above. In this context, the fact that each actor (the computing Dpt, the network Dpt and the resource owners) does not want to provide detailed information about its own infrastructure or available resources is a serious constraint. More generally, these actors do not want to disclose confidential information to their competitors, such as the quantity of a resource available, the average load, or worse even the future estimated load. Such a problem is not technical but refers to commercial business and organizational constraints.

Thesis structure

This thesis is organized as follows. Chapter 1 presents a state of the art of the essential Grid and networking concepts and technologies used to provide integrated computing services. Chapter 2 proposes some architectural solutions for integrated computing service provisioning. Chapter 3 analyses the control mechanisms to reserve computational and network resources. The performance of these control mechanisms are evaluated by means of discrete time event simulation applied to a queueing model of the problem. Chapter 4 investigates and presents one aspect of efficient resources utilization. Finally, we conclude this thesis in providing a few perspectives in the continuation of our work.

Chapter 1

State-of-the Art

1.1 Introduction

This chapter will present a brief of the current ideas on business models, technologies and architectures to provide computing services through a wide area network. Although some examples of storage services will be shown and discussed, this chapter does not intend to cover storage services.

The first section presents the business models, the second describes existing Grid toolkits and their standardization status and the third describes how the network resources are managed and taken into account in many recent Grid projects.

1.2 Business Models

Outsourcing today drives the service industry. More companies are focusing on their core business and competencies, leaving other specialized firms to take over non-critical business functions. Companies like Accenture or IBM manage for their customer's business functions such as payroll editing, procurements, etc... Those actors are combining resources, employees and IT, to provide a common service to their customers. The benefits for their customers are better-controlled costs, lower operating expenses (OPEX), more qualified staff and higher utilization of resources... This outsourcing trend also applies for computing services.

1.2.1 Different services

As described earlier, a computing service can be provided at three different levels:

- Raw computational resource access, the end users have to provide an image of an Operating System and the running programs
- Computational service, the end users have to provide an image of the source or binary to execute on a given platform
- Application service, the end users have to provide input files or they simply interact directly with the application

On-line Application Rental

In this model, the service provider is an application provider. It leases access to an application. Its customers access the application through the network. The application interface can be via a web browser or a dedicated application (e.g. a Java application). The application cannot work offline. Some examples of this kind of online application are Google Docs, Calendar and Mail programs.

Raw computational resource Rental

In this model, the service provider provides access to a computational resource. The user sends to the service provider an image file of a fully operational operating system and the required application. The provider only executes the image. Some examples of this are the Amazon Elastic Compute Cloud service [1, 2]. The Amazon computational service works with the Amazon storage service. Through a Web Service interface [1], the user can submit an image file and then run it.

Job execution service

In this model, the service is a job execution service. The user sends their jobs and its input files to the service provider. The files must contain either a binary executable program or a source code and an executable file to start it with. The files must be compatible with the architecture of the service provider. Most of the time, the jobs are executed on the same cluster. Examples of such services are Sun Grid Compute Utility [23] and Cluster Computing on Demand provided by TTI [4]. In the Sun example, jobs are sent

1. SOAP and Query as of today

via archive files (.zip, .tar.gz), they must contain a binary executable and be compatible with the Solaris 10 environment.

Storage services

Some Internet service providers are providing a storage service to their customers. A simple example is Google's "on line storage service". Users can store their files via various means: from a simple HTTP POST method, (i.e. a web page file upload), to an FTP access or SFTP, or an application integrated to their operating system. The file retention policies can vary: it can be based on inactivity or user class.

1.2.2 Pricing, Payment

Service presented above are priced in proportion of the use of the service: 0.8$/CPU/hour or 0.15$/MB/Month, but other pricing framework can be used. Buyya [51] analyzed several payment models: prepaid, pay as you go, postpaid. The pricing can be a flat rate for unlimited use, or a variable rate for a price proportional to usage like before, subscription rate or demand and supply based rate. All computing services presented above are required to have a binary program compatible with each independent resource provider. If a computing department of an operator wanted to provide a computing service hiding the heterogeneity of the Operating system, local scheduler vendor, processor architecture to its customers, it would need to use a Grid toolkit. Several Grid toolkits will be presented in the next section.

These services, as they exist today suffer from the same problem. They do not guarantee network QoS. Applications and services are overlaid on the network with no resource management information exchanged. Furthermore, the network department of the operator does not interact with the service delivery, the traffic is not even QoS classified. As a consequence, all these application suffer from poor, indeterministic and unreliable network performance a majority of the time. Before going into more detail on how network resources are taken into account by some recent Grid projects (as in section 1.5), the next section describes the existing Grid Toolkits and their standardization status.

1.3 Web services

Web services typically refer to two technologies:

- REST, Representational State Transfer [67]

- SOAP, Simple Object Access Protocol [49]

Briefly, REST is an architecture that associates a resource to everything, via a unique URL. Each resource can be retrieved, created, updated and deleted with the HTTP messages GET, PUT, POST and DELETE.

SOAP is a Remote Procedure Call protocol using XML as a standard way to transport formatted data. Today's Grid toolkits rely heavily on SOAP over HTTP, although standards try to be independent of the underlying technology. Since HTTP is a stateless protocol, and given that resources must manage states, a resource framework has been added to the traditional web services. WSRF permits the management of stateful resources, whose state now becomes available through classic SOAP calls. In the rest of this thesis, WS will only refer to SOAP like web services.

Example of a SOAP request:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>827635</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

And here is a possible response to the client request:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productName>Toptimate 3-Piece Set</productName>
        <productID>827635</productID>
        <description>3-Piece luggage set. Black Polyester.</description>
        <price currency="NIS">96.50</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

WS Description Language (WSDL) is a way to standardize the description of the different interfaces a web service can provide. It defines a language to describe a web service, its interface, operations, input and output and error messages.

1.4 Distributed Resource Management Systems

This section presents typical features of a Grid toolkit. A Grid toolkit is also called a distributed resource management systems (DRM). Then, major toolkits are presented, followed by a description of their current standardization status.

1.4.1 Generic features

Grid toolkits were originally developed to interconnect clusters belonging to different organizations. Each cluster has its own cluster management software also called a local resource manager. These local resource management software tools often include a scheduler, sometimes they are only a scheduler. Examples of these are:

- Platform Load Sharing Facility (LSF) [18]
- Altair Portable Batch System (PBS) [19]
- Sun Grid engine (N1GE) [24]
- IBM LoadLeveler (LL) [26, 89]
- DataSynapse GridServer [6]

Of course, interconnecting clusters require having a global security infrastructure in place. So, the most important features of a Grid toolkit are:

- Interoperability
- Security
- Job execution
- Data exchange

As a consequence, all Grid toolkits need to perform the same functions, authentication, authorization, job execution and file transfer. They usually share the same components:

- An interface to adapt (An Adapter) to each local resource manager's "language" (LSF, PBS, N1GE, LL, ...)
- A security component, for authentication and authorization

- A job execution engine
- A user client
- A data management system, either via a "global" file-system or a file transfer program

Basic Grid toolkits have all of these components. In many research projects a global file-system provides the file management and interconnects between the sites. This solution separates execution tasks from data transfers. This greatly simplifies the execution tasks. Grid toolkits provide a simple job execution interface: all the user needs to do is to submit an execution command specifying the name of the program to be executed and the machine or cluster where it is supposed to run. If a global file-system is available, the job submit command triggers the execution of the job on the remote machine, after having performed security checks and translated the command in the remote machine's "language". If no global file-system is available, the job submit command also triggers the job transfer from the source machine to the remote.

A complete description of Global file-systems is out of the scope of this thesis. The word global [10] file-system is misused here, its true meaning refers to a file-system that makes files available through a global network (as opposed to a local area network) and avoids host-dependent mount points (hence filenames). Strictly speaking, the following list refers to some distributed file-systems used over global networks:

- Global File System [145, 9]
- Google file-system (Not freely available)
- zFS [128]
- and many more can be found on Wikipedia's file-system list [15]

For a network operator to provide an integrated computing service, it can not assume the existence of a global file-system. Although for simplicity, this chapter might assume the presence of a global file-system as it is the case in many Grid projects when a dedicated network interconnects sites.

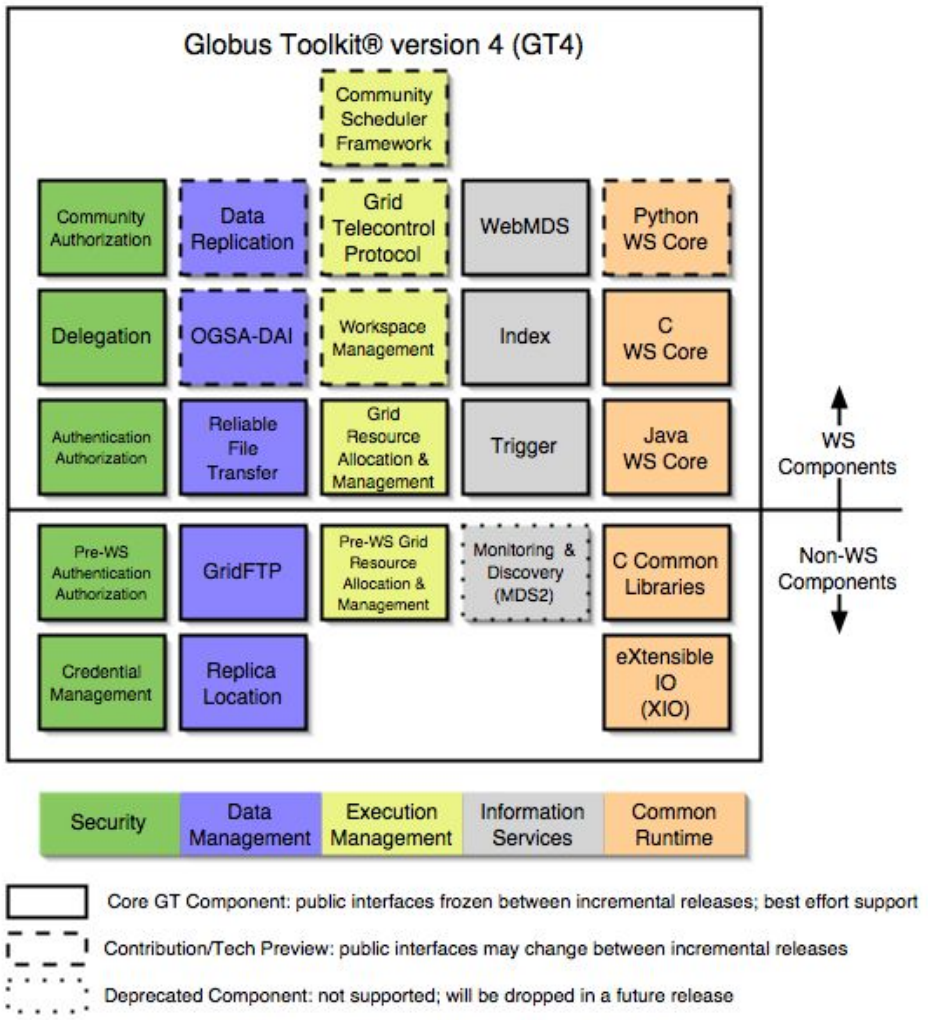
Most Grid toolkits support plug-ins. Schedulers often use plug-ins to connect to other software tools. A Grid scheduler [149, 152], or meta-scheduler, schedules jobs at the Grid level by interacting with local schedulers.

1.4.2 Globus

Globus was amongst the first Grid toolkits. Created by Ian Foster [69], it is still widely used. Version 4 is the last version released in 2007. It is fully WS based and was built

around the WS Resource Framework (WSRF [43]).

Figure 1.1: Globus Toolkit



Globus functions are grouped within five domains:

- Security
- Data Management
- Execution Management
- Information Services
- Common Runtime

Information services

Information services in Globus 4 are provided by Monitoring and Discovery System 4, MDS4. "Monitoring is the process of observing resources or services (e.g., computers and schedulers), for such purposes as fixing problems and tracking usage. A user might use a monitoring system to identify resources that are running low on disk space, in order to take corrective action." Discovery is the process of detecting resources and their configuration.

MDS4 is built with three modules. Together, they provide a directory service.

- Index service centralizes resource discovery and status messages.
- Trigger service permits to trigger messages when an event occur. For instance, it's possible to send an email when a storage area is full.
- Aggregator service permits the creation of groups of entities to multicast information. (I.e. Disseminates information among a group of entities)

Execution management

Grid Resource Allocation Manager (WS-GRAM) is composed of a set of services that enables localization, submission, monitoring of jobs on computing resources. It is not a scheduler but a set of services and clients that communicates with local schedulers. WS-GRAM manages submitted jobs, follows resource status, manages credentials and triggers data stage in and out if necessary.

Community Scheduler Framework is a Grid scheduler framework. It is composed of a job service that creates, submit, monitors jobs, a resource reservation service, a queue management service. The project GridWay [108, 13] is a Grid scheduler for Globus. Unicore's meta-scheduler will be described in more detail. Both Grid schedulers have similar functionality.

Data management

GridFTP is a data transfer program that is more efficient than the traditional FTP to transfer huge files.

Reliable File Transfer RFT is a web service that supervises data transfer, resuming interrupted transfers in case of failures. It can also execute post transfer actions and broadcast failure notification messages.

Replica Location Service is a service to manage replicas. It dissociates logical resources from their physical location.

Security management

Welsh describes Globus Security Infrastructure in [156]. Credential management services manage credentials. Furthermore, WS-Authentication and WS-Authorization provide sender authentication, message encryption, integrity verification and delivery verification. These two standards support different security architectures, Access Control Lists, special authorizations and SAML protocol [55]. Delegation web service permits authorization delegation as well as credential renewal. Community authorization service centralizes credentials for a given VO.

1.4.3 Unicore

Unicore [65, 66] is an acronym that stands for uniform interface to computing resources. Its development conception goes back to 1997, when German supercomputer centers wanted to provide a seamless, secure and intuitive access to the heterogeneous computing resources at their centers. The project was sponsored by the German Ministry for Education and Research (BMBF) and had the following objectives:

- Hide the seams resulting from different hardware architectures, vendor specific operating systems, incompatible batch systems, different application environments, historically grown computer center practices, naming conventions, file-system structures and security policies, just to name the most obvious.
- Security was to be built into the design of Unicore from the start, relying on the emerging X.509 standard for certificates authenticating servers, software and users and encrypting the communication over the Internet.
- Finally, Unicore was to be usable by scientists and engineers without having to study vendor or site specific documentation. A graphical user interface was to be developed to assist the user in creating and managing jobs.

In addition several other conditions had to be met: Unicore had to support operating systems and batch systems of all known vendors represented at the partner sites. It had to be non-intrusive such that it does not require changes to computing center practices especially for users and system administration. Besides, Unicore's own security model site specific security requirements had to be supported.

In 2007, Unicore released version 6. This version relies on WSRF and provides WS interfaces. So far it is not fully interoperable with Globus 4, but interoperability tests

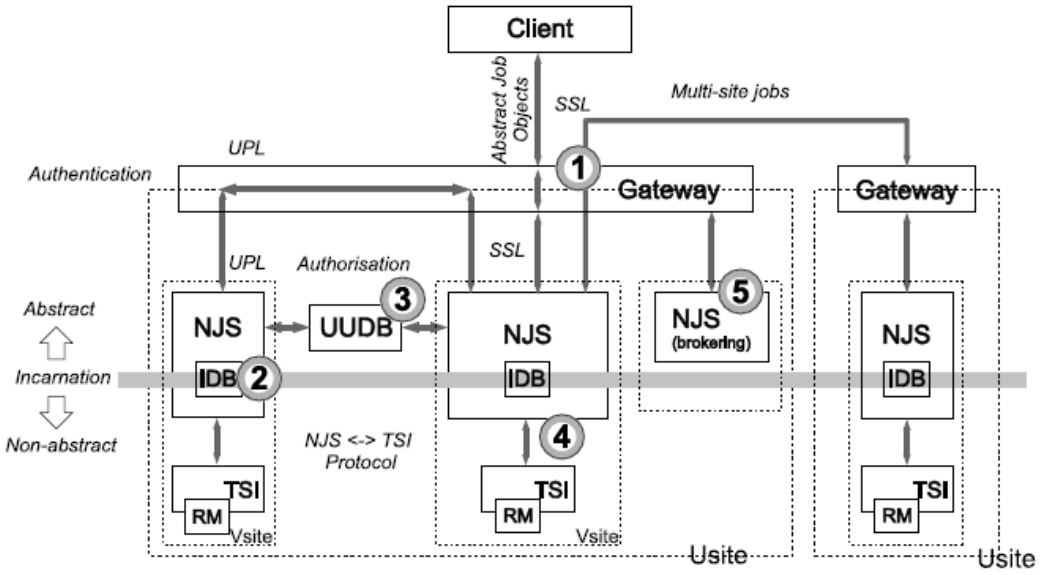
are to be performed, either by the Open Grid Forum [17] (see below) or the European Telecommunication Standard Institute (ETSI).

Unicore Architecture

The article [107] briefly describes the main components of Unicore.

As shown in figure 1.2 Unicore introduces a three-tier Grid architecture consisting of user, server and target system, an implementation of which is realized entirely in Java 2. [65] provides a more extensive insight.

Figure 1.2: Unicore architecture



The user tier is represented by the Unicore Client, which provides a graphical user interface to interact with the functionality offered by the server. Sending and recieving Abstract Job Objects (AJO) via the Unicore Protocol layer to achieve this. The AJO is the implemetation of the job model and central to Unicore's philosophy of abstraction and seamless integration. It contains platform and site neutral descriptions of computational and data related tasks, resource information and workflow specifications along with user and security information. AJOs are built within the user tier and sent formatted as serialized and signed Java objects to the Gateway.

The Gateway is the secure entry point to a Unicore site, a Usite, which authenticates user requests and transfers them to the Network Job Supervisor (NJS). The NJS translates the abstract job into a target system specific one, a process called Incarnation, making use of the Incarnation Database (IDB). Furthermore the NJS contains a workflow engine which, amongst other tasks,

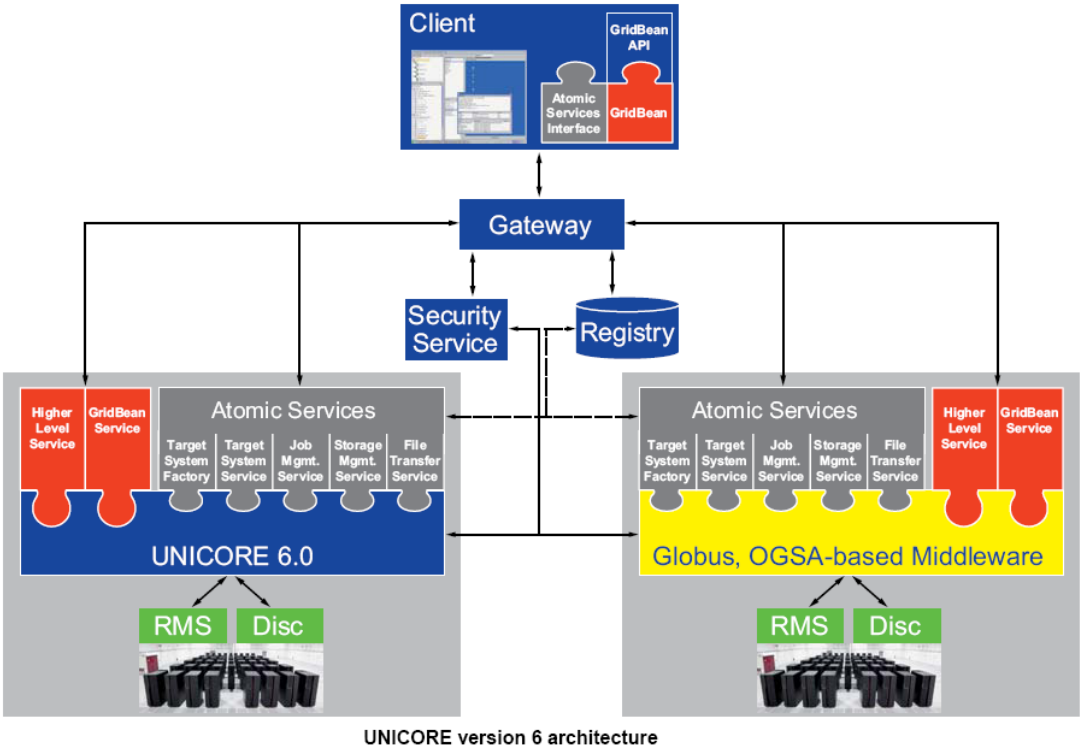
performs pre and post-staging of computational data and authorizes the user by contacting the Unicore User Database (UUDB). The Gateway and NJS execute typically on dedicated secure systems behind a firewall.

Unicore's communication endpoint is the Target System Interface (TSI), a stateless daemon executing on the target system. It interfaces with the local resource manager represented either by a batch system like LoadLeveler, a batch system emulation on top of Linux (for example) or an entity capable of providing access to Grid resources like Globus.

Permanent X.509 certificates are used to establish SSL connections between Unicore components 4 (as shown in Diagram 1.2) and to sign Abstract Job Objects (see [78] for a full analysis of Unicore's security model).

Unicore version 6 is supposed to interoperate with Globus, the interaction will be done following the architecture shown in figure 1.3. A Grid scheduler has been developed for

Figure 1.3: Unicore 6 & Globus



Unicore, see [152].

1.4.4 Sun N1 Grid Engine (SGE)

Sun has developed an independent Grid toolkit. Two versions are available, a freely distributable version and a commercial version. SGE provides a policy based workload

manager. It includes a policy based scheduler and is composed of the following modules:

- A master daemon is responsible for the management of jobs on execution hosts.
- A Scheduler which is part of the master daemon, is responsible for the selection of execution queues for each job.
- An execution daemon is responsible for the communication between an execution host and the master. It receives job execution commands and provides feedback on the load to the master in order to optimize the scheduling process.

SGE was installed, maintained and used during this thesis as the main tool to share the workload of simulations of chapter 4.

1.4.5 Standard status

Grid technologies rely heavily on existing standards. The Grid community gathers two or three times a year at the Open Grid Forum (OGF) to discuss them. The OGF resulted from the merger of the Global Grid Forum and Enterprise Grid Alliance.

"The Open Grid Forum (OGF) is a community of users, developers and vendors leading the global standardization effort for Grid computing. The OGF community consists of thousands of individuals in industry and research, representing over 400 organizations in more than 50 countries. The work of OGF is carried out through community initiated working groups, which develop the standards and specifications."

OGF tasks are divided in areas, where each area is composed of several working groups or research groups. The main areas are Applications, Architecture, Compute, Data, Infrastructure, Liaison Management and Security.

OGSA

OGF's Open Grid Services Architecture [74] "does not define any standards or technical recommendations". It's a service oriented architecture that can be used to describe the main components of a Grid. The functions are handled by components, accessed using a client/server communication model: functions become services. High level functions or services are realized using lower level components. This way, a hierarchy of functions and associated services is defined. OGSA describes six families of functions:

- Execution and Management

- Data
- Resource management
- Security
- Self management
- Information

Job Execution and Management functions are related to instantiation, execution and management of units of works. One of its most important functions is resource selection. Data regroups functions used to move data to where it is needed, manage replicated copies, run queries and updates and transform data into new formats. Resource management functions are related to the management and control of resources, such as reservation, monitoring, supervision, as well as management of other top-level components. Security contains security and policy related functions, such as authentication, authorization, audit and identity mapping. Self-management automates as many tasks as possible, such as healing, configuration and optimization. Information refers to dynamic data or events used for status monitoring, relatively static data used for discovery and any data that is logged. Information services aim to efficiently access and manipulate information about applications, resources and services.

The execution and management service described in OGSA and the computing area will be described in more detail.

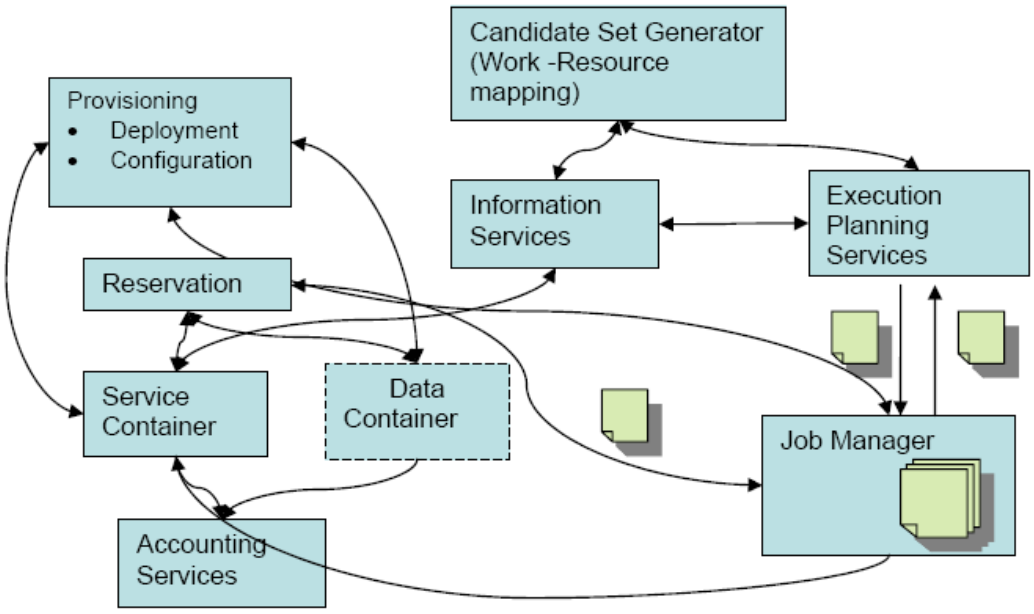
Execution and management services

According to OGSA, prior to the receipt of a job request by the Job manager, a service container is created to represent the computational resources. This container communicates with the information services (IS). The IS provides access to a directory of services and resources.

If a user wants to run a job interacting with a portal or queue job manager, there are four basic phases to starting the job:

- Job definition phase. What are the input files? What are the service level requirements? E.g., job must complete by noon tomorrow. What account will be billed for the job? Etc.
- Resource Discovery: Discover the resources available and select the resources required to execute the job.
- Scheduling: Enact the schedule, e.g., provisioning of resources, accounting, etc.

Figure 1.4: Execution and Management Service



- Monitoring: Monitor the job through its lifetime(s). Depending on the service level agreements the job may need to be restarted if it fails to complete for any reason.

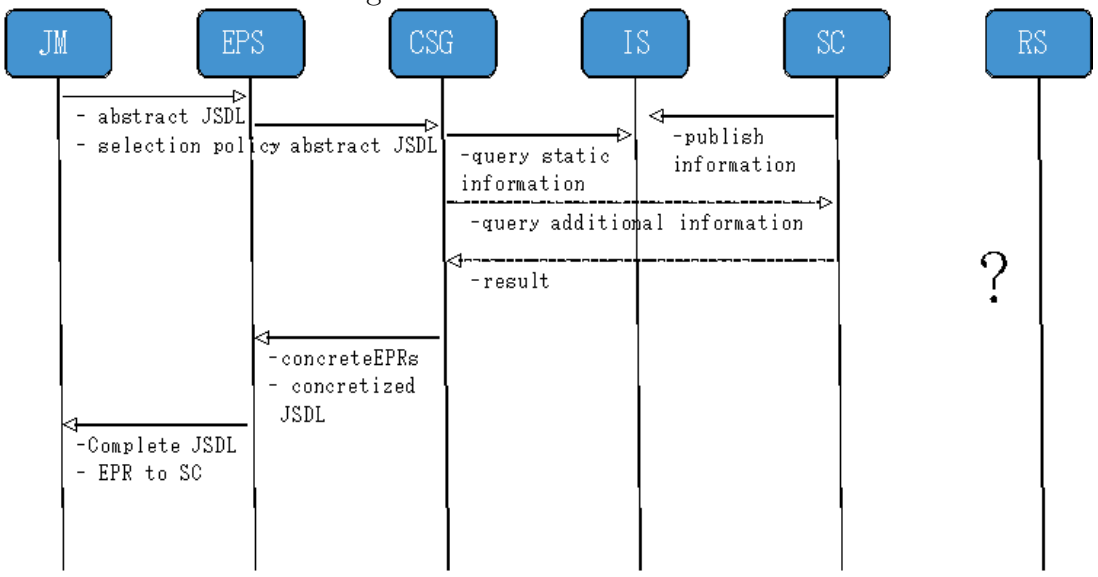
To realize this using Execution and Management Services, the Job Manager (JM) creates a new job with the appropriate job description (e.g., written in JSDL [42]). The JM then calls an Execution and Planning Service (EPS) to get a schedule. The EPS in turn calls a Candidate Set Generator (CSG), which queries information services (IS) to determine where the job can be executed based on binary availability and policy settings. The EPS selects a service container, after first checking with the service container (SC) that the information is accurate. The EPS then returns the schedule to the JM. The JM interacts (if necessary) with reservation, deployment and configuration services to set up the job execution environment. This may involve interaction with the data container as well. The service container is then invoked to start the job. Logging services are used for accounting and the audit trail. When the job terminates the job manager is notified by the container. If the job terminates abnormally, the whole cycle may repeat again (see Figure 1.4).

The list of the different meaning of the acronyms used in figure 1.5 are:

- RS=Reservation Service
- SC=Service Container

- IS=Information Service
- JM=Job Manager
- EPS=Execution and Planning
- CSG=Candidate Set Generator
- EPR=End Point Reference
- JSDL=Job Submission Desc Lang
- BES WG= Basic Execution Service Working Group
- WS=Web Service
- WSRF=WS Resource Framework
- UDDI=Universal Description Discovery Integration
- WS GRAM=Grid Res Alloc Mngr (not standard but WSRF based)

Figure 1.5: Execution Scenario

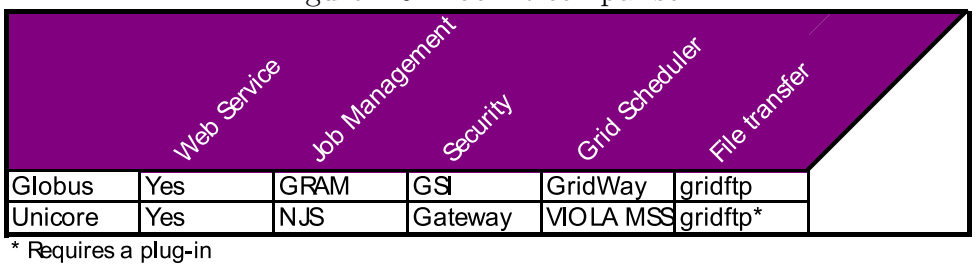


The OGF and Resource Selection Services working groups are defining protocols, schemas for the candidate set generator and the execution and planning service. More generally, OGF has an open policy towards OGF draft documents. OGSA does not set any constraints on Grid toolkits to be "OGSA compatible". The consequence is that most of Grid projects have independent protocols that do not necessarily interoperate.

Furthermore, OGSA does not make any recommendations, it's more like a vision of the different functions required to manage a Grid. Each of the different OGF working groups proposes some protocols and elements that interact to provide a high level service. Defining compatible toolkits relies on the description of different services through profiles. An approach to defining interoperability standards is emerging.

1.4.6 Conclusion

Figure 1.6: Toolkit comparison



The two major toolkits are very similar. Although Globus is older, both toolkits have a huge set of extensions and user base. They could both claim to be "OGSA compatible", but OGSA is too general to guarantee real interoperability. So far, several initiatives exist to provide real interoperability [123, 122]. ETSI has launched an initiatives to facilitate interoperability tests.

"Diagram" 1.6 makes the mapping between Globus and Unicore components.

Current standards and toolkits do not support resource co-allocation. The next section will describe a state of the art of network and computational resource co-allocation scheme.

1.5 Grid Networks

Grid Networks is a research domain going from "passive" networks to the integration of network resource by Grid toolkits. "Passive" means that Grid applications are overlaid on the network. I.e. there is no interaction between the network QoS mechanisms and its applications, whether QoS mechanisms are per hop behavior like DiffServ or connection oriented like (G)MPLS. In 2004, the overlay model was the predominant model in Grid projects, where huge dedicated networks were built to run bandwidth greedy applications. Other research projects not dedicated to Grid applications were focusing on network control and management plane research. Before 2004 the Generalized Multi Protocol Lavel Switching GMPLS [8] standard, as a connection oriented control plane for multiple switching technology (packets, circuits, lambda, ports) emerged. In 2007, many research Grid projects are integrating network resources, capable of using a guaranteed QoS service (Bandwidth reservation) or premium IP service (DiffServ service). A book entitle "Grid Networks" was published [148].

This section will briefly summarize a survey performed in 2004 by the EGEE [29, 33] project adding more recent references.

The purpose of the survey was to provide an overview of the current technologies

and to understand what can be learnt from these previous efforts in respect of interfaces, architecture, design decisions, resource management, authentication, authorization and multi domain aspects.

The following network resource management systems were considered in the survey:

- Network Resource Scheduling (NRS) [12]
- Globus Architecture for Reservation and Allocation (GARA) [7, 71]
- GARA based DataTAG [57]
- User Controlled Light Paths (UCLP) [27]
- Internet2 QBone Bandwidth Broker [20]
- EGEE Bandwidth Allocation and Reservation (BAR) [32]
- Dynamic Resource Allocation in GMPLS optical networks (DRAGON) [99]
- AkoGrimo [37]
- EuQoS [133]
- Phosphorus [104]
- Nortel's DRAC [16]
- VIOLA [28, 152]
- Enlightened [45]
- G-Lambda [146]

The next section is a brief network refresher of basic networking concepts, before going to the survey.

1.5.1 Network concepts

This section is not a networking course. It's just a quick refresh of concepts commonly used in the rest of the thesis.

Autonomous System

Networks like the Internet are hierarchical. Two levels must be distinguished: within an autonomous system and between autonomous systems. An autonomous system is composed of network nodes like routers and switches managed by a single administrative entity. Internet's inter AS routes are advertised with Border Gateway Protocol [125] (BGP).

Traffic Engineering

Traffic engineering is the capability to allocate and manage network resources for a portion of the traffic. It permits traffic prioritization and optimal resource allocation. Link's state traffic engineering extensions are stored in the traffic engineering database (TEDB). This database is distributed by a TE extended routing protocol like OSPF-TE [90] (Open Shortest Path First Traffic Engineering) or ISIS-TE [143] (Intermediate System to Intermediate System Traffic Engineering). Information such as available bandwidth on each link for reservation or reservable bandwidth is contained in this database.

Distributed control plane

The network control plane is the set of functions relating to the control of the network. The two main functions are signaling and routing. The signaling functions in GMPLS relies on Resource Reservation Protocol with Traffic Engeneering RSVP-TE [46], while the routing function on OSPF-TE or ISIS-TE.

Inter-Domain LSP scheduling

For the immediate Label Switched Path, LSP set-up request, RSVP-TE is enough. However no inter domain TE routing information is propagated with GMPLS. Furthermore the GMPLS control plane only works with immediate reservations and only propagates immediate link state. Future link state information and future resource reservation can not be done with GMPLS.

When an LSP set-up has to be done in the future, but the reservation of resource should occur now, domains need to cooperate. They need to exchange signaling and routing information. Two models permit an inter domain LSP set-up:

- Coordinated, see Diagram 1.7
- Daisy Chain, see Diagram 1.8

Figure 1.7: Coordinator Model

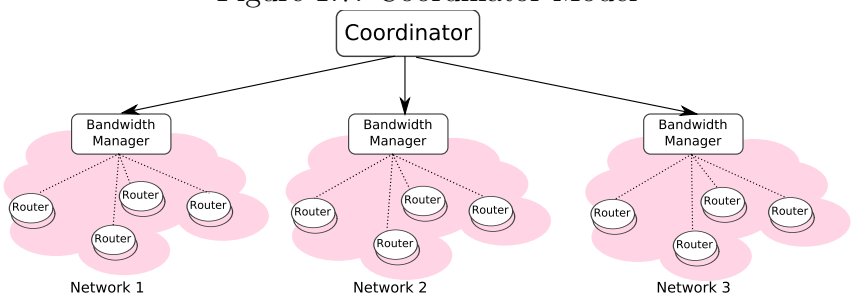
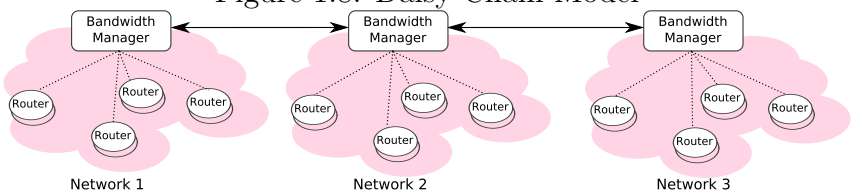


Figure 1.8: Daisy Chain Model

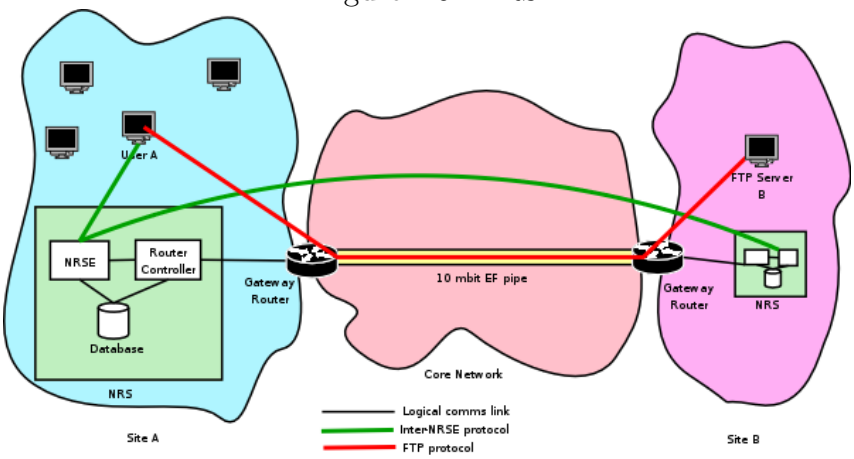


In the coordinated framework an element part of a domain, the source domain for instance communicates with all other domains to signal the LSP. While in the daisy chain framework, each domain is responsible for its own and triggers the signaling in the next domain.

In a very similar way, routing information could be theoretically centralized, but actually it is distributed across domains.

1.5.2 Grid Resource Scheduling (NRS)

Figure 1.9: NRS



The Grid Resource Scheduling project [12] has implemented a system to schedule access to network resources at the edges of a single domain which is assumed to be "over

provisioned" internally. NRS is designed to control the access to the DiffServ Expedited Forwarding service (EF). The principal NRS component is the Network Resource Scheduling Entity (NRSE), which lies at the source and destination sites. An NRSE allocates a portion of the DiffServ allocation to a user's flow on demand. The main limitation of NRS is that it is designed for a single network domain and has no mechanism for control of flows that cross multiple network domains.

The NRSE provides a signaling protocol at the application stratum (see below NGN model). This is an open, human readable protocol based on XML that enables users to request that the NRSE add, modify and delete QoS reservations.

The NRSE also provides admission control. Each NRSE maintains a database of reservations and only accepts a new reservation if it has sufficient bandwidth unallocated. For non real-time requests, such as file transfers, it may suggest an alternative booking that still results in the file being transferred before the user's specified deadline. The NRSE also has an inter-NRSE protocol. This is necessary because each site has its own NRSE and admission control must be performed at both the source and destination networks for bi-directional flows. (Having the client communicate with both local and remote NRSE would not scale well, because every NRSE would need to be able to authenticate every client.)

The project focuses on a DiffServ class of service, Expedited Forwarding and tries to manage bandwidth with a per hop behavior without connections. The architecture chapter will assume a network capable of reserving resources and establishing virtual connections.

1.5.3 Globus Architecture for Reservation and Allocation (GARA)

The Globus Project [69, 11] has been actively working on integrating QoS into Globus. This is called Globus Architecture for Reservation and Allocation (GARA) [7, 71]. Although the GARA work has been within the context of Globus, it only relies on Globus slightly.

GARA provides QoS for different types of resources: networks, CPUs, batch job schedulers, disks and graphic pipelines. It also provides mechanisms to allow both advance reservations and immediate ("right now") reservations for quality of service. It was one of the first projects for resource co-allocation (network and CPU) in a Grid context.

A simple resource specification language (RSL) is used to describe a reservation. When a reservation is created, a handle is provided to the application regardless of the type of reservation it is (Network, CPU etc). Reservations can be created, modified and cancelled. Reservations can be monitored, either through a polling mechanism or through callback

functions.

Although initially designed for DiffServ, GARA has been extended to interact with Multi Protocol Label Switching (MPLS) networks.

1.5.4 GARA extensions

The GARA based DataTAG Advance Reservation Architecture [57] is mainly the result of the joint effort of the DataTAG project in order to extend GARA to support connection oriented networks like MPLS. MPLS is a mechanism to switch packets through a network based on a label. The support of DiffServ by MPLS is specified in RFC 3270. Indeed, MPLS supports QoS but does not introduce any new QoS paradigms.

A "Path" defines the general network entity providing unidirectional connectivity between two network nodes. It can be end to end or per domain. It can be inter domain. Each "Path" is managed by a specific Network Resource Manager, which depends on the path type, and is associated to the authentication/authorization entity (a Gatekeeper instance or Generic AAA server) for the user access control.

1.5.5 User Controlled Light Paths

The User Controlled Light Paths (UCLP) [27] software is a tool to partition an optical SDH/SONET network providing end users with the ability to control, provision and configure an optical VPN. Users in turn can hand off control and management of optical network resources to other users. This approach provides end users with full control to dynamically reconfigure the optical VPN without any intervention by the network operator. The UCLP software is now deployed across the CA*net 4 network [3], which is the Canadian research network. Four software teams have been funded to develop different versions of the software. Not all version share exactly the same features like multi-domain for instance.

SDH point to point capacity is represented in the system as Light Path Objects (LPOs). LPOs are under the control of a certain user. The user can use the LPOs that are under their control to establish an end to end lightpath by concatenating LPOs. The user can also use LPOs that are advertised by other users and lease them to gain control on them. In addition, users can partition an LPO. This implies that for instance a 10G SDH circuit can be represented as a single LPO with a capacity of 10G or as four LPOs with a capacity of 2.5G each.

The inter domain component within the first domain is responsible for coordinating the lightpath setup over the different involved domains. Note that this approach only requires

a full network topology overview within each network. No AS topology is maintained, only the service locations (i.e. where the appropriate web services can be contacted) should be known for each AS.

User management is an important concept in the UCLP system. Basically, there are users and administrators. The latter can add/remove users as well as adding/removing root LPOs, which represent SDH capacity between adjacent nodes.

UCLP is used in the production network of the Canadian NREN. UCLPv2 is being developed, it relies on Web Services, ontologies and WS workflow execution, BPEL, (Business Process Execution Language).

1.5.6 Internet 2 Qbone Bandwidth Broker

The QBone initiative from Internet2 [20] has been active between 1998 and 2001. Their aim was to design and implement an infrastructure to control DiffServ in the Abilene backbone. The work consisted of two parts:

- Design of a Bandwidth Broker whose main purpose is to manage the Premium IP resources in the network. It performs admission control to take care that the amount of Premium IP traffic does not exceed a certain percentage of the link capacity. Each domain has its own bandwidth broker.
- Signaling protocol SIBBS, to facilitate the inter domain communication between the different bandwidth brokers (i.e. between different domains).

The bandwidth broker design included a clear split between inter domain and intra domain functionality. This bandwidth broker was designed to setup Premium IP reservations that possibly could span multiple administrative domains. The inter domain signaling protocol SIBBS was designed for the communication between bandwidth brokers of different domains. SIBBS is an application protocol that runs on top of TCP.

No implementations were produced.

1.5.7 EGEE BAR

The European Research Area is currently served by a set of NRENs linked via a high-speed pan-European backbone, the GEANT network, built and operated by DANTE. The EGEE Grid will use these networks to connect the providers of computing, storage, instrumentation and applications resources with user virtual organizations. Grid demonstration projects have shown that Grid applications can generate very high volumes of

network traffic that can exceed the current aggregate flows from non-Grid usage, and will therefore demand new and innovative features of GEANT and the NRENs over and above the current best efforts IP service.

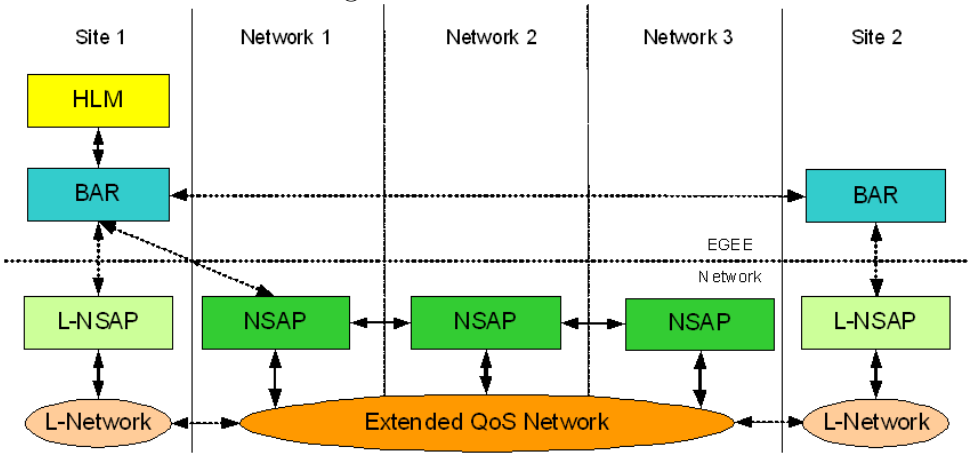
EGEE is putting in place a web service to implement bandwidth allocation and reservation [32] (BAR). This will allow the usage of the network to be controlled and balanced and to categorize and prioritize traffic flows so that users and the layers of Grid middleware receive the required level of service from the network.

BAR is the culminating product of EGEE after having analyzed all of the projects described above. None of them provided all of the necessary requirements.

Architecture

Figure 1.10 shows how a BAR service interacts with a High Level Middleware (HLM) (a.k.a. Grid toolkits) and network services (NSAP). There are three web services involved

Figure 1.10: EGEE BAR



in Bandwidth Allocation and Reservation: BAR, Network Service Access Point (NSAP) and Local Network Service Access Point (L-NSAP).

- BAR receives the HLM request in a network-neutral language. It passes the request on to its designated NSAP and L-NSAP for the configuration of the backbone and local network respectively, and also to the BAR at the target destination. When interacting with a NSAP and L-NSAP, BAR translates an HLM request into a network oriented request.
- NSAP is present in the backbone (GEANT and NRENs). They are concerned with the configuration of network equipment on the backbone. Their functionality involves sending notification to the BAR of the success of the request. As explained

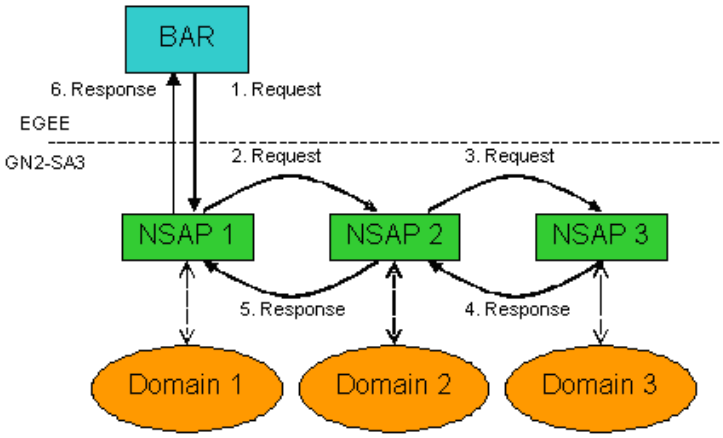
in [33], the NSAP abstracts the network specific services but still speaks a language that is network oriented.

- L-NSAP is of equivalent functionality to the NSAPs but concerned with the configuration of equipment on the local network of the source and destination sites (last mile problem).

The HLM and BAR entities belong to the EGEE administrative domain and as such they are developed by EGEE. Network service providers are free to implement NSAP (and L-NSAP) however they see fit.

GEANT2 proposed an implementation of NSAP called AMPS, Advance Multi-Domain Provisioning Service [31]. L-NSAPs belong to the administrative domain of the end-sites and the resulting diversity of local equipment and policies complicates the last mile problem. Each AMPS communicates with their counterparts in other domains, as depicted by figure 1.11.

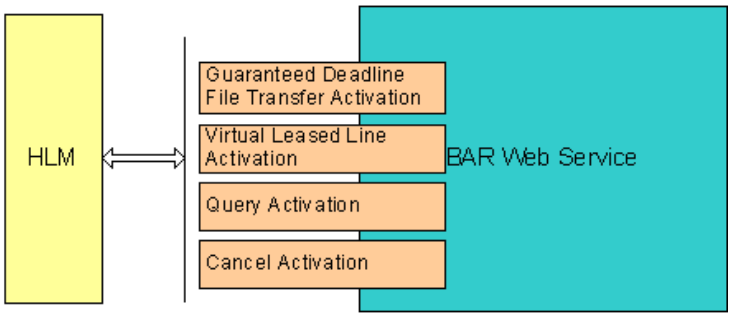
Figure 1.11: NSAP inter domain communication



BAR

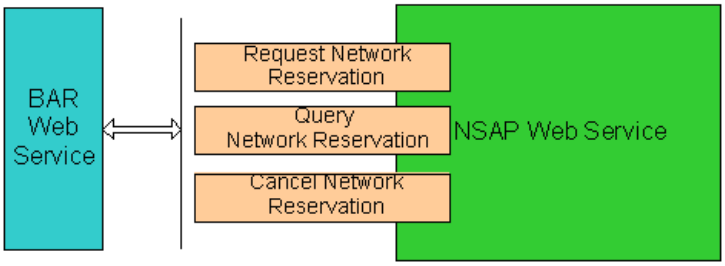
The HLM-BAR and BAR-NSAP interfaces are represented as web service interfaces, each being composed of set of Port Types (PTs) showing the operations that can be performed. BAR Web Service (figure 1.12) shows the BAR service interface that is exposed to the HLM. The first version of the interface proposes different services: Bulk Transfer, Virtual Leased Line, Video and Visualization. The framework is extensible as extra Port Types can be added to support additional Service Types. It is also flexible since each service type is free to define, if necessary, a completely different interface. That is, operations and their signatures can be completely different from one service type to another. EGEE BAR services require two phases: reservation and activation.

Figure 1.12: BAR Web Service



NSAP

Figure 1.13: NSAP Web Service



The specification of the NSAP is outside the scope of EGEE, an overview of its functionality is provided here. The NSAP specification, implementation and deployment is currently ongoing work within the SA3 activity of the GEANT2 project.

BAR-NSAP interface and NSAP Web Service (Diagram 1.13) shows the interface exposed by the NSAP Web Service. The interface is used by a BAR service to access network-oriented services in order to fulfill HLM requests. The parts of the NSAP interface and the parameters they accept are described in [33]. The port types RequestReservationPT, ModifyReservationPT, CancelReservationPT, ReservationStatusPT and ReservationNotificationPT correspond to the Request Network Service, Modify Network Service, Cancel Network Service, Query Network Service Status and Network Service Notification.

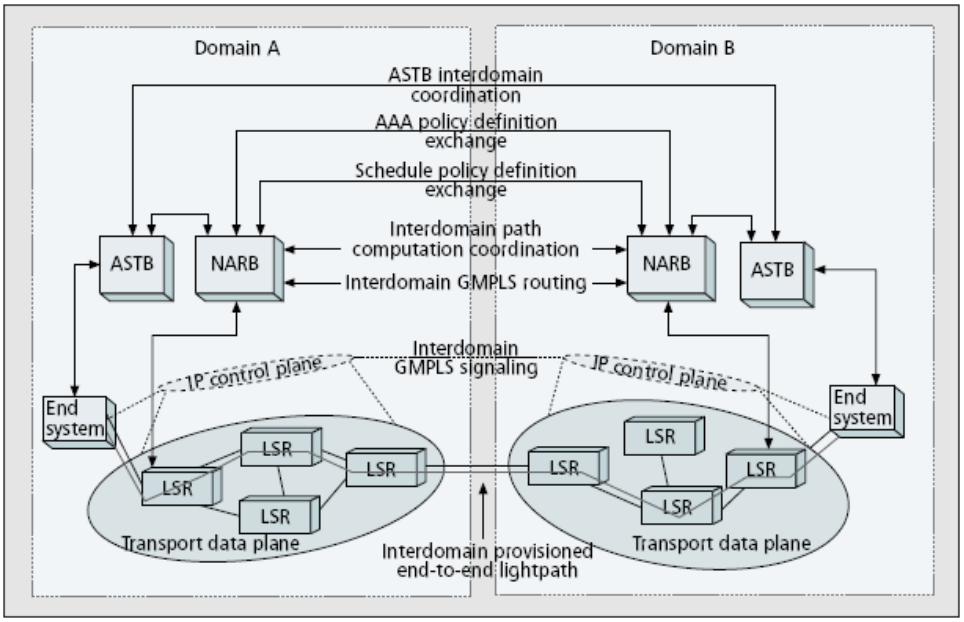
Note that translation is necessary between the operations in the BAR service and the ones in the NSAP service.

The NSAP-NSAP interface is not yet documented. But it must be composed of two functions: routing and signaling. It needs to permit the discovery of topologies and to forward reservation requests.

1.5.8 DRAGON

Dynamic resource allocation in GMPLS optical networks (DRAGON [99]) defines a research and experimental framework for high performance networks required by Grid computing and e-science applications. The DRAGON project is developing technologies and deploying network infra-structure that allows dynamic provisioning of network resources in order to establish deterministic paths in direct response to end-user requests. This includes multi-domain provisioning of traffic engineering paths using a distributed control plane across heterogeneous network technologies, while including mechanisms for authentication, authorization, accounting (AAA) and scheduling. A reference implementation of this framework has been instantiated in the Washington, DC area and is being utilized to conduct research and development into the deployment of optical networks technologies toward the satisfaction of very high performance scientific application requirements.

Figure 1.14: DRAGON Control plane architecture



The DRAGON project provides a topology on demand service (ASTP, Application Specific Topology Builder). It uses a Topology Description Language and interacts with a Network Aware Resource Broker NARB to check if it's possible to set-up the topology requested. Inter-domain routing and path computation are done by NARB. DRAGON also developed a Virtual Label Switch Router to propose an emulation for the GMPLS control plane in a local area network.

To compare with the EGEE architecture, an NARB would be the equivalent of an NSAP.

1.5.9 AkoGrimo

Akogrimo [37] is a project funded by the EC under the FP6-IST program. The project runs from July 2004 until September 2007. Among other objectives, the project aims to bring Grid services to mobile users. To do so, an architecture combining a dual signaling SIP and SOAP is used with signaling occurring in parallel.

QoS is provided in Akogrimo via a QoS broker. The application communicates with the QoS broker via a SOAP/HTTP WS interface.

1.5.10 EuQoS

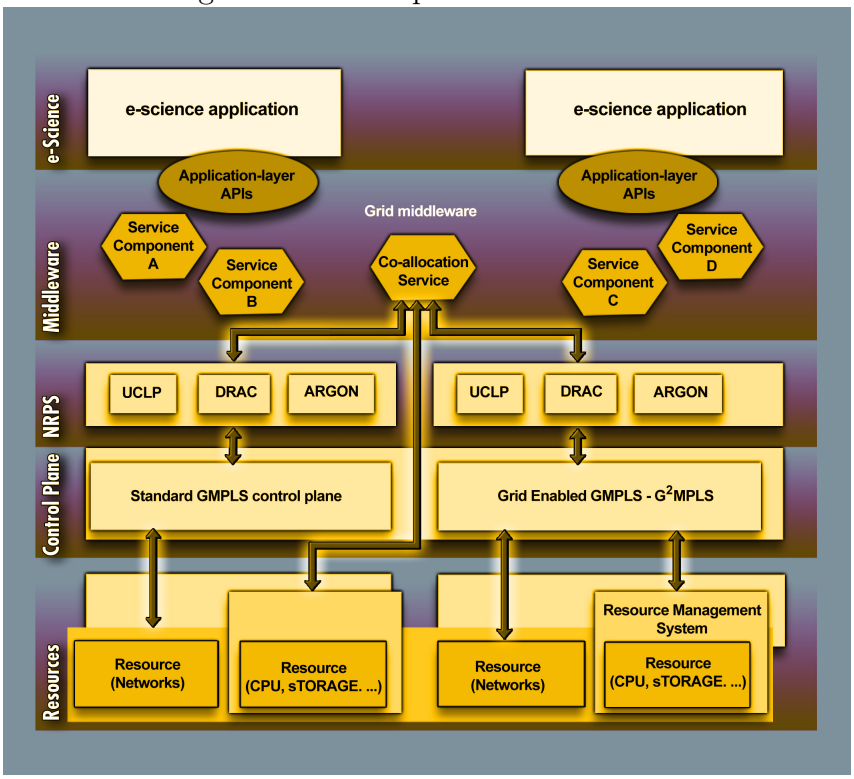
Although, applications targeted by this project are not "labeled as Grid aware" it proposes an interesting framework for the interaction of applications and networks. Applications in EuQoS [133] communicate with Session Initiation Protocol (SIP). A SIP proxy in the network domain receiving the session creation message will trigger the provisioning network resources. The project describes two important scenarios. The first scenario involves two methods of signaling: the first signaling is an "application control signaling" based on SIP, whereas the second is a network control signaling used to reserve network resources based on RSVP-TE. All hosts know both protocols. The aim of the first signaling is to negotiate codecs. Once codecs are negotiated, the bandwidth necessary for the session is known by the source and the destination host, only then can network resource reservation occur. Once network resources are available, ring notification occurs. The second scenario tries to remove this dual signaling: the source host sends SIP messages to its proxy, the proxy is then responsible to trigger network signaling. These scenarios are also described in the IMS architecture, and called the push and the pull model.

A more detailed description of the application of these ideas in a Grid context is described in chapter 2.

1.5.11 Phosphorus

Phosphorus [104] is an IST Grid project started in 2006. One of the objective of the project is to implement a Grid enabled GMPLS control plane (GGMPLS). The idea is to manage computing and storage resources within an extended GMPLS control plane. This idea was investigated by Alcatel-Lucent in 2004. It will be discussed in more details in chapter 2.

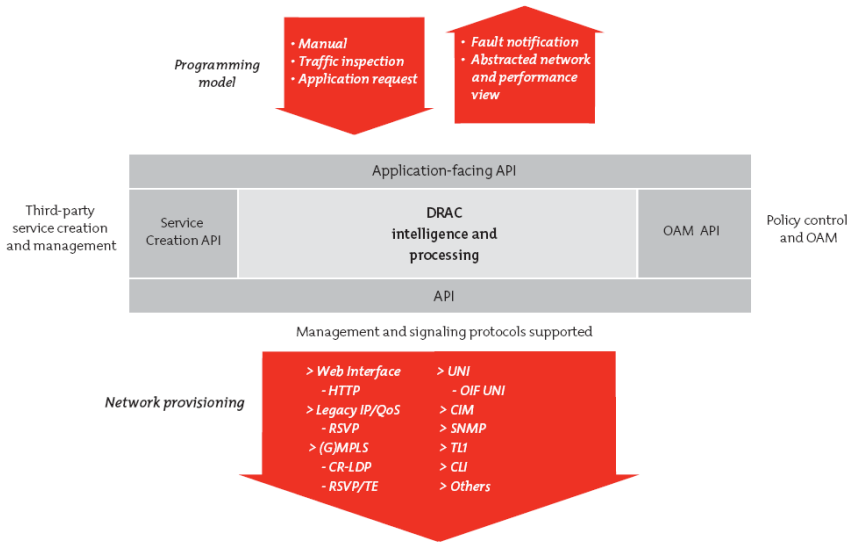
Figure 1.15: Phosphorus Architecture



1.5.12 Nortel's DRAC

Nortel through Franco Travostino was one of the first equipment manufacturers supporting Grid Networks. They launched the Dynamic Resource Allocation Controller (DRAC [16]) in 2004. It is used in SurfNet, the Dutch NREN.

Figure 1.16: Nortel DRAC



DRAC provides a policy based engine to make the link between (Grid) applications and network elements for provisioning, fault notification and abstract network and performance view. DRAC has no inter domain capability and currently works only with Nortel equipment.

1.5.13 VIOLA

Vertically Integrated Optical Network for Large Application [28] is a German research project. They have implemented an optical testbed with the main goals:

- Test of advanced network equipment and architectures;
- Development and test of software tools for the user-driven dynamic provision of bandwidth;
- Inter-operability of network equipment from different manufacturers;
- Enhancement and testing of new advanced applications (e.g. Grid, Virtual Reality);
- Co-operation with similar projects in Europe and elsewhere.

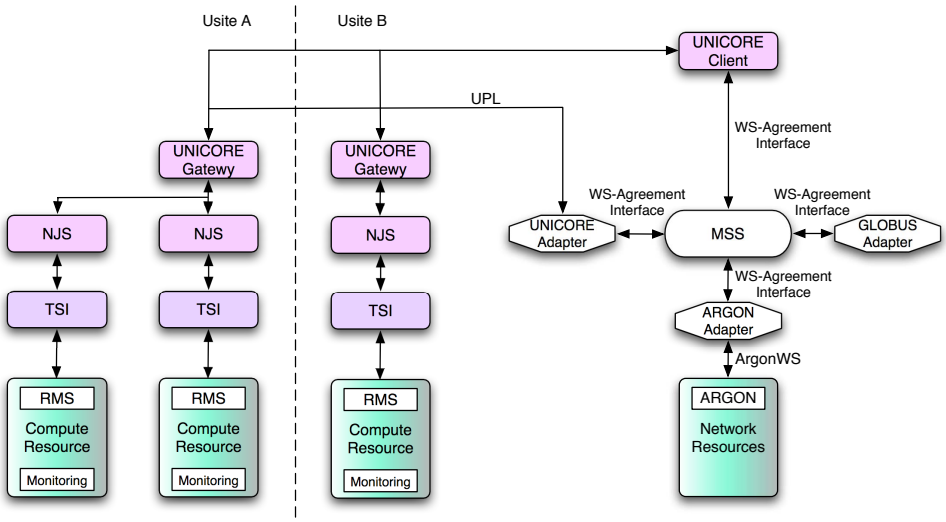
In this framework, a Meta-Scheduling Service (MSS) (a.k.a. Grid Scheduler) was developed [152] to co-allocate network and computing resources.

The MSS is a Grid scheduler capable of orchestrating and co-allocating resources. In order to co-allocate resources the MSS mediates between the Resource Management Systems (RMS) involved. During this process, the MSS creates resource reservations, which are coordinated in time.

The communication between MSS and the involved RMSs can either be done directly or through a standard Grid middleware system. At the moment the MSS supports Unicore [66] as one Grid middleware. Unicore provides seamless and secure access to distributed resources and enables the MSS to access computational resources' management functions in a uniform way, even if the resources are located in different organizational domains. However, since the MSS should be extensible for the future, the integration of the MSS into Unicore was done using the adapter patterns [76]. This allows the MSS to support different Grid middleware systems. Adapters have been developed for computing and network resources. ARGON, Allocation and Reservation in Grid-enabled Optic Networks is the network manager's adapter.

A more detailed description of VIOLA is done in chapter 2, while a performance analysis and comparison of VIOLA's co-allocation protocol is done in chapter 3.

Figure 1.17: VIOLA Architecture



1.5.14 Enlightened

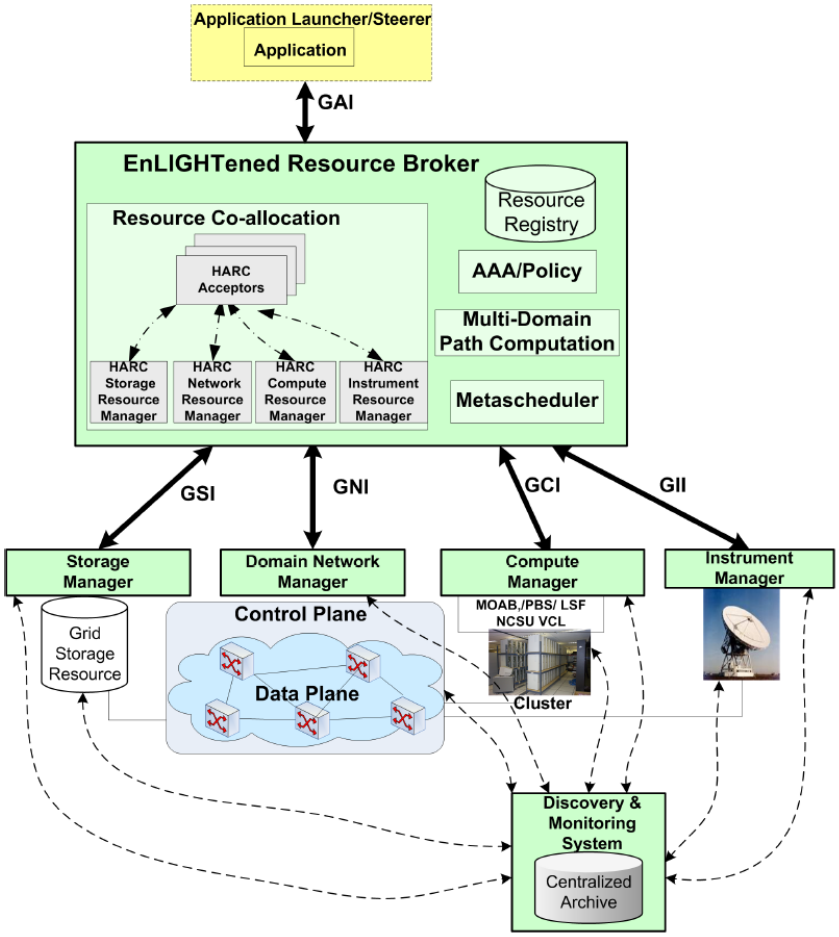
Enlightened [45] is a seed funded NSF research project, which started in 2005. The project has designed an architectural framework that allows Grid applications to dynamically request (in-advance or on-demand) any type of Grid resource (computers, storage, network paths and lightpaths). They are building a network that will interoperate with other major research initiatives of this type, like Phosphorous, Geant 2, G-Lambda, ...

Its architecture is similar to the one proposed in chapter 2 section 2.4. Alcatel-Lucent are currently developing ideas that are similar, although they have not yet fully specified protocols for the Grid Network Interface.

Figure 1.18 shows Enlightened's architecture. "The Grid application uses the Application Launcher/Steerer to initiate its activities. The Application Launcher/Steerer makes a reservation request for Grid resources from the Enlightened Resource Broker. Enlightened Resource Broker queries its Resource Registry and the Discovery and Monitoring System and then chooses the appropriate resources. The Broker then contacts the Highly Available Resource Co-allocator (HARC [102]), which attempts to co-allocate the required resources for the selected time range by using the HARC's Resource Managers for network, compute, storage and instruments." So far, the main focus of the project team has been the design and development of the Domain Network Manager.

HARC is an extensible, open source co-allocation system that allows clients to book multiple heterogeneous resources in a single transaction. HARC uses the Paxos Commit protocol [79, 97]. The messages exchanged are XML over HTTP. The main difference with the proposal of chapter 3 is that chapter 3's proposal relies on WS-Agreement standard while HARC uses its own proprietary protocol. Diagram 1.19 sums up the other main differences between both approaches. In HARC, messages are plain XML messages sent

Figure 1.18: Enlightened Architecture



directly over HTTPS (POST and GET). SOAP is not used. Some consider that the use of POST to send information and GET to retrieve it is a REST like behavior, even if HARC does not support DELETE and PUT.

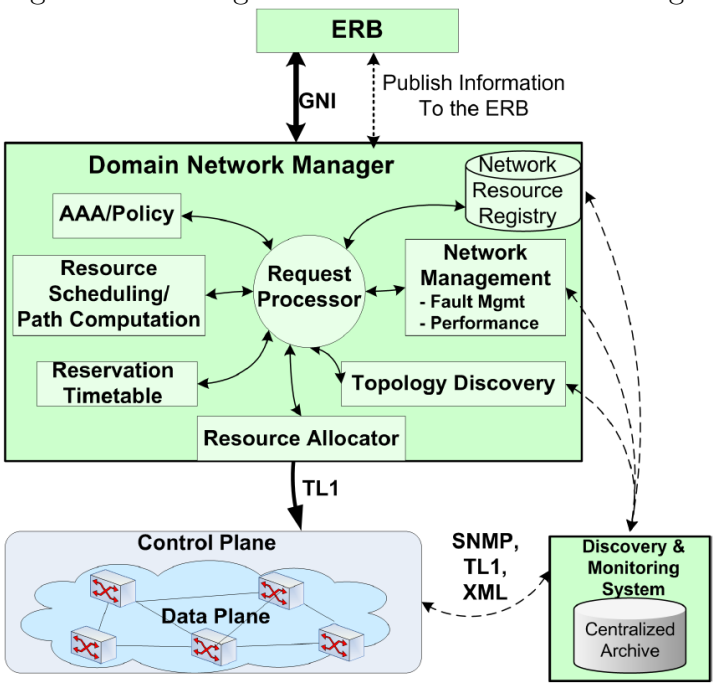
Figure 1.19: HARC vs VIOLA

	Web Service	Create Co-Alloc	Modify Co-Alloc	Move Co-Alloc	Delete Co-Alloc	Commit Protocol	Negotiation	Hide Heterogeneity
HARC	REST	Yes	Yes	Yes	Yes	Paxos (2PCP)	Timetable	Resource Managers
VIOLA	SOAP	Yes	No	No	Yes	1PCP*	Earliest time	Adapters

*2PCP see chapter Protocols

HARC is composed of clients, a set of replicated processes called acceptors that play the role of the coordinator, resource managers that are like VIOLA's adapters. Clients first communicate with resource managers to get resource availability in the form of a timetable, then to the coordinator for the reservation of resources. The coordinator is in charge of creating the reservation with the different resource managers.

Figure 1.20: Enlightened Domain Network Manager



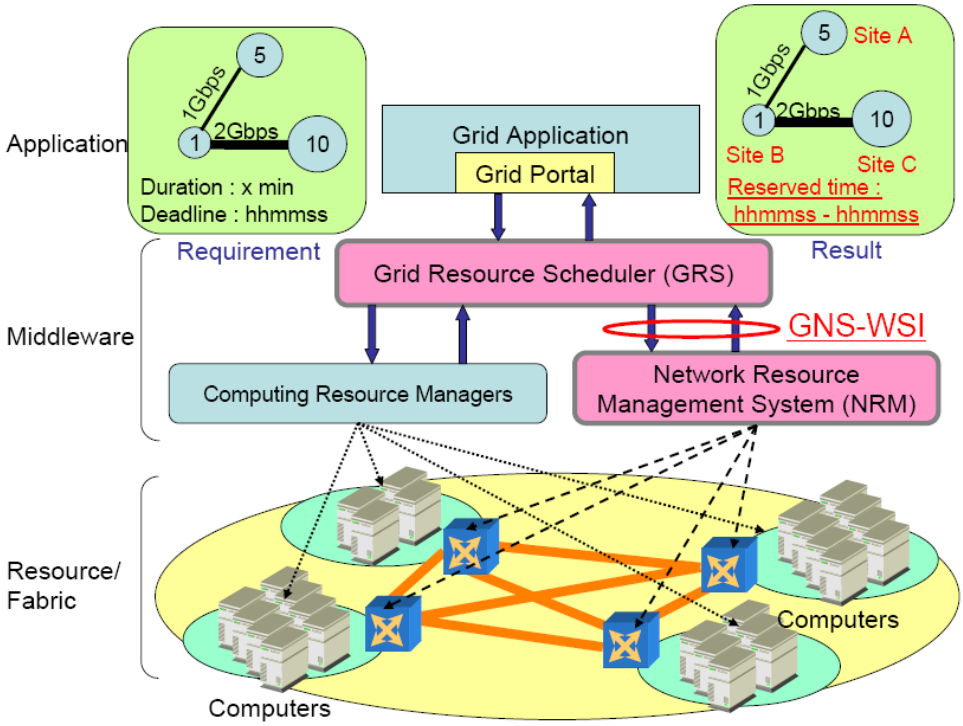
The Domain Network Manager controls all network resource allocations to dynamically setup and delete dedicated circuits using GMPLS control plane signaling; see figure 1.20. It does the resource scheduling (both links and wavelengths) and path computation within an Administrative Domain. It also keeps the network reservation timetable. It obtains network topology information and up to date status of the network resources from the Discovery and Monitoring System. It uses its Resource Allocator to talk to the network control plane to manage (setup/tear-down) the lightpaths. This implies a novel relationship between the middleware and the network control plane, as the Domain Network Manager will be responsible for both path computation and resource allocation. In the Enlightened testbed the resource allocation is done via TL1 by instantiating a GMPLS RSVP-TE command. The resource scheduling and path computation component of the Domain Network Manager supports in-advance reservations.

1.5.15 G-Lambda

G-Lambda [146] is a Japanese project which started in December 2004. The goal of the project was to provide a standard interface between a Grid resource manager and a network resource manager provided by a network operator: GNS-WSI. The latest version as of 2007 (version 2) was developed with the WSRF. The G-Lambda interface, GNS-WSI can be used for inter domain provisioning if used in conjunction with a coordinator, but as is, there is no communication between two NRMs belonging to different operators.

G-Lambda and Enlightened conducted an interoperability test in 2006 to provision an

Figure 1.21: G-Lambda Architecture

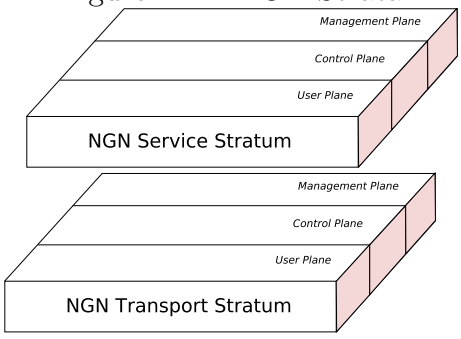


inter domain label switch path (LSP).

1.6 Conclusion

Table 1.23 summarizes all of the project's characteristics. Next Generation Network strata were considered to classify the different types of signaling involved in these projects. Diagram 1.22 illustrates the International Telecommunication Union NGN model. NGN

Figure 1.22: NGN Strata



is a model to deliver commercial services on any device, such as personal computers, digital assistants, mobile phones connected to any access network, wired or wireless. This

model is composed of two strata: the service and the transport stratum. Each stratum is composed of three planes: a user plane, a control plane and a management plane. The user plane encompasses functions to transfer user data, the control plane encompasses functions related to the instantiation of the service, such as set-up, execution, tear-down for a given user session and the management plane encompasses functions related to the billing and monitoring. OGSA functions belong to the service stratum and form its control and management plane. Sometime also referred to as the application control and management plane.

Before NGN, when an operator wanted to deploy a service, it had to build the corresponding network. For instance to offer a telephone service, it deployed a fixed telecommunication network, a mobile telephony service, a mobile telecommunication network, an Internet access or an Internet Protocol (IP) network. One of the NGN objectives was to rationalize network deployment costs and have one network for all services. Most Grid deployments over wide area networks have followed the traditional approach: sites are interconnected by a dedicated network. They have not followed the NGN principles. Some deployments are using the Internet, sacrificing performance over cost.

Given this model, application (or service) signaling can be distinguished from network signaling, labelled respectively A and N in table 1.23. Signaling is a function of the control plane to "alert" next elements that they should do something or react. The NGN model distinguishes two level of signaling, network and application. Examples of network signaling are network resource reservations like RSVP-TE, application signaling like SIP, or the signaling inter bandwidth broker like SIBBS. It should be noted that DiffServ doesn't need signaling. So the signaling column of table 1.23 when QoS "Type" is PHB (i.e. DiffServ) is necessarily an application signaling.

When the QoS "type" is connection oriented, it is assumed that a distributed control plane like (G)MPLS is always used, except for GridJit [147]. In this case, if the originating signaling node is a host, the signaling is end to end, if it's a router in the operator's network, it's edge to edge. Table 1.23 considers that there is Network signaling only when it's an end to end signaling. In the edge to edge case the LSP setup is triggered by a vertical interface. Inter-domain TE routing and scheduling message exchange (like DRAGON NARB) are not network signaling.

A connection oriented paradigm is the major concept being investigated. It is the only way to really guarantee QoS and it does not preclude any per hop behaviour. Inter domain traffic engineering routing issues were not considered in this thesis. The main focus was once Grid resource management software manages network resources, how can the Grid scheduling be improved? Chapter 2 presents the architectures proposed to integrate network and computational resources. The architectures proposed in the context of this thesis were proposed in parallel by other research projects during this period. Chapter 2 will detail the contribution made to the development of new technologies, as well as describe in more detail the three major architectures:

Figure 1.23: Survey summary

	QoS "Type"	Multi-Domain	Signalling Level	Year of last activity reported
NRS	PHB	No	Appli	2005
GARA	PHB	No	Appli	1999
GARA extensions	CO	No	Net	2004
Qbone	PHB	Yes	Appli	2001
UCLP	CO	Yes	Both	2007
Aquila	PHB	Yes	Appli	2003
GridJit	CO	Yes	Net	2004
EGEE	Both	Yes	Both	2007
DRAC	CO	No	Appli	2007
AkoGrimo	PHB	No	Appli	2007
EuQoS	Both	No	Both	2007
VIOLA	CO	No	Appli	2007
Dragon	CO	Yes	Net	2006
Enlightened	CO	Yes	Appli	2007
Phosphorous	CO	Yes	Net	2007
G-Lambda	CO	No	Appli	2007

PHB : Per Hop Behaviour
CO : Connexion Oriented

- WS based (EGEE or VIOLA multi domain like),
- GMPLS based (Phosphorous),
- SIP based (EuQoS).

Chapter 3 will evaluate the performance of the protocol used for co-allocation in the dominant solution (WS-based). Chapter 4 proposes a cross-optimization algorithm and evaluates its benefits to better co-allocate network and computational resources.

Chapter 2

Architectures

The previous chapter presented different architectures to co-allocate network and computational resources. This chapter presents the architectural contributions developed during this thesis.

Chapter 4 proposes new algorithms for Grid schedulers to better co-allocate resources, minimizing a job completion time seen by the end-user. These algorithms provide what is called "a cross-optimization" (XO). This chapter provides an answer to the question: "In which architecture these new algorithms can be included?" But to do so, a review of the services that are to be provided by the network operator is needed. Even when services provided by the telecommunications company are well known, the survey conducted in chapter 1 illustrated the number of potential architectural solutions. All these architectures could be a target for XO algorithms. Section 2.1 provides a unifying vision of the features and services that could be provided by a telecommunications company to support or provide computational services. It will also describe how each project discussed in the previous chapter can be included into this vision.

Section 2.2 details how XO is included into VIOLA.

Section 2.3 proposes some enhancements to network protocols to better support Grid application's needs and co-allocation. Those extensions were patented and are part of this thesis contribution.

Section 2.4 describes in more detail a very network oriented architecture, GGMPLS, to better support Grid applications. This idea was also patented. Sections of the research community are currently heading in that direction.

Section 2.5 briefly describes IP Multimedia Subsystem architecture, the fixed/mobile network architecture for service delivery and proposes some extensions to provide computational services. This section complements a published article I wrote.

2.1 A unifying vision

2.1.1 Introduction

Grid doesn't refer to any precise type of resource: Computational, Storage, Sensor, Instruments, Captors... Whatever the service provided to the end user, it requires access to the corresponding resources. Those resources will be heterogeneous and not collocated. The service provider will have to aggregate the information coming from all of the resources and make them seem to behave as if they were all similar: it will provide a virtualization of the resources. For example a computational service will aggregate the computing power of different clusters spread accross the network and provide a simple and common interface to its customers.

As described in the general introduction, the telecommunications company Grid service provider can be a new department, which will be a customer of the network department. The introductory chapter contains some examples of different Grid services that can be provided (computing at the three different level, storage, orchestration, ...). Section 2.1.3 explains the different features characterizing a computing service. Section 2.1.2 describes services that can be provided by the network department to support the computing department. The interface between the two departments can authorize a different level of information to be exchanged. It will provide structure to the different kind of interactions between the network and the Grid infrastructure.

- A *public* interface discloses as little information as possible to the user
- A *private* interface discloses abstract network information to the user in order to help them makes his decisions

This raises the question, what services should be provided through this interface ?

2.1.2 Network services

Network services can be classified by two types:

- Connectivity services provides connections (may be virtual) or QoS guarantees
- Publication services provides information about the network in an abstract form preserving confidential information

Of course "Premium IP" services with DiffServ could also be provided but are not discussed here. The focus is on signaling approaches like MPLS or GMPLS.

Connectivity services

The questions a network operator must answer to provide such connectivity services are:

1. Will the QoS guarantee be end-to-end or edge-to-edge?
2. Will the QoS guarantee provided be inter-domain and involve multiple operators?
3. How much information should be provided to the users?
4. What is the correspondence between a QoS guarantee request and LSP provisioning?

The first question will distinguish architectures that require just a service signaling from those that require a service and a network signaling (dual signaling). An example of a dual signaling is detailed in section 2.5. If an LSP has to be set-up, according to the policies, the edge-to-edge scenario does not require the network signaling to start from the customer's network. In this case, the LSP set-up trigger comes from a vertical interface. The end-to-end scenario requires the network signaling to start from the source router in the customer's network. The operator must have security mechanisms in place to check that a customer's initiated network signaling is authorized. The second question is crucial to determine whether the network operator has to be involved or not with others to provide inter-domain QoS guarantees. Today, in public networks, only a limited number of operators have agreements to treat QoS packet classifications. What is required by Grid applications are on-demand in advance inter-domain LSP set-up. Only research networks have been able to provide such services. Furthermore no TE BGP extensions have been widely accepted. Only some of the solutions will be proposed to deal with the inter domain scenario. It is thought that the network operators are not yet ready to collaborate, delivering on demand QoS guarantees. (Despite initiatives like IPSphere and others) The third question is very important to network operators. They are very secretive when it comes to disclosing network information, but it's an important discriminating factor in the selection of an operator. Chapter 4 requires a private interface to have access to the network topology and a known amount of reservable bandwidth. The last question is a policy based decision that must be made by the network operator. Those policies are out of the scope of this thesis.

The main Grid applications connectivity service requirements can be summed up by two services [135]:

- a bandwidth on demand service
- a low-latency service

The bandwidth on-demand service provided by the network operator must guarantee a requested throughput. The low-latency communication service provided by the network operator must guarantee that packets are delivered within a delay boundary. Grid

applications must be able to trigger those services without human interaction. Furthermore, these services must provide advance reservation capabilities: the ability to reserve in advance network resources for a future use. For both of these services, protection and restoration schemes should be offered. This will guarantee the reliability of the communication. Of course security of transmission is an important requirement, but not specific to Grid applications.

Publication services

The main publication services requirements are:

- An abstract network topology
- Reservable Bandwidth or other performance related information
- Monitoring services
- Failure Notification
- Network service capabilities publication

The network service capabilities information is published and made available to the subscriber of the service and is controlled by an authentication process. The two most important requirements are monitoring and failure notification. Their importance is even more critical if the network operator is not responsible for setting-up inter-domain LSPs. Publishing an abstract network topology can be done at different levels: inter AS, aggregated internal network view or full internal topology. The level of disclosure defines whether the type of interface is public or private. AS availability information is the minimum needed by the Grid service provider if it is in charge of coordinating multiple LSP setups in different networks (as explained in State of the art section 1.5.1). Additional performance information can be published like the reservable bandwidth or average delay towards each AS. The *XO* algorithm of chapter 4 requires the full internal topology and the amount of reservable bandwidth on each link. Capabilities of the connectivity service like the switching technology, the SLA template, or any additional useful information should be provided in order to automate the process.

2.1.3 Integrated Computing services

To provide a integrated computing service, several functions have to cooperate: Resource Discovery, Resource Selection, Resource Reservation & Execution. Examples of Grid toolkits and their major components are given in chapter 1. The focus is on cross optimization. What features are needed to provide a cross-optimization?

- Information Gathering about computing and network resource, static (topology, location, Processor type...) and dynamic information (availability, reservable resources, ...)
- Multi-criteria resource requirement description & selection
- Policing (by the Grid service provider, the resource provider, the user)
- Negotiation
- Optimal Resource Selection (with the *XO* algorithm for instance)
- Reservation

Not all of these features are mandatory. Section 2.2 describes the modifications needed to existing toolkits to provide a cross optimization. These modifications do not provide all policing features one could imagine. An architecture that could work has been proposed but this ideal system hasn't been designed or tested yet.

2.2 WS approaches

2.2.1 Introduction

This section has two objectives: To describe in more detail a Grid toolkit architecture and to propose some extensions to embedded *XO* algorithm proposed in chapter 4.

As of 2007, most Grids used toolkits such as Globus, Unicore and maybe G-Lite. Sun Grid Engine is more used as a cluster management software tool and not a Grid toolkit. The main difference is that resources managed by SGE are usually located in one site and belong to the same administrative domain.

Globus and Unicore in their basic versions have no Grid scheduler. Both basic toolkits cannot interact with network connectivity services like Bandwidth on Demand services or the publication services of sections 2.1.2 and 2.1.2. The focus of this section is on the architectural aspects of co-allocation. The missing modules and interaction needed to support co-allocation and cross optimization will be described. Differences between execution management, resource discovery & monitoring, AAA [1] will not be discussed more than they have already been in chapter 1.

Grid schedulers have been developed for both toolkits: GridWay [108] for Globus, Meta-Scheduling Service [152] for Unicore. These Grid schedulers contain the scheduling

1. Authentication, Authorization and Accounting

algorithm capable to optimize resource allocation. This is where *XO* algorithm should run. The interaction between the Grid scheduler and the network connectivity and publication service is described in the following.

2.2.2 Architecture

The architecture of VIOLA and its different components will be described to provide a global overview of Grid schedulers.

This section focuses on VIOLA's meta-scheduling service and architecture. How VIOLA provides support for Grid applications is explained (see also chapter 1). The VIOLA project and its unique features will be placed in context with other major Grid projects. VIOLA's Grid support is provided by: Unicore, a Meta-scheduler and several adapters for the communication with local resource management systems, such as local schedulers and network management systems.

2.2.3 Components

VIOLA relies on the Unicore toolkit that has been described in section 1.4.3 in chapter 1. One of the project's objectives is to propose a Grid Scheduler (MSS in figure 2.1) that is able to co-ordinate different types of resources, like network and computing. For each job request, the Grid scheduler will reserve network and computational resources. The Grid scheduler has to find a computing resource that is available at the same time as the network resources. To do so, the Grid scheduler interacts with:

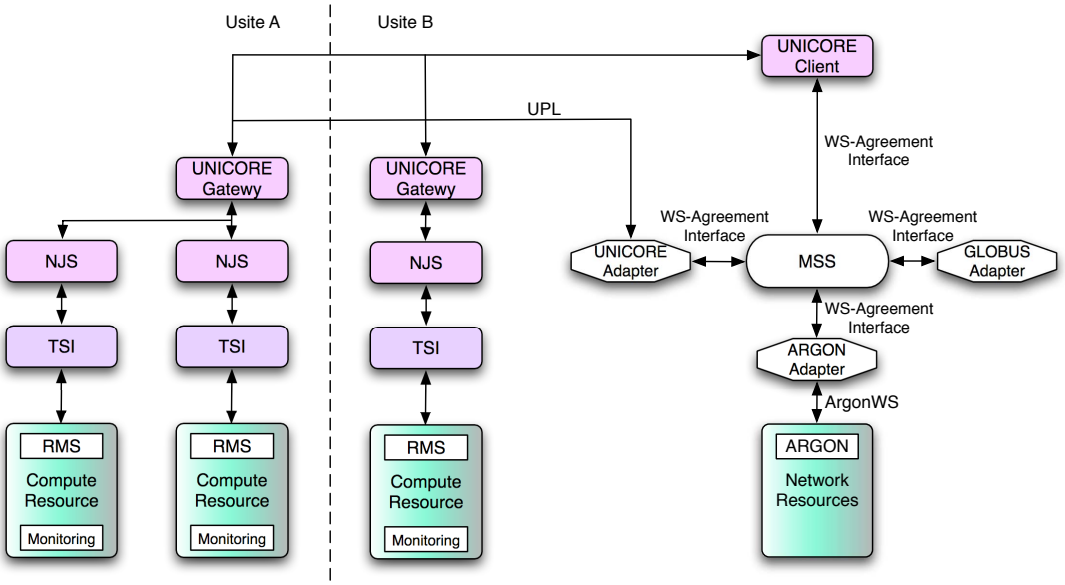
- A client that will send a job request, a Unicore client meta-scheduler plug-in
- A local resource management systems through different adapters

The role of the Grid scheduler is to find and reserve resources for the job request. Once resources are identified and reserved, the unicore client is notified and can submit the job to the relevant sites almost as if the Grid scheduler had never existed.

Client

The unicore client is a set of applications that enables an end-user to submit jobs on a Grid. One of the client applications is a graphical user interface that enables the creation, submission and monitoring of jobs. A job can be composed of several sub-jobs. The MS is not part of the standard client, hence a job must be assigned to a resource. The

Figure 2.1: VIOLA and the Meta-Scheduling service



unicore client contains a list of the available resources. The client communicates with the meta-scheduler through a plugin which is not represented in the previous diagram. The plugin communicates with the MS using the WS-Agreement protocol (see chapter 3).

Grid scheduler

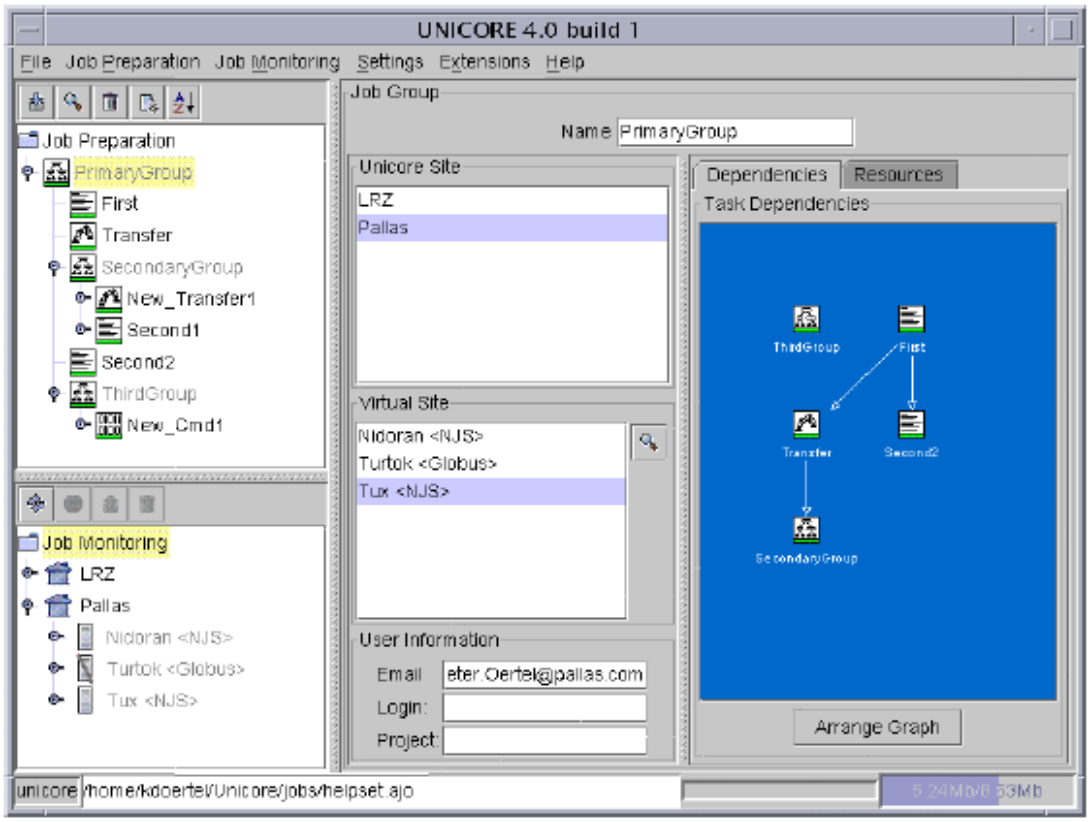
To achieve co-allocation [152] of resources managed by multiple, usually different scheduling systems, the minimal requirement these systems have to fulfil is to provide functions to

1. schedule a single reservation some time in the future (e.g. “from 5:00 pm to 8:00 pm tomorrow”)

2. give an aggregated overview of the usage of the managed resources between now and a defined time in future.

Once a reservation is scheduled, the starting time is fixed, i.e. it may not change except for the reservation being cancelled. This feature is called “advance reservation”. There are at least two possibilities to create an advanced reservation. The first is to schedule a reservation for a requested time, called “fixed time scheduling”. The second is to schedule a reservation not before a given time, which means a scheduling system tries to place the reservation at the requested time, otherwise it will be scheduled for the earliest possible time after the requested time. This is referred to as first fit scheduling. The Grid scheduler implemented interacts with first fit local schedulers but is not limited to these. The

Figure 2.2: Unicore Client



main function of the Grid scheduler are to negotiate the reservation of network-accessible resources that are managed by their respective local scheduling systems and to request reservations. The goal of the negotiation is to determine a common time slot where all required resources are available for the requested starting time of the job. The major challenges are:

- to find Grid resources suitable for the user's request,
- to take security issues like user authentication and authorization into account,
- to respect the autonomy of the sites offering the resources and
- to cope with the heterogeneity of the local scheduling systems.

As discussed earlier, the Grid scheduler should run an *XO* algorithm described at the end of chapter 4.

The Grid scheduler negotiation mechanisms and protocol will be discussed in more detail in chapter 3. The Grid scheduler communicates with the unicore client to receive job requirements and with local resources through a common interface: the adapter.

Adapters

The adapter's main function is to hide the local resource management system's heterogeneity. It provides a common interface independent of the underlying resource, whether it's a computing or a network resource. The adapter interface provides 5 different functions: couldRunAt(), submit(), cancel(), state() and bind(). The bind() function is specific to the network adapter. These functions are implemented using web service interface (SOAP). Following are their prototypes, brief descriptions and sample code:

CouldRunAt `Date CouldRunAt(Job job_description)`

: returns the earliest possible runtime for a job, the `job_description` can contain a minimum starting time offset.

Submit `String submit(Job job_description)`

: submits a job to a local scheduler. The `submit()` call results in an advance reservation being requested on the specified resource. The string returned is a unique reserved job id.

Cancel `Boolean cancel(String job_id)`

: cancels a previously made reservation.

State `Job state(String job_id)`

: provides the state information relating to the reservation, including the scheduled start time.

Bind `Boolean bind(AddressList addresslist)`

: `Bind` effectively maps reserved resources with real IP addresses and ports, because the ports are not known during the reservation. They are only known once the job execution starts. The return value is a success or failure indicator.

In VIOLA, the network is managed almost like any other resource on the Grid. A special network adapter has been developed to provide the interfaces described above: ARGON, Allocation and Reservation in Grid enabled Network [44]. ARGON provides one interface to the Grid scheduler and another to the real network resource management system. The network adapter must contain at least the following components: Future TEDB, Network Topology and Policy Repository. The future TEDB is the database containing the amount of bandwidth available for reservation on each link in the future. Amongst many other internal functions, the most important ones are:

- The ability to reserve a connection giving a source-destination node pair, start time and duration
- The ability to cancel a connection given a connection reference

- The ability to find the first available time slot where there is a path available for a given source-destination node pair and duration.

ARGON's internal functions and external interface are similar to those that should be provided by a connectivity service network management extension module.

An example of algorithms to find the first available time slot where there is a path available is given at the end of chapter 4.

In a very similar way as ARGON, adapters for computing resources have been developed but won't be discussed here.

One could extend this interface to provide a cross optimization. *XO* requires the full topology with the reservable bandwidth on each link to better select network and computational resources. This network information should be published through an interface's extension. Such interface could be implemented using an XML representation of the network topology and link state. The information could be adhering to a publish/subcribe communication model with WS-Notification for instance.

2.2.4 Future extensions

OGF is working on inter scheduler protocols and architecture, to build a distributed scheduling architecture. Today, the Grid scheduler communicates with the network and computing scheduler in a hierarchical way. Leaving each scheduler its autonomy, i.e. the ability for local users to submit network request and jobs on their respective schedulers.

VIOLA is not capable of automatic discovery of resources. These mechanisms should be developed and applied to computing and network resources.

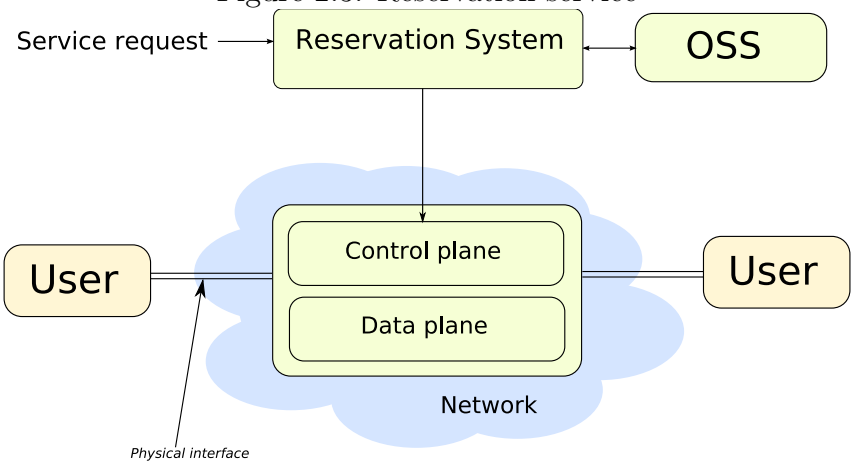
2.3 Control plane time extensions

Co-allocation requires the network to be able to reserve in advance network resource. Today's network control protocols cannot do it. The simplest way to provide this service would be to develop a module on top of the network management system that would receive all network reservation request, answer them and provision the network accordingly. Such ideas are being proposed at IETF by Yong [36]. Such modules would need to perform the following functions:

- Provide a network reservation interface

- Use a network provisioning interface
- Maintain a database of network resource affected over time
- Maintain a database of network resource reservations over time
- Allocate network resources to network reservations through an enhanced routing algorithm for planned requests

Figure 2.3: Reservation service



Operations Support System (OSS) includes the network management systems, billing systems and required business operations functions.

Other approaches require "Future" extensions to existing protocols:

- Future Signaling (RSVP-TE extensions)
- Future Computation (PCE)
- Future Routing (OSPF-TE)

Those extensions were already subject to patents written in the last few years held by Antoine Pichot and Alcatel.

2.3.1 Future Signaling

Instead of building a box on top of the network that will use a network management interface to trigger the provisioning of network resources, it's possible to extend the existing control plane to include new functionalities, like advance signaling. To do so, a change

in network resource reservation protocol (RSVP-TE) is needed to support advance reservation. This modification simply requires RSVP-TE to include two new fields: a start date/time and a termination date/time.

When a node needs to reserve resources in advance, the head node forms the RSVP-TE PATH message, with unmodified fields as before and new fields with the date of the reservation start and end time. The tail node forms the RESV message with the new fields. The same mechanisms apply except that the computed route is based on the future network contained in the Future TEDB. This new database is a prediction of the TEDB. It can be populated by the network management plane or distributed by a future routing protocol (see below) or populated by a dedicated service, like a network weather service. Such network weather services could aggregate all future network reservation requests and be a policy decision point for future request. One could imagine that all in-advance reservation requests could trigger a request to this element by a policy.

2.3.2 Future Computation

The Path Computing Element PCE activity at the IETF has started in 2004. The PCE is a "method" to "externalise" the path computation function from the network node. It standardizes the communication protocol between a Path Computation Client (PCC) and a Path Computation Element (PCE). Recursive computation can be done and a communication protocol inter-PCE is also being standardized. The advantages of this architecture are:

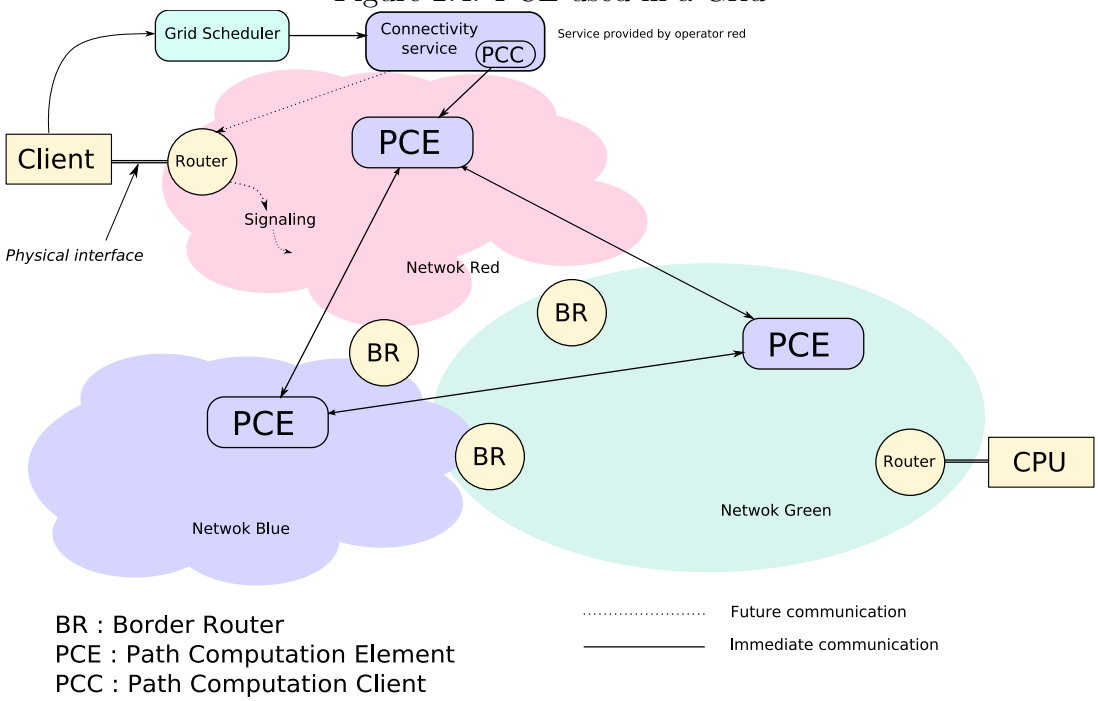
- Reduced node controller load from CPU intensive path computations
- Permit the path computation if a client has only partial visibility on the network
- Compute when nodes are outside the network domain
- Facilitate inter-domain TE path computation

The entire list of advantages can be found in the architecture RFC [35].

It was proposed to rely on PCE to compute the advance route in the network. A connectivity service could then be able to provide an inter-domain connectivity service relying on the PCE.

Diagram 2.4 shows an example of such an architecture. A client initially submits a job request to the Grid scheduler. In the simplest case, the Grid scheduler does not know the network topology or anything about the network resources, it must request a connectivity provider to reserve future virtual connections. To do so, it asks the connectivity service to reserve resources. The connectivity service must compute the future path. The plain

Figure 2.4: PCE used in a Grid



arrows in diagram 2.4 show these steps. Once the path computation is performed and the connectivity service knows which path will be used by the inter-domain connection, it must reserve in advance resources. To do so, it can use an inter-domain reservation mechanism, which could be an extension of the PCE as the diagram suggests or a future signaling as explained above. Using the PCE (in the unsolicited mode) to reserve resources is a possible use of the PCE that is not currently standardized and needs extensions to support in advance reservations. In this mode, inter-PCE communication should compute the path and reserve network resources and then when the resources are really needed the network signaling can be triggered (dotted lines). The network signaling must contain a token identifying that the resource had been previously reserved. In the later case, when the PCE cannot reserve the resources in advance, a future signaling message must be sent along the path to reserve the resources.

2.3.3 Future Routing

An important element to provide advance reservation is to rely on the future network state. Today's solutions use a centralized architecture, which has the knowledge of present and future use of the network. This centralized architecture has access to future connectivity requests between every point in the network and the future network's topology. When a new route needs to be set up for a future use, a human being launches an optimization algorithm on a network's manager station. This algorithm computes future routes in the network to satisfy new connectivity requests. Once new routes are computed, they are

downloaded to every node in the network, but only when these new routes are needed.

This centralized architecture forces a central server to administer and collect connectivity requests and to perform all route computations for every node in the network. Furthermore, there is no global architecture that automates the whole process: future connection request expression, network's weather forecast service and future routes computation. Currently, planning involves a human being, making the process not automated. The idea of future routing is to distribute the network forecast service in every node in the network, so that every node has the knowledge of the future network's state, more precisely, the future link's state prediction. As a consequence, every node would be able to perform future route computation.

Today's signaling and routing protocols enable an approximately immediate reservation and routing of connectivity requests. OSPF-TE protocol contains and diffuses link's state information on the present use of the network. Signaling protocol, RSVP-TE, enables a connection to be "instantly" set up to reach a given destination. When an ingress node receives a connection request it computes an Explicit Route Object that represent the path to reach the destination. The signaling protocol then reserves resources on every link along the path. These two protocols work well for immediate requests. The idea of future routing is to add time information into OSPF's Link State Advertisement messages to indicate that the state advertised corresponds to the state predicted for the time period given in the message. This time information could be stored in a new OSPF-TE's link sub TLV [2]. Every node would then flood the network of future link's state prediction, in order to build the future link state database. Then, when an ingress node receives a new connection request for future use, it has the ability to compute a future path for that destination.

The new link state advertisement messages generates traffic on the network control plane. This increase in the control plane traffic must be carefully controled in order to avoid flooding the control plane and undermining its reliability. This problem has not been studied in detail and is left for further analysis.

2.4 Grid GMPLS

GMPLS is a set of control protocols to advertise and propagate the network's state (TEDB) and reserve network resources. The idea of GGMPLS [132, 34] is to re-use these protocols (RSVP-TE) and (OSPF-TE) to advertise and reserve other resources, like computational and storage.

A new type of Opaque Link State Advertisement (LSA) object is proposed for the computational and storage resources. Opaque LSA are OSPF-TE objects conceived for

2. OSPF-TE object, (Type, Length, Value)

future use. The new Opaque LSA would be advertised by the resource and propagated in the network by other routers as all Opaque LSA. Only Grid applications, other resources and GGMPLS routers would be capable of understanding the new Opaque LSA. Information in the Opaque LSA could support service Grid ID, Maximum CPU power, available CPU power, Maximum storage capacity, available storage capacity, ...

In parallel to the routing extensions, resource reservation extensions are proposed to support computational and storage resource reservation. The new RSVP-TE would include new fields such as Grid service ID, Max CPU power utilization, data storage utilization, or any other Grid service parameters.

In 2004, these ideas have already been proposed [135] but not detailed. The interface built on top of GMPLS is called Grid User to Network interface (GUNI). It's capabilities are very ambitious:

- Job Management (Job request, submission, monitoring, control)
- Resource Management (Reservation, Brokering, Scheduling, Monitoring, Aggregation)
- Provide Grid Security Services
- State Representation and Manipulation
- Ability to access and manage the state
- Notification
- Flexible Bandwidth Allocation
- Support for claiming existing agreements
- Automatic and timely light-path setup
- Fault detection, protection and restoration
- Propagation of service and agreement related events
- Traffic classification, grooming, shaping and transmission entity construction

The benefits of extending GMPLS mechanisms are never given besides a "global integration of network resource and other resources". In the last years, network have seen the emergence of MPLS and GMPLS as a common control plane supporting different protocols (IP, ATM, Frame Relay, Ethernet) and integrating the control of all switching capabilities. This trend was driven by the huge costs operators were facing managing all these technologies simultaneously. One operator had to manage and have dedicated equipment for every technology. When resources are managed by different organizations,

having an integrated systems is not always a benefit. What is required is co-ordination, collaboration, co-allocation mechanisms but not necessarily with the same technology. Furthermore, GMPLS manage binary objects, not textual objects. This heavily slows down all development processes on top of this technology.

GMPLS is being standardized by the network community that do not necessarily understand the IT community. This will make all extension standardization if not impossible, very slow. GMPLS was never developed to manage another type of resource and contains some inherent missing features (security, refresh intervals, ...) GMPLS is running in very reliable, stable and highly availability environments, any modification to the standard (in terms of refresh interval for resource status) is likely to generate traffic on network controllers whose reliability could not be put at risk. This will make operators very reluctant to deploy this kind of architecture. In parallel to these very slow developments of GMPLS, existing toolkits and functions already exist to manage Grid resources. They use protocols which are very easy to manipulate as a developer as they are XML based. They are also extensible and easily reused (service oriented architectures).

This integrated architecture may provide no additional benefit, because in one respect, drivers for such an integrated architecture are not clearly defined and in the other, making modifications to GMPLS is a very slow process while existing toolkits already do the job.

2.5 IMS extensions

IP Multimedia Sub-system (IMS) [14] is the network architecture for NGN being standardized by the 3rd Generation Partnership Project (3GPP) [25], it proposes a common architecture to deliver multimedia services: voice and video through any kind of device. Additional services related to multimedia are also supported by the IMS architecture. Actors of the IMS business models are end-users, network operators and service providers hosted on application servers. One of the main IMS objectives is to propose a network architecture to efficiently deliver services to any kind of terminal and to ease the integration of 3rd party services to the existing network infrastructure. IMS is flexible enough to support different charging schemes, pay as you go, prepaid, flat rate and content/context based. Although originally designed to support multimedia services, this architecture to support Grid services is one of this thesis contribution.

The idea to use SIP to deliver resource access services has been implemented by Campi [53, 54, 52, 54].

2.5.1 IMS architecture

IMS architectures can be seen as a two strata architecture as described by the ITU NGN framework: a service stratum and a transport stratum, each stratum decomposed in a user plane, control plane and management plane. IMS architecture identifies functions and interfaces that need to be standardized in each stratum/plane so that the network operator, the service provider and the end-user could play their respective role. As a consequence, protocol choices and functions grouping were made. IMS protocols are developed by the IETF, they include IP, Session Initiation Protocol (SIP), Session Description Protocol (SDP) [85] for the application control plane and Diameter for the management plane. The transport stratum relies on the Internet Protocol and its associated control protocols. As [103] discussed, a simplified IMS architecture will be proposed.

P-CSCF

The first entry point of an IMS terminal in the network is handled by the proxy call session control function, P-CSCF. It is discovered by a terminal using Dynamic Host Configuration Protocol or DNS SRV [83] mechanisms. It's a server in the visited network that acts as a SIP proxy, by forwarding messages to the correct next control node, either an I-CSCF or S-CSCF. It performs the following functions: charging support, QoS management and authorization of users to request for a QoS level, monitoring and identification of the I-CSCF.

I-CSCF

The interrogating call session control function is the contact point for roaming users in a visited network and for external users accessing a IMS network. It assigns an S-CSCF to users performing a SIP registration and route SIP requests coming from external networks to the right S-CSCF. It is the inter-operator entry point in the IMS network.

S-CSCF

Whereas P-CSCF and I-CSCF assure functions related to application control and network control interactions, the serving call session control function is an intermediary to access service platforms. An S-CSCF can be a SIP registrar, to locate end-users; it can behave like another SIP proxy to forward messages to the right CSCF or application server.

Multimedia resources Functions

These functions in the IMS architecture operate on the multimedia flow. They are decomposed into gateway processing and gateway control functions. The processing functions is a media gateway, convert media streams (for example audio encoding, media analysis), multiplex streams and create new streams (for announcement). The control functions control the processing function and are an intermediary with other CSCF. The protocol between the processing function and the control function is H.248 or the gateway control protocol [81]. SIP is used to communicate between CSCF and the gateway control function.

Home Subscriber Server

The home subscriber server is the central repository containing user information, profiles, policies, filters and is part of the management system. It is accessed by CSCF via the Diameter protocol.

Application Servers

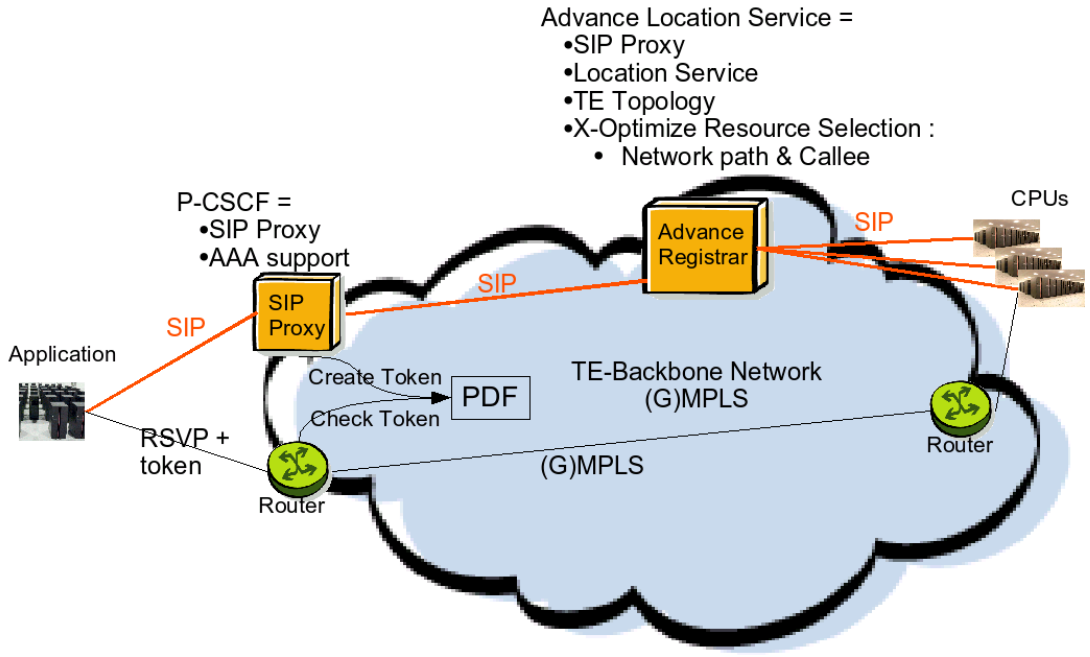
Application servers host service execution logic in the IMS architecture. They communicate with the rest of the IMS control plane through SIP. The IMS service architecture Open Service Access (IMS OSA) provides an API based on Web services: Parlay X R2 [30].

2.5.2 Grid over IMS

It was proposed to build a Grid architecture on top of the IMS architecture. Business actors in a Grid, resource providers, resource consumer, Grid application vendors, could all be considered as IMS users. Indeed, all of them would be identified on the IMS network by a SIP identifier and sessions could be established between them using the IMS. For instance, all storage facilities of a storage area provider X could be reached as storage@providerX.com and a Grid application vendor serviceY.com whose main function is to convert multimedia files could be reached via multimediaprocess@serviceY.com to convert multimedia files. ServiceY.com could use storage resources provided by providerX.com. Users willing to convert a multimedia file would use an application developed by serviceY.com, this application would setup a session with multimediaprocess@serviceY.com to process files.

To understand this proposal, an example of a user willing to process huge video files by a very specific application will be given. Diagram 2.6 represents this scenario. We'll

Figure 2.5: Grid over SIP architecture



consider that the processing application and its input files are tasks spawned by users residing on the "application" site on the left of Diagram 2.5.

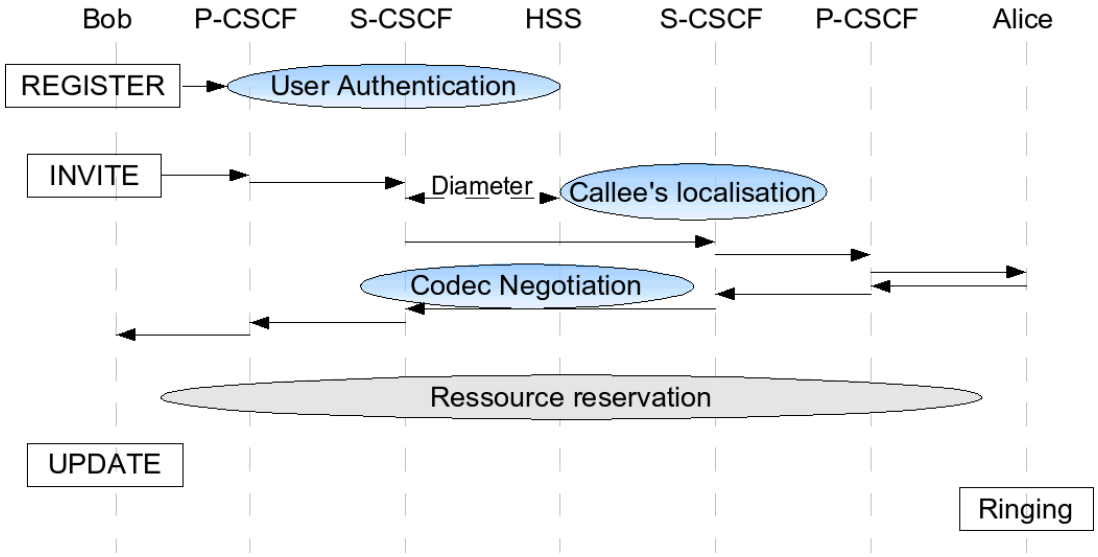
Integration with existing OGSA components

It is assumed that applications running on each host on both sides of Diagram 2.5 have various software components that form the application control plane. Part of this control plane is composed of the SIP protocol and extensions. For instance, some OGSA functions such as workflow management, data functions, self-management functions, are running on hosts. Only a few functions of OGSA need to be modified to integrate them with the control of network resources.

Network operator's architecture

Diagram 2.5 represents an example (G)MPLS network architecture. The two boxes represent application control functions; the box on the left could be a P-CSCF (other CSCF are not represented). Two routers are represented below those boxes. Links represent signaling at the application control layer using SIP and at the network control layer using RSVP-TE for instance. The box labeled PDF at the entrance of the network is a policy decision function. It's the entity that decides whether or not to authorize a user to reserve network resources, according to various policies. Policies will be enforced in a policy enforcement point: the router on the left in the example.

Figure 2.6: Resource consumption scenario



User authentication

The scenario begins when resources on the right hand side host of the network register as potential processing resources that can be reached at multimediaprocess@serviceY.com. The first action of our consumer is to authenticate itself on the network, to do so, it sends a SIP REGISTER message to the network. The application control plane handles this message and the operator can authenticate the user. It is the authentication process described in diagram 2.6. The process used here for users and resources is the same as that used for IMS users. Like in IMS, it could be controlled by CSCF and the Home Subscriber Server.

Resource selection

When the user wants to submit their tasks on the Grid, an application on its behalf starts a SIP session. It sends a SIP INVITE message to multimediaprocess@serviceY.com with the description of the task. This message is forwarded from SIP proxies to other SIP proxies until it reaches the enhanced registrar. The registrar knows the location of hosts that can provide this service. It is the enhanced location service box represented in diagram 2.5. It has to select which hosts will receive the SIP INVITE message. The selection takes into account: user's preferences, hosts performance criteria such as average number of instructions per seconds and current load, current network traffic, network QoS criteria, such as available bandwidth or average delay between the user and potential destination hosts, network routes availability and operator's and resource provider's serviceY.com policies.

This enhanced location service is provided by the network operator, this element brings the added value of the network operator.

It is proposed to adapt the basic SIP forking mechanism implemented in registrars: from "ring all device that share the same address", to a more intelligent resource selection, such as "ring devices that share the same destination address and meets given policies and optimization objectives". For instance a service provider willing to balance load between two server farms would like the network operator to implement this policy in its enhanced registrar. Load balancing could be achieved taking into account servers current load and available bandwidth between a consumer and the two server farms.

Features negotiation

Once the destination's processing site has received the SIP INVITE message, the session participants are identified and the features negotiation process starts. It is proposed proposed to rely on and eventually extend various SIP mechanisms to create a generalized feature negotiation process. These basics mechanisms are the offer/answer [129] model for a multimedia session described by SDP and also those described in [131, 130]. For instance the offer/answer model is used in IMS networks for codec negotiation. QoS preconditions and this offer/answer model could be used for processing host features negotiation, such as availability of certain libraries or executables could be checked using this mechanism.

QoS support

Once features are negotiated and capabilities verified, network resources need to be provisioned. Participant of the session need to initiate the network signaling. For instance, for the tasks sent by the user to the processing site, the user's application sends the first RSVP-TE message, this message contains a reference and a token to the ongoing SIP session. This token has been created by the PDF upon a request of the first SIP proxy during the exchange of the first SIP messages. This token represents that an authenticated user is starting a session. When the ingress router receives the signaling request to reserve resources, it asks the PDF to validate the authenticity of the token and to approve the reservation of network resources. The framework described here is similar to what was called dual signaling in chapter 1. Extensions to support advance reservation have not been described here.

Multimedia services in an IMS network require strong QoS support, as well as Grid services. However Grid services for corporate users require much more bandwidth than traditional multimedia service. Admission control functions currently being standardized for the access part of the network will need to be extended for the Core part of the network. It will be in the scope of TISPAN Release 2. Scalability of existing signaling approaches have to be further studied within the perspective of Grid services.

Per user/session charging

Multimedia services in an IMS network require a per-user and per-session specific charging. As a consequence, a common charging and billing support framework is developed for IMS services. A Grid service also shares this requirement; it could rely on all these existing mechanisms.

Supervision environment

Furthermore, Grids require a supervision environment. A monitor must receive messages (alerts, notifications, etc.) published by its monitored entities. SIP is capable [127] of satisfying this requirement.

Many developments in the IMS architecture and in SIP protocols propose the basics for many functions required in a Grid. With the current development of one converged network for all services, Grid architectures could rely on this application control plane and interoperate with it instead of proposing a complete redesign of functions to be fully compliant with an SOA approach. This proposal frees Grid application vendors and resource providers from the management of certain functions. It proposes a Grid architecture integrated to the next generation network architecture. Further work to assess performance of this proposed architecture still needs to be performed.

2.5.3 IMS extensions conclusion

A comparative study of Grid network architecture with the prospect to deploy Grid services as utility services over the Internet was performed. It was shown that the SIP protocol and the IMS architecture provide the basics for important Grid functions: user authentication, authorization, per user/session billing, feature negotiation, QoS support and resource selection. The integration of these functions with the network architecture would generate value for network operators, Grid users and Grid vendors. The NGN IMS architecture can be relied upon to deliver Grid services, although it might require some modifications to existing protocols and architectures. It will benefit from next generation network deployments in the next few years and increase the possibility of seeing a Grid utility service becoming readily available.

2.6 Conclusion

This chapter has presented three main approaches to support an integrated computing service based on the web services, IMS and GMPLS. It also presented time extensions

to the network control plane to distribute network planning. The web service approach is the most commonly used today. It has already been implemented in several projects (see chapter 1). The GMPLS approach does not seem realistic as too many modification to the existing control plane needs to be made and alternatives already exists. The IMS approach could be interesting if IMS deployments become more commonplace.

Time extensions of the control plane are seen by more people as being needed. PCE time extensions are realistic but routing time extension's scalability issues have to be analyzed and solved.

My architectural contribution was to propose the IMS based architecture for Grids, the PCE and OSPF-TE time extensions and the SIP extensions containing network information.

The next chapter (3) will focus on the protocol between the Grid scheduler and local schedulers and the network manager for co-allocation. It focuses on the web service approach as it will be probably the most deployed architecture in the future. It describes the negotiation process and proposes an extension to the existing standard WS-Agreement for negotiation and analyses the performance of the SLA negotiation and creation protocol.

Chapter 4 studies the cross optimization algorithm. This algorithm and the results of this chapter is independent of the future architecture.

Chapter 3

Protocols

In previous chapters, different high-level architectures to provide Grid services have been discussed. The following chapters will describe in more detail the architecture and performance of the resource selection process. The two main topics that will be discussed are: 1) the performance evaluation of protocols(this chapter) used by the Grid scheduler to negotiate and select the best resources 2) the algorithms and performance seen by the end-user (chapter 4). Section 3.1 is a qualitative comparison of the different "negotiation" protocols. The detailed mechanism used by VIOLA's Grid scheduler will be described in section 3.2 in order to evaluate and compare VIOLA's current Grid scheduler performance against potential improvements. Section 3.3 describes the model used and section 3.4 gives the main results.

The Grid scheduler is the component that has to negotiate, select and schedule resources in order to execute a user's job and fulfil its requirements. As can be seen, co-ordinating the access to multiple resources at the same time requires specific protocol features that negotiation and agreement protocols do not necessarily have. Schopf [136] already described the different actions to be performed for job scheduling. Diagram 3.1 describes the different steps performed by a Grid scheduler to co-allocate resources: Resource filtering, SLA negotiation, SLA creation, Job submission.

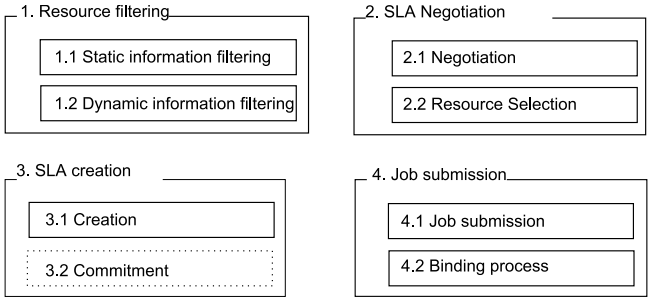


Figure 3.1: Resource selection & reservation

To run a job that requires several resources, like networking and computational resources, managed by different resource management systems (RMS), several steps must be performed by a Grid scheduler. Upon receipt of the job request, the scheduler starts the first phase: resource filtering based on static information and dynamic information. Static information does not change over time: number of CPUs, operating system, location, etc. Dynamic information changes over time: availability, load, etc. The second phase is the negotiation process and results in the selection of resources that can satisfy the job request. The third phase is the SLA creation phase concluded by the commitment of all service providers (or local RMS) involved leading to a reservation of the negotiated resources as described in the SLA. The last phase is the job submission followed by the execution. Monitoring tasks and reporting to the end user are two of the functions that could be transferred to another component, like job execution and/or monitoring service. Once a job has been submitted to the Grid scheduler and all resource reservation have been made, the Grid scheduler can transfer the job to an execution and monitoring service (EMS). This service then sends back to the end-user enough information to communicate with the EMS. The EMS will be in charge of the job execution, eventual cancellation and monitoring on behalf of the end-user. Said differently, the Grid scheduler finds and schedule access to resources for each job and the EMS controls the jobs on those resources.

3.1 SLA Negotiation, SLA creation and commit protocols

Negotiation is a widely studied topic and there are numerous publications addressing different aspects, e.g. [139] is a general purpose negotiation journal, [50] is a survey about negotiation in distributed resource management systems, while [96] and [95] discuss aspects of service negotiation in the Grid. In this context and in the simplest case, a user's job has to be executed and the Grid scheduler has to select between different target systems. If all systems are identical and only one parameter influences the selection, i.e. price, this case is similar to a typical business negotiation between one buyer and several sellers. An auctioning mechanism like the ones described in [51] can be used. The point of view of an end user is adopted, but from a resource provider's point of view, several jobs compete for one resource, i.e. several buyers and one seller. From the scheduler's point of view, many jobs compete for several resources, i.e. many buyers and many sellers. Buyya [51] (page 36) also surveyed several distributed resource management systems based on price.

3.1.1 Price consideration

"**English Auction (first-price open cry)** all bidders are free to increase their bids exceeding other offers. When none of the bidders are willing to raise the price anymore, the auction ends and the highest bidder wins the item at the price of his bid. [...]

First-price sealed-bid auction each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of his bid. In this case a bidder's strategy is a function of the private value and the prior beliefs of other bidders' valuations. The best strategy is to bid less than its true valuation and it might still win the bid, but it all depends on what the others bid.

Vickrey (Second-price sealed-bid) auction each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of the second highest bidder. [...]

Dutch Auction the auctioneer starts with a high bid/price and continuously lowers the price until one of the bidders takes the item at the current price. It is similar to a first-price sealed-bid auction, because in both cases the bid matters only if it is the highest, and no relevant information is revealed during the auction process. From the broker's bidding strategic point of view, a Dutch auction is similar to an English (first-price sealed-bid auction). The key difference between them is that in an English auction bid starts with a low opening price and increases progressively until demand falls, whereas in a Dutch auction bids start with a high opening price and decrease progressively until demand rises to match supply. The interaction protocols for a Dutch auction are as follows: the auction attempts to find the market price for goods and services by starting at a price much higher than the expected market value, then progressively reducing the price until one of the buyers accepts the price. The rate of reduction in price is up to the auctioneer and they may have a reserve price not to go below. If the auction reduces the price to the reserve price with no buyers, the auction terminates. In terms of real time, a Dutch auction is much more efficient as the auctioneer can decrease the price at a strategic rate and the first higher bidder wins. In an Internet wide auction, it is appealing in terms of automating the process wherein all parties can define their strategies for agents that can participate in multiple auctions, to optimize their objective functions.

Double Auction This is one of the most common exchange institutions in the marketplace, whose roots go back to ancient Egypt and Mesopotamia [126]. In fact, it is the primary economic model for trading of equities, commodities and derivatives in stock markets (e.g., NASDAQ). In the double auction model, buy orders (bids) and sell orders (asks) may be submitted at anytime during the trading period. If at any time there are open bids and asks that match or are compatible in terms of price and requirements (e.g., quantity of goods or shares), a trade is executed immediately. In this auction, orders are ranked highest to lowest to generate demand and supply profiles. From the profiles, the maximum quantity exchanged can be determined by matching asks (starting with lowest price and moving up) with demand bids (starting with highest price and moving down). Researchers have developed software-based agent mechanisms to automate a double auction for stock trading with or without human interaction [60]. The double auction model has the highest potential benefit for Grid computing. [...]

All of the above auctions differ in terms of whether they are performed as open or closed auctions and the offer price for the highest bidder. In open auctions, bidding agents can know the bid value of other agents and will have an opportunity to offer competitive bids. In closed auctions, the participants' bids are not disclosed to others. Auctions can suffer from collusion (if bidders coordinate their bid prices so that the bids stay artificially low), deceptive auctioneers in the case of a Vickrey auction (auctioneer may overstate the second highest bid to the highest bidder unless that bidder can vary it), deceptive bidders, counter speculation, etc."

However in the current state of development of Grid systems, price is not the only differentiating factor. Just like for financial products [87], before one could trade derivatives on an open exchange market, computational products and storage products have to be well defined and standardized. For instance, a future contract on corn specifies the type of product, its quality, the delivery date, delivery methods, payment arrangements, etc. Products or services negotiated on an open market must be very well defined. Actually, what is being exchanged are contracts and not just a product. To apply such mechanisms to Grids, a pre-requisite would be to standardize Service Level Agreements and their terms. In this way, resource owners would not sell a product or a service on the market, but an SLA. However, the market is not mature enough for such approaches to emerge, in the meantime automatic SLA negotiation systems automate the process, while price and real automated business transactions are left behind.

3.1.2 Automated Negotiation

Automatic negotiation of SLAs is a complex and time consuming process [88, 140, 80], when even two users have to find an agreement on multiple criteria. Imagine how difficult the problem becomes when multiple entities have to reach an agreement [58]. When at least two resources are needed at the same time to run a job, e.g. a network connection and a processing resource, several steps have to be performed before reaching an agreement between the resource providers and the consumer. Green [80] cites two main frameworks for automatic negotiation: ontologies and web services. He states that automated negotiation has three main considerations: The negotiation protocol, the negotiation objects and the decision-making models. He considers two current options in order to achieve this type of negotiation. One option is for the originating agent to negotiate separately with each Autonomous System (AS) along each potential path to ensure that an end-to-end path is available. The dominant choice however, is to use a cascaded approach where each AS is responsible for the entire path downstream of itself. This approach enhances agent autonomy as it is only responsible for its immediate links. The autonomy of the cascaded approach struggles however with the issue of price. In a cascading scenario an intelligent agent would need to know the utility functions of all the downstream domains if the best price combination is to be determined, which is private information. In contrast, in this chapter the scope was limited to protocols that permit the negotiation of agreements between two parties based on WS-Agreement [41] rather than tackling the full complexity

of automated negotiation. These bi-lateral agreements might then be combined into one single agreement

3.1.3 Commit protocols for distributed databases

Distributed transactional systems have been widely studied. One of their objectives is to propagate a consistent state across several systems, in a way that at any time all systems can show a consistent state to users. The consistent state or consistent view maintains and propagates between systems a logical coherent state. To provide crash recovery, several operations are logically grouped into transactions. Those transactions permit the change from one consistent view to another. For instance, you do not credit a bank account if you have not debited another bank account. However, these are two independent operations. A bank's distributed database system must group these two operations in one transaction. Thus it permits the change from one consistent state "before the transfer" to another "after the transfer". Database state changes are visible by other users once a transaction is committed to the system. In distributed systems, each transaction can impact several different systems not co-located. Thus distributed database experts have developed commit protocols [47, 92, 110]. As Skeen described in [142], "The processing of a single transaction is viewed as follows. At some time during its execution, a commit point is reached where the site decides to commit or to abort the transaction. A commit is an unconditional guarantee to execute the transaction to completion, even in the event of multiple failures. Similarly, an abort is an unconditional guarantee to "back out" of the transaction so that none of its results persist. If a failure occurs before the commit point is reached, then immediately upon recovering the site will abort the transaction. Commit and abort are irreversible."

When a user needs to make a change in a distributed database, a coordinator will propagate this change on all systems. As Skeen explains, upon receipt of a change request the coordinator forwards it to all distributed systems. Upon the change request receipt, all slaves go to the wait state. Then they can decide whether or not to accept this change, and send their response. The coordinator collects all responses to the change request, if one of them is negative, it goes in the abort state and sends an "abort" to all systems, if all responses are positive, then the coordinator goes in the commit state and sends a "Commit" to all systems. Upon receiving a "Commit"/"Abort" all systems must commit/ abort the change request. Diagram 3.2 (left) represents a slave's two phase commit protocol finite state machine (FSM). This process is the two phase commit process, supported by a two phase commit protocol.

The problem with this process is in the case of a system failure. It's impossible to know whether the transaction was committed or aborted. The wait state leads to both commit and abort state. For instance, when the coordinator fails after having sent a "commit" to some slaves but not all, the remaining slaves can not know whether the transaction should be aborted or cancelled. The two phase commit protocol is an example of a

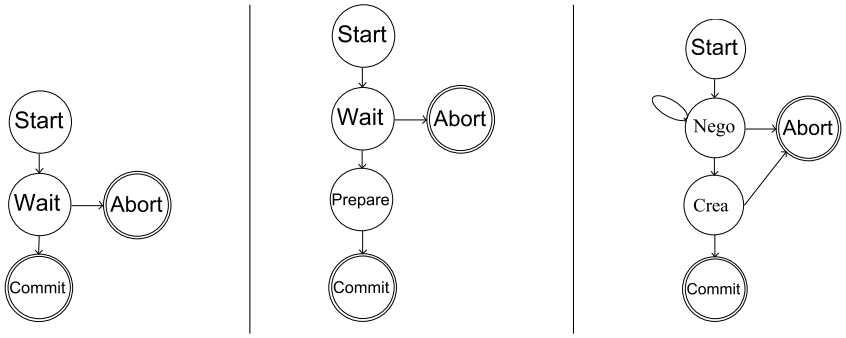


Figure 3.2: Two phase commit slave's FSM (left), three phase commit slave's FSM (middle), and SLA negotiation and creation resource provider's FSM (right)

blocking protocol.

To provide crash recovery, and avoid blocking problems, Skeen introduced a three phase commit protocol. He added an intermediary state before the commitment as shown in Diagram 3.2 (middle). This state corresponds to a prepare to commit. It's impossible to jump from this state to an abort state. He proved that if a state transition was possible between the prepare and the abort state, the protocol would be blocking. As a consequence, from any state on the slave's finite state machine it is possible to determine whether the transaction should be committed or aborted in case of failure. In case of failure a slave in the "Wait" state must abort, while a slave in the "Prepare" state must commit.

3.1.4 Commit protocols for distributed resource management systems

In an environment with distributed RMS providing guarantees on resource usage, a Grid scheduler may create SLAs with its users. In a co-allocation use case, this SLA takes into account several resources coming from several resource providers. With each independent resource provider a bilateral SLA has to be negotiated and created. A Grid scheduler has to create these bilateral SLAs on behalf of its users. For instance, in VIOLA, users may request network and computational resources with a dedicated QoS. The Grid scheduler has to orchestrate the individual reservation of network and computational resources. These two reservations are realised as two bilateral SLAs.

The essence of a distributed databases' commit protocol is the transaction: a group of individual operations, logically linked. In a distributed resource management system, co-allocation requires multiple bilateral SLAs. For a user or a particular service requiring multiple resources, either all of the individual bilateral SLA must be created, or none. The user SLA creation process is a transaction composed of multiple bilateral SLA creation.

Before reaching an agreement, two steps must be performed: negotiation and creation. The negotiation process can involve all resource providers. Its results are input to a resource provider selection process. When two resources are needed, e.g. network and computing, even if the negotiation involves many compute resource providers, only one computational resource will be selected. For many resources offered, the negotiation process does not lead to an SLA creation process. This is the main reason why negotiation must neither obligate the provider nor the consumer of the SLA. However, the SLA negotiation and creation process should minimize the number of discarded agreement creation requests when it has been previously negotiated. This should occur only when there is a race condition: when two or more users are competing simultaneously for the same resource at the same time. The separation of agreement negotiation and agreement creation process and minimising the number of discarded agreement creations after negotiation are conflicting objectives.

One way to observe atomicity of the SLA creation is to use a transaction and to rely on a two phase commit protocol. Once resources have been negotiated, the orchestrator starts the SLA creation process by sending an SLA creation request to the selected resource providers. Then each resource provider responds to the request with yes or a counter offer. If all providers agree, the orchestrator sends a commit reservation to all systems. Upon receipt of this message, the reservation is committed and the SLA created. Diagram 3.2 (right) shows this process.

When the resource provider receives an SLA negotiation offer, its state changes from "Start" to "Nego". It then answers the negotiation offer by either accepting it or making a counter offer. In the case of a counter offer, it stays in the "Nego" state. It can also abort the negotiation and proceed to the "Abort" state. Once the orchestrator decides to start the SLA creation process, upon receipt of the SLA creation request, the resource provider's state changes to the "Crea" state. It stays there if it accepts the reservation otherwise it goes to the "Abort" state. The final "Commit" state is reached when it receives a "Commit" message from the orchestrator and that resources are reserved and made unavailable to the rest of the world. As mentioned above, this simple two phase commit scenario can lead to a race condition during the SLA creation process. While the resource provider is in the "Crea" state, other users see the previous consistent state where resources are still available. To prevent this, the "Crea" state could imply "locking" resources thus providing a pre-reservation for the transaction lifetime. This prevents other users from reserving the same resource at the same time. In case of a lock request, second users' transaction must wait for the lock to be released. Although the FSMs shown in the middle and on the right-hand side of Diagram 3.2 look similar, the SLA negotiation and creation process is not a three phase commit. It is a blocking protocol as described by Skeen [142] and it does not provide any guarantees against crashes. One could still imagine a non blocking SLA creation protocol relying on a three phase commit providing crash recovery. The negotiation process has now been described in detail. Following is a description of how VIOLA works, before explaining the model used to analyse its performance.

3.2 VIOLA's signalling Architecture

Chapter 2 and 1 already described VIOLA's architecture and components. Section 3.2.1 describes VIOLA's "old" negotiation process.

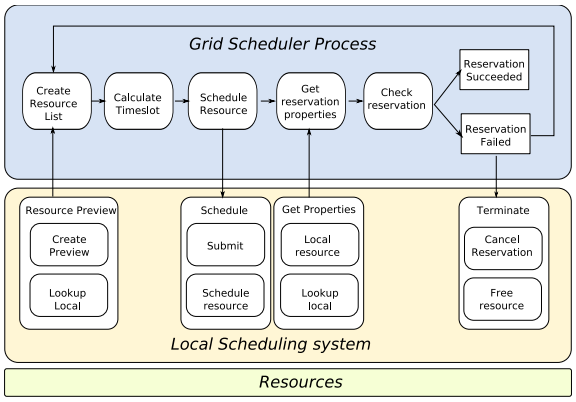


Figure 3.3: Negotiation process

3.2.1 Original VIOLA 2PNP

To achieve the co-allocation of different resources, the Grid scheduler communicates with the selected resource management systems (RMS) through a set of adapters. These adapters provide a uniform interface to the Grid scheduler and may implement missing functionalities of the RMS. At the first step of the co-allocation process the Grid scheduler queries the adapters of the selected RMS to get the earliest time the requested resources will be available. This time possibly has to be after an offset specified by the user. The adapters acquire a preview of the local resource availability from the individual scheduling systems. Such a preview comprises a list of time frames during which the requested resources (e.g. a fixed number of nodes) can be provided. It is possible that the preview contains only one entry or even zero entries if the resource is fully booked within the preview's time frame. Based on the preview the adapter calculates the next possible start-time.

These start times are sent back to the Grid scheduler. If the individual start times match, the Grid scheduler will try to reserve the resources at the computed start time via the adapters, making use of the advance reservation capability of the local RMS. If the individual start times do not match, the Grid scheduler uses the latest possible start time indicated by the RMS as start time for the next scheduling iteration. The process is repeated until a common time frame is found or the end of the preview period for at least one of the RMS is reached. The latter case generates an error condition. In case the Grid scheduler was able to find a common timeslot and reserve the resources, it later checks the scheduled start times of each reservation. This step is necessary because after

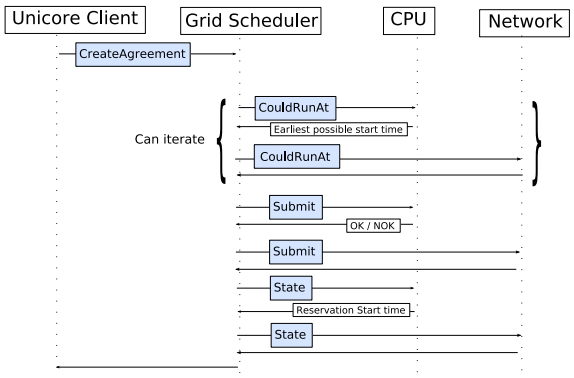


Figure 3.4: 2 Phase Negotiation process

negotiating the common start time, other reservations may be submitted by other users or processes to the local RMS, preventing the scheduling of the reservation at the requested time. If the Grid scheduler detects one or more reservations that are not scheduled at the requested time, all reservations will be cancelled. The latest effective start time of all reservations will be used as the earliest start time for the co-allocation attempt and the Grid scheduler will try again to negotiate of a common timeslot as described.

3.2.2 Original WS-Agreement

Resource negotiation and reservation is the scope of the Grid Resource Allocation and Agreement Protocol (GRAAP) working group at the OGF. The group was created in 2002. Initially this working group planned to develop one protocol for the negotiation and reservation of resources. This protocol was developed to be used for communication between a Grid scheduler and other resources, computing and network. Building one protocol for negotiation and reservation proved too difficult, so negotiation features were removed in 2004 from the scope of the WS-Agreement protocol.

In order to co-allocate different types of resources and/or resources from different domains, a Grid scheduler has to negotiate SLAs for the required resources. The easiest way of SLA negotiation is a one step process, where the context, subject and constraints of the negotiation problem are defined. The WS-Agreement protocol natively supports this kind of negotiation by the getResourceProperties method. This method returns a set of agreement templates representing acceptable agreement offers for an agreement provider. These agreement templates only provide hints on agreement offers which might be accepted by an agreement provider. They do not guarantee the agreement will be accepted. An agreement template defines one ore more services that are specified by their *Service Description Terms* (SDT), their *Service Property Terms* (SPT) and their *Guarantee Terms* (GT). Additionally an agreement provider can constrain the possible values within the SDTs, SPTs and GTs by defining appropriate creation constraints within the templates.

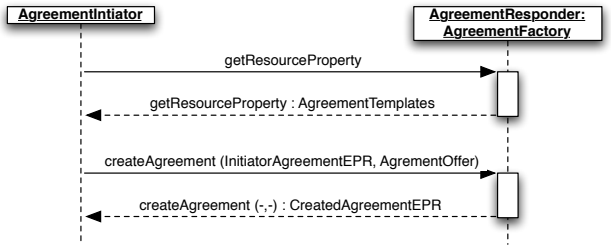


Figure 3.5: WS-Agreement one step negotiation

The creation constraints in an agreement template can be static or dynamic. Typical examples of a static creation constraints are the minimum and maximum numbers of CPU, nodes or memory. As these are properties of computing systems that are not likely to change frequently agreement templates that only contain static information usually are not restricted in their lifetime. Agreement templates can also contain more dynamic information. Such dynamic information can be used to e.g. restrict the guaranteed execution time of a given service based on the current resource availability. Since the availability of resources is likely to change frequently, templates that contain such dynamic components have a short lifetime. A Grid scheduler can use these dynamic templates to efficiently find suitable time slots in order to e.g. co-allocate resources. However, it is not always desired to expose availability information, or sometimes it is even not possible to do this in a convenient way. A typical example for this is the creation of an SLA in the network domain. Here, it is simply not possible to include the availability information for all possible network paths in a domain within one single SLA template. This would make the templates far to complex and therefore practically unusable. Therefore, the efficient agreement on time constraints in SLAs in only one phase is simply not feasible in this case. More advanced multi-step negotiations are needed to solve this problem.

3.2.3 WS-Agreement 3PNP

Due to the lack of native support for negotiation in WS-Agreement, Wäldrich and Ziegler proposed a simple three phase negotiation protocol based on WS-Agreement [154]. In order to negotiate co-allocation of resources effectively and efficiently, a Grid scheduler must be able to (i) identify a common timeslot, and (ii) to allocate resources at a specified time. The identification of a common timeslot can already be done by using WS-Agreement templates for publishing the availability of local resources. Furthermore, it seems to be feasible that agreements can be created while specifying a start time for a resource usage within an agreement offer. However, since WS-Agreement only allows creating agreements within one step (createAgreement), one can not be sure whether the creation of an agreement on multiple sites succeeds or not. Additionally, penalties may be associated with an agreement, preventing a broker to simply terminate all agreements and re-negotiate, if one site fails to create an agreement at the proper time.

To overcome these problems, the creation of different types of agreements within a negotiation process was proposed. These types are a Declaration of Intention Agreement, a Preparation Agreement and a Commitment Agreement. All of these agreements are normal WS-Agreements as specified in [41], following a certain naming convention. Subsequent agreements (Declaration of Intention Agreement, Preparation Agreement; Commitment Agreement) reference the prior created agreement using wsag:Service Reference, which contains an EndpointReference (sort of URL) of the related agreement. The negotiation process is described in diagram 3.6.

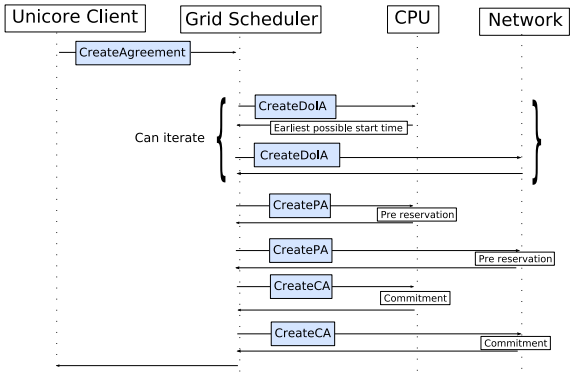


Figure 3.6: 3 Phase Negotiation process

Declaration of Intention (DoI Agreement)

A Declaration of Intention (DoI) indicates whether a service provider is able to accept the agreement offer or not. This declares at least that an agreement responder is willing to provide resources according to an agreement offer. The decision, whether to accept a DoI offer or not, depends e.g. on the requested resources (can the requested resources be provided), the requested QoS levels (can the requested guarantees be fulfilled), and the requested start time (are the resource available at the requested time). If an agreement responder is able to deliver the specified service with the appropriate QoS, but not at the specified time, the agreement responder may accept the DoI offer with a changed start time (normally the next time after the requested the service could be provided). An agreement responder can query the DoI agreement and check, whether it fulfills his requirements or not. This enables an agreement initiator to negotiate a common timeslot for a resource usage as described in [153]. Since a DoI Agreement only declares the general intention to provide resources at a specified time, normally no costs and penalties will be associated with a DoI-Agreement. Therefore an agreement initiator may terminate a DoI agreement without any penalties.

Preparation Agreement

The main purpose of a Preparation Agreement (PA) is to provide a pre-reservation of resources (with defined QoS) for a specified time. This pre-reservation has a defined lifetime (pre-reservation duration), in which an agreement initiator can choose whether the specified resources will be used or not. If an agreement initiator chooses to use the pre-reserved resources, a subsequent Commit Agreement has to be created that references the PA using the wsag:ServiceReference. If no subsequent Commit Agreement was created within the pre-reservation duration, the PA expires and the reservation is deleted, in order to have the resources freed. After a PA has expired, the agreement initiator can not reference it anymore to create a subsequent Commit Agreement. Since resources will be reserved when establishing a PA (at least for a defined time), this agreement may or may not be associated with any costs or penalties, in case of canceling the PA. This depends on the local charging model, which should be published in the Agreement Template.

Commit Agreement

A Commit Agreement (CA) is a final outcome of a negotiation process. A CA references a PA in order to indicate that an agreement initiator definitely wants to use the resources reserved with the PA. This is the final contract between the agreement initiator and the agreement responder. On acceptance of a CA, all of the defined costs, penalties and reward defined in the agreement become effective.

3.2.4 Negotiation of Agreement Templates

Negotiation requires an iterative process between the parties involved. To rely on WS-Agreement and minimize the extensions to the proposed standard, it was suggested not to directly negotiate SLAs but instead negotiate SLA templates that will then be used to create an SLA. Here, the focus is on the bilateral negotiation of agreement templates.

In the following scenario how an agreement initiator (e.g. the Grid scheduler) negotiates agreement templates with two agreement provider (e.g. a network scheduler and a CPU scheduler) will be described. A simple offer/counter offer model is proposed. In order to use this model in the WS-Agreement protocol, a new function *negotiateTemplate* is used. This function takes one template as input (offer) and returns zero or more templates (counter offer). The negotiation itself is an iterative process. The following scenario describes a simple negotiation process. During the negotiation process, the agreement initiator (e.g. a Grid scheduler) is called 'negotiation initiator'. Accordingly the agreement providers (e.g. the resource providers) are refered as 'negotiation responders'.

1. *Initialization of the negotiation process*

First, the negotiation initiator initialises the process by querying a set of SLA templates from agreement providers. To do so, it sends a standard WS-Agreement message, getResourceProperty request, to agreement providers (not shown in Diagram 3.7). From this templates, the initiator chooses the most suitable one as a starting point for the negotiation process. This template defines the context of the subsequent iterations. All subsequent offers must refer to this agreement template. This is required in order to enable an agreement provider to validate the creation constraints of the original template during the negotiation process, and therefore the validity of an offer.

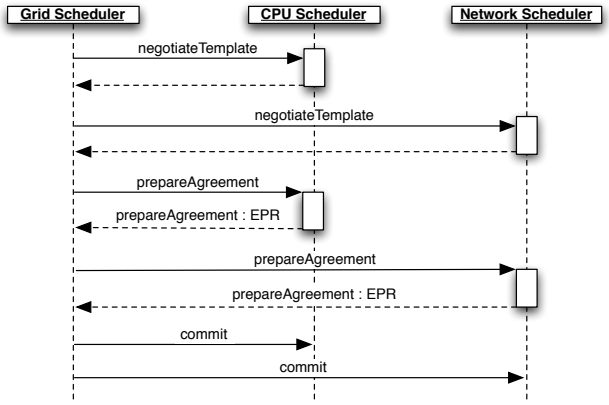


Figure 3.7: Extended WS-Agreement SLA negotiation

2. *Negotiation of the template*

   After the negotiation initiator has chosen an agreement template, it will create a new agreement template based on the chosen one. The newly created template must contain a reference to the originating template within its context. Furthermore, the agreement initiator may adjust the content of the created template, namely the content of the service description terms, the service property terms and the guarantee terms. These changes must be done according to the creation constraints defined in the original template. Additionally, the negotiation initiator may also include creation constraints within the new created template. These constraints provide hints for the negotiation responder, within which limits the negotiation initiator is willing to create an agreement. For instance, the initial CPU scheduler template can contain "any number of 2GHz x586 CPU between 5pm to 6pm". And the initiator can request "at least 5 1GHz x586 CPU anytime". After the initiator created the new agreement template according to its requirements, the template is sent to responders via a *negotiateTemplate* message (as shown in Diagram 3.7)

   When a responder has received a *negotiateTemplate* nessage, it must first check the validity of the input document (refined template). This step includes (i) retrieving the original agreement template that was used to create the input document, (ii) validating the structure of the input document with respect to the originating template, and (iii) validating the changes of the content in the input document with

respect to the creation constraints defined in the originating template.

Once this is done, the agreement provider now checks whether the service defined in the request could be provided or not. In our example, it's only then that the CPU scheduler decides that 5 1GHz x586 CPUs can be provided. If the service can be provided, it just returns the agreement template to the client, indicating that an offer based on that template will potentially be accepted. Otherwise, the provider employs some strategy to create reasonable counter offers. During this process the agreement provider should take into account the constraints of the negotiation initiator. Counter offers are basically a set of new agreement templates that base on the template received from the negotiation initiator. The relationship between dynamic created templates and original ones must be reflected by updating the context of the new templates accordingly. After creating the counter offers the provider sends them back to the negotiation initiator (*negotiateTemplate* response).

3. *Post-processing of the templates*

   After the negotiation initiator received the counter offers from the negotiation responder, it checks whether one or more meets its requirements. If there is no such template, the initiator can either stop the negotiation process, or start again from step 1. If there is an applicable template, the initiator validates whether there is need for an additional negotiation step or not. If yes, the initiator uses the selected template and proceeds with step 2, otherwise the selected template is used to create a new SLA.

3.2.5 SLA creation

After the negotiation of an agreement template acceptable for both parties, the initiator needs to create the agreement. At this point, a problem similar to the transaction problem of distributed database systems arises. The goal of a Grid scheduler is to create a set of SLAs with different resource providers in order to provide co-allocation. Therefore, the scheduler first negotiates a set of templates with the providers, which identify the possible provisioning times of the required resources. However, it must not be forgotten that templates only provide hints of what SLAs an agreement provider might accept. There is no guarantee associated with a template. This means that a strategy to create all SLAs or none is needed. In principle there are two major strategies to achieve this:

1. *to use transactions to create the SLAs, or*
2. *to create each SLA within one step, applying policies to the SLA.*

The usage of transaction mechanisms to create distributed SLAs, namely the usage of the two phase commit protocol, was already discussed in this chapter. Since there is no support for a two phase commit in today's WS-Agreement requires a extension

to the standard to address this problem. This process has been started recently in the OGF working group that created WS-Agreement. A solution exists by adding a type of agreement that must be created in two phases: the first phase is a creation of the agreement triggered by a new *prepareAgreement* message and the second with a new non-standard *Commit* message as shown in diagram 3.7.

The other approach is to create an SLA in one step using today's WS-Agreement functionalities, cancellation mechanisms and incentives. However, it is not very obvious how the co-allocation problem can be solved with this approach. In order to achieve this, the content of an SLA must be investigated. On one hand, an SLA describes the service and its properties. On the other hand, it specifies the guarantees for a specific service. In a co-allocation scenario, where a Grid scheduler uses SLAs to coordinate e.g. network and computational resources, it employs execution guarantees in order to assure that the different services are provided at the same time. These guarantees may also include costs that are associated with the service if it is provided successfully, as well as penalties that arise when a guarantee is violated. However, an SLA might be prematurely terminated by the agreement initiator, before the service is actually provided. In fact, this is a cancellation of an SLA. When a service provider guarantees a certain execution time for a service, this normally comprises resource reservations. Therefore, the resource provider wants to prevent the termination of an existing SLA. This can be achieved by including a basic payment within the SLA. The basic payment is potentially a very small amount of money that is even charged if the SLA is terminated by the agreement initiator before the service was actually provided. It is therefore a termination penalty and represents the costs for the overhead produced by the resource reservation. In order to enable the Grid scheduler to efficiently negotiate and create SLAs, there could be a certain time period in which the SLA can be terminated without penalty. The duration of this period can dynamically be specified during the negotiation process. The Agreement provider could use a certain trust index in order to determine the maximum length of this period. For example, such a trust index could be computed by the ratio of successful created agreements and prematurely terminated agreements. This offers a feasible solution for the orchestration of multiple resources using the current one-step SLA creation of WS-Agreement.

3.3 Model description

One of the driving focuses of the VIOLA architecture is that it's a good representation of the current state of the art of Grid systems interacting with network resources. VIOLA uses web services that are often criticized for their poor performance. It is often said that XML processing introduces a big latency in the control system and that text-based representation of information is a waste of memory and bandwidth. One of the reasons for this analysis is to be able to quantify latency introduced by the Grid scheduler and its memory requirements. The comparison of the performance of a three phase negotiation

protocol with a two phase was required.

To conduct these analyses an analytical approach based on queuing network's theory was adopted. The model is similar from the one developed by Gurbani [84] for evaluating a SIP proxy's performance. The difference is that a Grid scheduler's performance will be evaluated here. These analyses have been conducted in collaboration with Alejandro Gaspar. The remaining sections will present the model used and analytical results.

Four models were built, two for the two phase negotiation protocol, a sequential and a parallel and two for the three phase negotiation protocol, again a sequential and a parallel.

Sequential 2PNP This model represents a sequential implementation of a two phase negotiation protocol. The term sequential refers here to the programming framework used to communicate with remote entities. In a sequential implementation, after having sent a message to a remote entity, the Grid scheduler waits to receive an answer from that remote entity before sending another message, or performing another action.

Parallel 2PNP This model represents a parallel implementation of a two phase negotiation protocol. Compared to the sequential implementation, after having sent a first message to a remote entity, a parallel implementation does not wait to receive a response before begining to communicate with other entities or to perform the next action.

Sequential 3PNP This model represents a sequential implementation of a three phase negotiation protocol.

Parallel 3PNP This model represents a parallel implementation of a three phase negotiation protocol.

The objective of the model is to evaluate a Grid scheduler's performance taking into account the negotiation protocol and its implementation variant (sequential or parallel). Interesting properties to compute, would be:

- the average total delay introduced by the Grid scheduling process described in section 3.2.1,
- the maximum number of job requests per second a Grid scheduler can support,
- the amount of memory required by the Grid scheduler.

To do so, it is assumed that the Grid scheduler receives a job request and maintains a state in memory for each job that represent the processing state of this job request. For instance, the Grid scheduler might have received a job request, performed some actions,

sent an intermediary message to a remote entity and be waiting for a response. To model this, two queues are used: the first introduces the Grid scheduler's job request processing time, while the second queue introduces the waiting time for the response. It will be assumed that the waiting time for the response equals the remote entity's processing time of the intermediary message and generation of the response. Transfer delays will not be taken into account.

Jackson's network theory will be used. One of its limitation is that each queue's processing time distribution must be exponentially distributed for the analytical formulas to be correct, which is obviously not a realistic assumption. However, this theory produces quick and simple results. As for every Jackson network model, the input parameters of our model are: the job request arrival rate and distribution, each queue's service time distribution and average service rate, probabilities for a job request to switch from one queue to another, i.e. from one state to another. A job request needs two resources: network and computing. The stationary solution of each model, that is a solution that represents a network at "equilibrium" is computed.

The upper part of diagram 3.8 is the queuing network model for the two phase negotiation protocol described in 3.2.1. In this diagram, the following notation was used: a queue is represented with a message name M, a letter X above it, and a letter μ encircled. μ is a processing rate, i.e. here the inverse of a processing time. X refers to an entity, it can be M for the Grid scheduler (a.k.a. Meta-scheduler), LS for a local-scheduler's adapter or A for the network adapter, ARGON. Each server in this queuing network introduces the processing time taken by entity X to receive a message M, process it, generate a response and send it to the relevant entity. To be more precise, when the entity X is not a Grid scheduler, the server includes the Grid scheduler's waiting time, which equals the processing time spent by entity X, to receive M, process it, generate a response and send it to the next relevant entity. Job requests are clients of this queuing network whose arrival time follows a poisson distribution, which is a good model for independent job requests triggered by human behaviour. When a client changes queues, it models a state change of this job request inside the Grid scheduler, the duration of this state corresponds to the time spent in the corresponding queue. Some examples are listed below:

In queue 1, the job request is in the "NotCreated" state, once it has been processed, it moves directly to the second queue, and to the "Waiting for LS&A CouldRunAtResponse" state. The first server intoduces the time needed by the Grid scheduler to receive, process the "createAgreement" message and sent two "CouldRunAt" messages to ARGON and the first LS's adapter. $1/\mu_2$ is the average time taken by ARGON and the first LS's adapter to receive the "CouldRunAt" message, process it and send the response back to the Grid scheduler. It is assumed that that these two operations are done in parallel by two different entities and that only one queue is needed. The next two queues model the treatment of the two responses by the Grid scheduler and the time needed to generate the next "CouldRunAt" messages for ARGON and the second LS's adapter.

It is assumed that transition probabilities are constant. Having sent and received

"couldRunAt" messages to all adapters, q is defined as the loop probability, i.e. the probability that there is no common timeslot between ARGON and an LS after having queried all adapters once. p is the probability of never finding a common timeslot and sending back an error message to the user. m is the probability of finding a common timeslot after a loop. By deduction, $m + p + q = 1$. The loop probability q depends on many factors, notably the load on each local-scheduler. To simplify, this model assumed it is constant. This probability represents the impact the real user-load has on the Grid scheduler. It is the impact the "user plane" has on the "control plane". The following sections will refer to this probability as the *loop probability*. Since it is not computed, all results will be functions of it, hence the x-axis of all graphs.

Existence of a stationary solution gives the maximum job request arrival rate each model can acheive. In all models, the Grid scheduler must cycle if there is no common timeslot found. As a consequence, it's easy to see on all equations that under "realistic" parameter assumptions the loopback queue k_0 is always a bottleneck. In diagram 3.8, $k_0 = 2$. That is to say, its internal load ρ_{k_0} is the biggest internal load. The maximum job request is given by $\rho_{k_0} < 1$, where for the three phase parallel model

$$\rho_{k_0} = \frac{\lambda/\mu_{k_0}}{1 - (q + (m(a + cb)))}$$

The stationary solution of these models enables us to compute the mean number of jobs N in the system.

$$N = \sum_{k \in Queues} \frac{\rho_k}{1 - \rho_k}$$

Calculating the memory requirement is equivalent to calculating the mean number of jobs in the system and the total delay introduced by the Grid scheduler T is equivalent to the mean number of jobs N in the system according to Little's formula $N = \lambda T$ where λ is the job request arrival rate. Hence the problem reduces to computing the mean number of jobs and the maximum arrival rate.

3.3.1 Parameters value choice

Input service rates for each queue were taken from real measurements of the time required to process job requests. The parsing time of a JSDL (Job Submission Description Language [42]) file was measured and used for the corresponding queue's main average service time. Values ranging from 1.7ms to 4.7ms were used for the average time needed to receive an XML message, check its validity and parse it, depending on the entity who receives the message and the size of the message. These values do not take into account the time delay introduced by Apache, Tomcat, Axis and SOAP processing. This delay could be taken into account to improve absolute values obtained by the model. However, it is only of little interest since the focus is on relative performance gains or penalties.

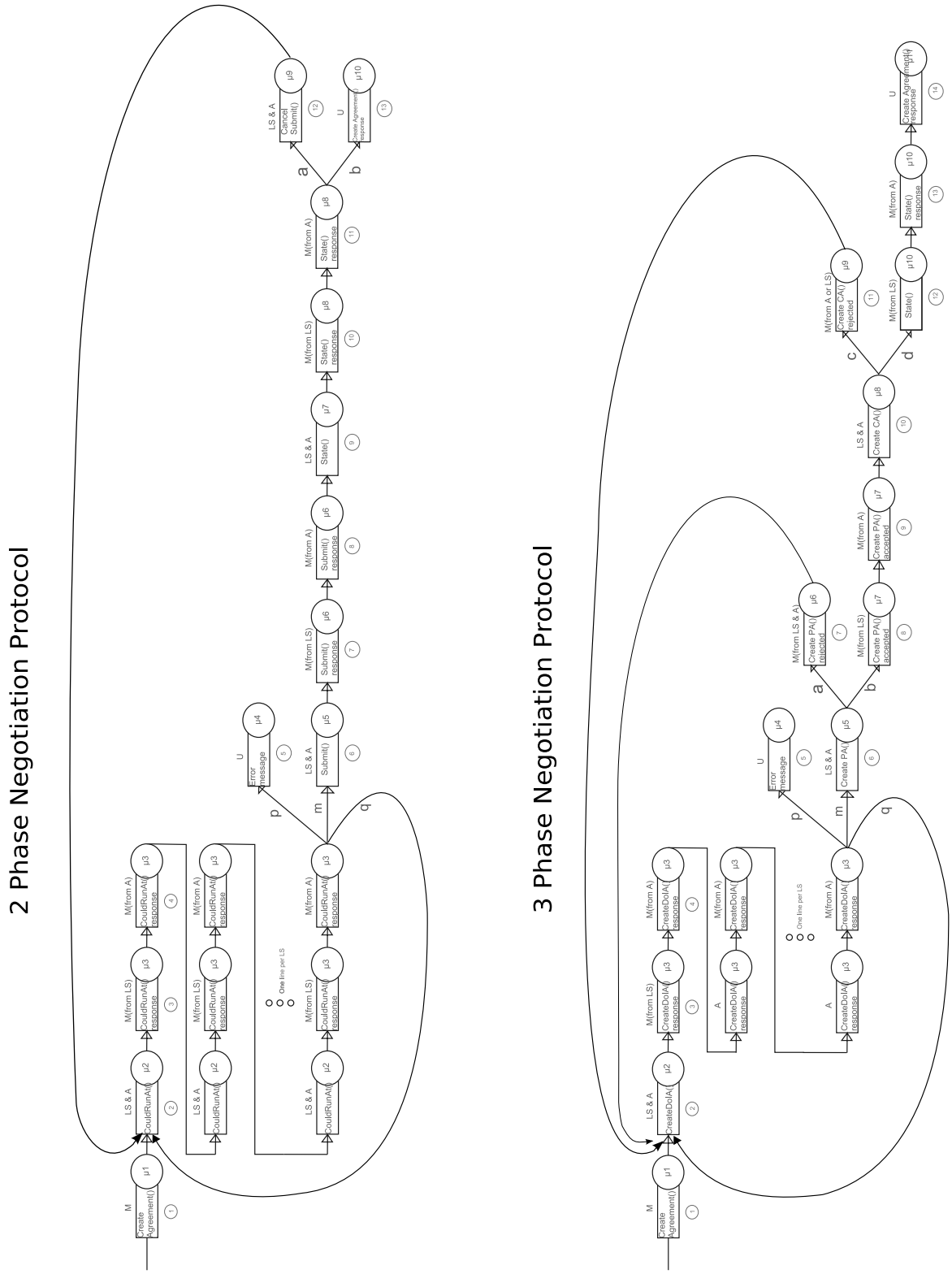


Figure 3.8: Parallel models

All four models take into account fixed probabilities. In the parallel three phase negotiation protocol model: p, q, m, a, b, c and d. m is the probability to find a common timeslot between the network and the computing resources. q is the probability of looping after having sent one "couldRunAt" or "CreateDoIA" to all adapters because there is no common timeslot. $p = 1 - m - q$ is the probability of getting an error while searching for a common timeslot, it includes the probability of never finding one. a is the probability of rejection of the Preparation Agreement by local adapters. It can happen when the resource has been reserved between the "CreateDoIA" response and the receipt of the "CreatePA". $b = 1 - a$ is the probability that the "CreatePA" response is positive. c is the probability that the Commit Agreement is rejected, for instance it can occur when the time taken by the Grid scheduler to send the commit agreement is larger than the pre-reservation lifetime, and $d = 1 - c$ is the probability the commit agreement succeeds. To generate the results, $p = a = c = 10\%$ was chosen for the parallel three phase negotiation protocol model.

The 2PNP model uses less probabilities: p, m, q, a and b. p, m, q are the same as in the parallel three phase. b is simply $1 - a$, where a is the probability that the network and CPU reserved times do not match even if the couldRunAt response said they do. To get the results $p = a = 10\%$ was chosen for the parallel 2PNP model. Doing so, the three phase protocol was penalised because of the 10% commit agreement rejection rate. This assumption is a reasonable starting point to compute a worst case scenario penalty of 3PNP over 2PNP.

3.4 Results

Parallel models' performance is expected to be better than sequential models and three phase negotiation protocols to be less efficient than two phase negotiation protocols. The remaining question is: by how much?

3.4.1 Arrival Rate

The results show that it's always the same queue that is limiting the system's maximum arrival rate. It is, the queue at the entrance of the main loop (second queue on the diagrams included).

Since the probability of not finding a common timeslot q can not be really approximated, results are plotted as a function of the probability of not finding a common timeslot. Hence the x axis of our diagrams is the probability of not finding a common timeslot q. When the y axis is a gain, it's computed with $\Delta P/P$. The first observation is the order of magnitude of the maximum arrival rate supported by the Grid scheduler, between 30 jobs per second to 150 jobs per second, depending on the probability of

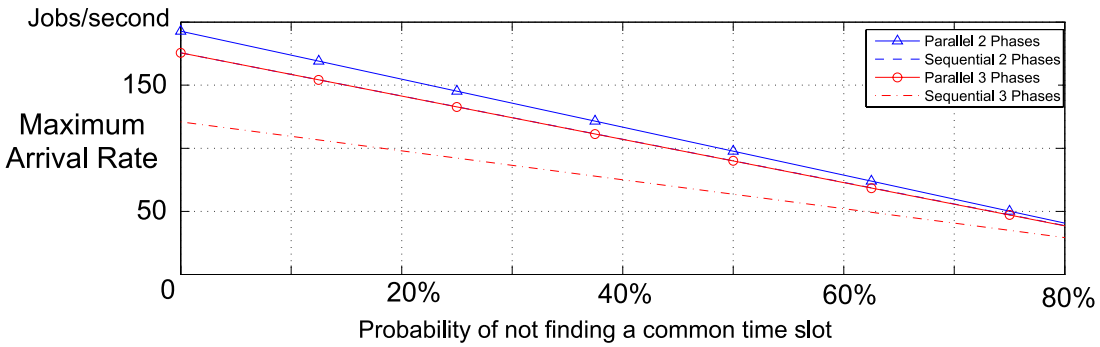


Figure 3.9: Maximum arrival Rate

not finding a common timeslot. In the worst case, the Grid scheduler supports 30 jobs per second. To better understand this diagram, a SIP proxy supports the forwarding of approximately 300 calls per second.

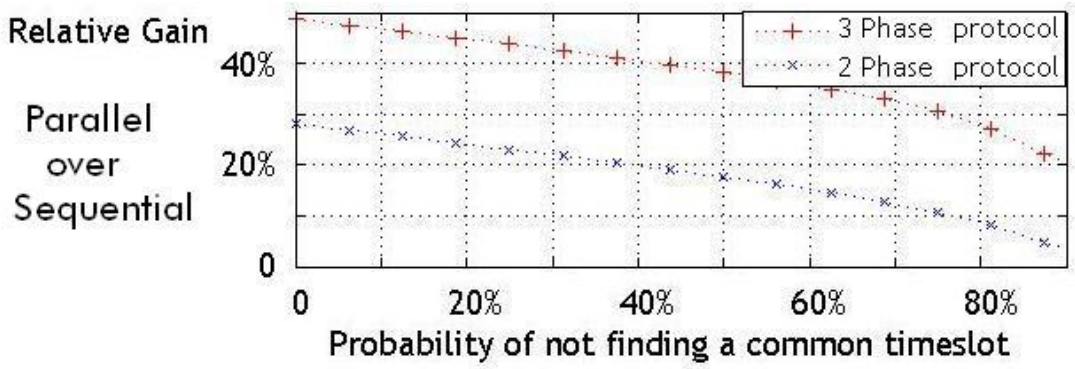


Figure 3.10: Sequential vs Parallel

A second comment is that the penalty on the maximum arrival rate of a 3PNP has over a 2PNP is small (less than 10%) for parallel models: almost all curves are supper-imposed. The penalty is bigger for sequential models. A third point, although very predictable is that a parallel implementation of the Grid scheduler is far superior than a sequential implementation. The performance gain on the maximum arrival rate varies from 20% to 50% for the three phase negotiation protocol. The gain is slightly less for the two phase negotiation protocol: from 10% to 30%.

The main conclusion from these two graphs are that the implementation should be parallel and that the maximum arrival rate penalty a 3PNP has over a 2PNP is less than 10%.

3.4.2 Mean Nb of Jobs

The mean number of jobs in the Grid scheduler for the four different models is computed by fixing a common arrival rate. The next diagram shows that when the arrival rate is set close to the maximum: 90% of the max arrival rate when $q = 50\%$, $\lambda = 63$job/s. A big difference in the mean number of jobs N in the Grid scheduler is observed between the four models. The job request processing delay is proportional to the mean number of jobs $Delay = N/\lambda$ and the memory requirement is the product of N by the memory footprint of one job request.

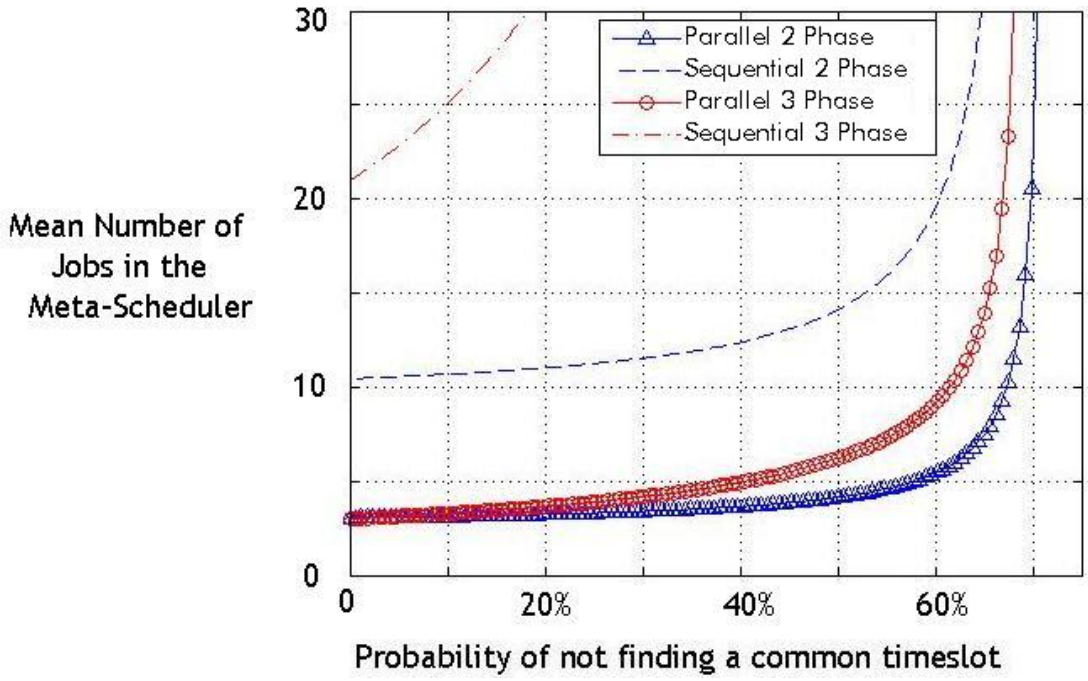


Figure 3.11: Mean Number of Jobs

The first result is that, when the arrival rate is fixed, the probability of not finding a common timeslot greatly influence the mean number of jobs in the system. Above a certain threshold, the system becomes unstable. There is a clear need to control the "loop" probability to provide performance guarantees like a boundary in the job request processing delay. A second result is that the 3PNP penalty is not that large.

Diagram 3.11 shows that for a fixed arrival rate $\lambda = 63$job/s, the penalty varies from 10% to 100%, which represent a factor two in terms of memory requirement or total processing delays.

The sensitivity of this penalty to the arrival rate was computed. Diagram 3.12 shows curves representing the penalty function of the "loop" probability for different values of the job request arrival rate (in Job/s): $\lambda \in \{5, 8, 12, 15, 19, 32, 44, 57\}$. The result is that

the penalty difference between all these curves is less than 10% for a ten times increase in the arrival rate.

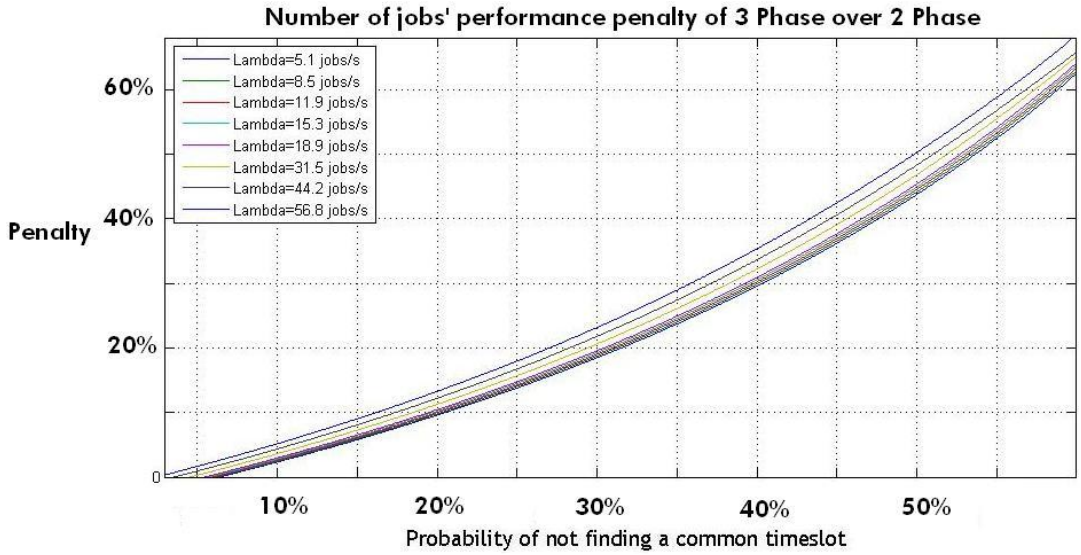


Figure 3.12: Penalty sensitivity to the arrival rate

The consequence of this result, is that for a given arrival rate, the memory requirement of the three phase negotiation protocol is roughly twice more than for the two phase. And if the arrival rate was multiplied by 10, one would only need an extra 10% of memory. However, the performance penalty on the total job request processing delay D greatly depends on the arrival rate: according to Little's formula $D = N/\lambda$, the job request processing delay is inversely proportional to the arrival rate. In the worst case, the processing delay for a 3PNP is twice that for a 2PNP.

The influence of other parameters on the mean number of jobs and the maximum arrival rate was analysed. The sensitivy to other service rate is very small whenever they vary in reasonable ranges. The impact of having over-estimated the commit agreement rejection rate is less than 10% on the mean number of jobs.

3.4.3 Conclusion

To conclude, there is a clear need to implement the Grid scheduler with parallel communication mechanisms with other adapters. The order of magnitude of the number of jobs per second a Grid scheduler can support is around 30 jobs/second. And the total job processing delay at this rate introduced is 200ms. When the Grid scheduler is implemented with parallel communications mechanisms, a three phase negotiation protocol does not penalise the maximum arrival rate supported by the system compared with a two phase negotiation protocol. However there is a clear need to control the number of loops the Grid scheduler will do for each job and reduce it as much as possible. The

Grid scheduler should be designed taking into account that on average a probability of not finding a common timeslot after one pass for each job higher than 60%, will make the Grid scheduler unstable. 3PNP memory requirement and processing delay are at worst twice bigger than for a 2PNP.

3.5 Conclusion

This chapter discussed basic functionality for resource orchestration in Grids, namely mechanisms to dynamically negotiate and create service level agreements using WS-Agreement. Inovations proposed to the WS-Agreement standard have been the object of a publication [119, 120]. SLAs are a basic building blocks for Grid resource orchestration and distributed resource management. Some WS-Agreement extensions were proposed to dynamically negotiate SLA templates: *NegotiateTemplate* was a method to support a simple offer/counter-offer model. The second relevant part of the resource orchestration process is the creation of distributed SLAs. Two different strategies to co-allocate SLAs in the Grid were proposed: one using a two phase commit with an WS-Agreement *Commit* extension and one using a single phase commit with SLA cancellation and incentives. The performance of the two phase commit approach has been evaluated. Once that resources can be efficiently co-allocated, the next chapter will investigate which information should be exchanged to provide a co-allocation. It will study resource cross optimization.

Chapter 4

Algorithms

The previous chapters show that interactions between a Grid scheduler and network management systems have increased in several Grid projects. The aim is to provide resource co-allocation, reserving processing and network resources simultaneously.

How many jobs can we allocate on the Grid? Is it necessary to go beyond a bandwidth on demand service? Should the network operator communicate more information like topology and reservable bandwidth about the network? This increased interaction would permit a Grid scheduler to better optimize resource allocation process.

This chapter briefly describes the general problem recalling Max Flow based techniques. Then the methodology developed during this thesis is exposed. A cross optimizing algorithm is proposed and evaluated. Appendix 4.6 describes an extension of this algorithm and appendix 4.6 describes in more details the tools that were built. An analytical approach gives a formula to estimate the number of jobs accepted in a network. This formula is verified and the previous questions are answered in the last section.

4.1 Introduction

One of the motivating ideas for this thesis was: if computing resources are managed with network resources, what could be done? When a user wants to execute a job, if he is interested in having the results as fast as possible, he does not want the network to be a bottleneck. So, with no exchange of information between the network department and the computing department, the computing department selects a processing site with enough network resource to reach it and enough processing power to execute the job. The computing department can only make an estimate of the network performance using unreliable estimations. If the computing department had enough information, it could select a computing site and a network path to reach it. So that according to the user's

job, “network power” could compensate “computing power”. This is the idea of cross optimization.

The Grid scheduler must select resources to execute each jobs. Today, this selection process is based on the computing server parameters: liveliness, load, speed, etc. One of objective of this thesis was to also consider network parameters in this selection process: reachability, maximum bandwidth, average bandwidth, latency, jitter, as well as other constraints. Taking into account network parameters will reduce the overall processing time. For example: a slower server with more bandwidth available could be a better selection than the fastest server with almost no bandwidth available.

Furthermore, jobs express their resource requirements with time constraints. So the Grid scheduler has to select:

- Which resource to use, (e.g. the network path, the processing site)
- The amount of resource to use (e.g. the amount of bandwidth, the number of CPU)
- Resource consumption start times (e.g. start time to use the network and start time on the execution site)

This leads to several parameters that have to be determined for each job and offers a larger number of combinations. Of course, the computing department can try to achieve several optimization objectives. In the most complex case, the computing department is able to propose a routing depending on the job characteristic. So a routing problem as well as and end-resource selection problem have to be solved simultaneously. Both problems are time dependent, the complexity of the problem to solve is at least larger than the complexity of problems taken independently.

The focus is on applications that can benefit from a cross optimization. Some applications simply requires resource x,y,z, starting from a moment the resource orchestrator has to determine. For instance, a streaming problem requires the network resource at the same time as the network resource. Such a problem is of very little interest because there is no cross-optimization possible, a streaming has to occur at a given precise rate, a processing unit powerful enough has to be available. The selection mechanism is overly simplistic.

4.2 Related Work

All techniques related to optimizing data location like [100] in order to minimize the access delay will not be discussed.

4.2.1 Basic Resource orchestration algorithm

Suppose a resource orchestrator has to be built. The aim of the orchestrator is to find a time when all resources requested by a job will be available to process the job. Here a job is composed of several tasks that are dependent on one another, each task must use one or several resources. Each task must start at a precise time after the start of the job, each task has a certain duration known beforehand. Suppose that a resource is either available or not and there is no compromise that can be made on the quantity of resource. Either it is available or not. For this kind of non tolerant jobs, a basic resource orchestration algorithm is obvious. All resources are scanned, the first set of resource available that match the request are allocated to the job.

4.2.2 Max Flow approaches

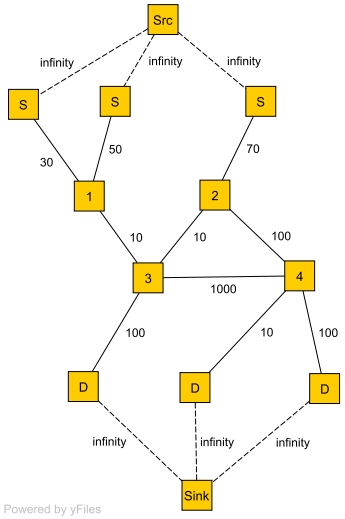
One of the first ideas to approach the job allocation problem is the maximum flow formulation [38, 77]. The objective could be to process as many jobs as possible.

Ford and Fulkerson [68] were the first to provide an efficient solution to the maximum flow problem. Before them, Integer Linear Programming techniques and the Simplex algorithm could have been used to find the maximum flow. Schrijver [137] gives a complete description of the history of the algorithm. From Kantorovich in 1939 to the US Army's formulation leading to Ford and Fulkerson's work, until recent development concerning its complexity.

> In their first report on maximum flow, Maximal Flow through a Network, Ford and Fulkerson mentioned that the maximum flow problem was formulated by T.E. Harris as follows:
>
> > Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other.

The mathematical formulation of Max Flow can be found in [77, 138] or any good graph theory textbook. The idea is to find the amount of maximum flow one can achieve between a source node and a destination node through a network, whose links have been assigned a capacity. To find this maximum flow, one has to identify the flow unit or the link capacity unit. Is it similar to the amount of goods in a transportation network or cars per second in an automobile network? Furthermore, the first max flow problem has been widely studied between one source and one destination. It's well known that finding the maximum flow in a network between several sources and several destinations is a problem

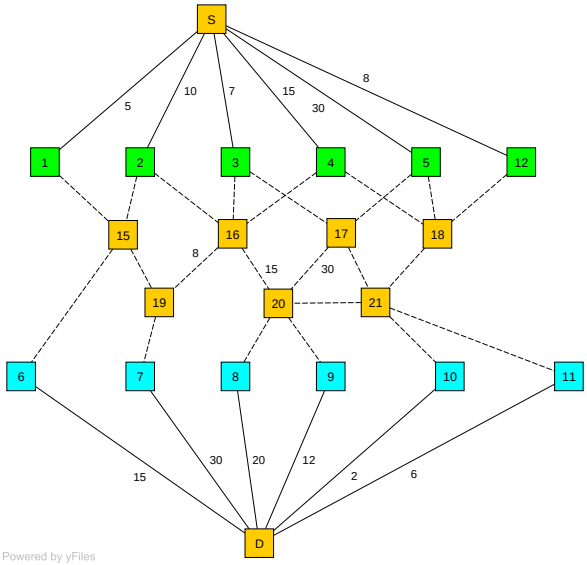
Figure 4.1: Max Flow's extended graph to handle multiple sources and destination



equivalent to finding the maximum flow between one source and one destination in an extended network. Diagram 4.1 shows how to build the extended network.

Suppose a network as described by diagram 4.2. The first row is the source nodes and the last row is composed of CPUs (destination nodes). One job and only one per CPU. Links capacity is expressed in number of jobs. The link capacity could be expressed in number of jobs if each job requires a fixed amount of bandwidth. This formulation

Figure 4.2: Max Flow [Number of Jobs]



determines the maximum number of jobs that can be allocated in the system composed of networks and CPUs. This number of jobs should be compared to the number of accepted jobs determined by the methodology developed in this thesis (see section 4.5.2).

A variant of the previous model can be developed with the flow unit expressed in job per seconds. Link capacity should also be expressed in jobs per second, taking into account a fixed amount of data to transfer for each job. This model is not the way to solve the job placement problem, however it could give some interesting results in terms of the maximum job processing rate achievable by the system. This number could be a performance upper bound interesting for a job scheduler's design. Such problems would have the drawback that some source nodes will not necessarily have a route towards a processing location. In order to make the algorithm compute a route from each potential source location to a computing location, this model could be extended by a compatible maximum flow search algorithm. This algorithm, although not really seen in the literature is very simple, it is just a compatible flow search, followed by the ford-Fulkerson algorithm that searches for the augmenting path as long as the augmenting path respects the flow compatibility constraint. The flow compatibility constraint in max flow is that on each link the flow must be lower than the link capacity, in the compatible flow search, the flow on each link must be lower than the link capacity and bigger than a lower bound. Using this technique, one could add a lower bound on each link between the super-source and job location node, this lower bound would be equal to its upper bound, it would correspond to the real job arrival rate. A compatible max flow search would solve the limitation that some jobs locations have no route towards a processing location. This approach could be used for:

- Getting an upper bound performance indicator for a job scheduler
- Getting a routing for a job placement problem, if the distribution of job follows the same repartitioning among nodes than the job emission rate, Maxflow applied to this model would give a routing of jobs. The job placement problem is solved, but it might assign more jobs to a CPU location than it can support, even if this placement is derived from a correct job rate flow. Of course, bandwidth can not be guaranteed in this placement, because when jobs are processed, the number of jobs that will come from this location is not known. But this allocation could be used for comparison purposes when other job placement techniques will be studied.
- Furthermore, maximizing the job-processing rate is not equivalent to minimizing the overall job processing time.

Because of these limitations, multi capacity links are interesting. I.e. capacity expressed in multiple dimensions, for instance, one dimension for the number of jobs supported by the link and one dimension for the link bandwidth.

A last variant of these Max flow models is the multi constrained. The algorithm used here could be a modified version of the compatible max flow search, because the second dimension of the flow needs to be taken into account. Firstly, the idea is to maximize the number of jobs processed and in the set of solutions that maximize the number of jobs processed, minimize the overall processing time or maximize the bandwidth flow. Here

Table 4.1: Max Flow Multi-constraints

Model	Flow is a job rate, its unit is job per second One node per job One node per CPU One node per network node One link per network link, its capacity is expressed in Mbit/s and in maximum number of job, saying for instance that each job must receive one Mbit/s One link between job i and the super-source (S). This link capacity is the job maximum bit rate, and 1 for the job dimension One link between CPU j and the super-destination (D). This link capacity is CPU network maximum bit rate and 1 for the CPU dimension.
Underlying hypothesis	Only one job can run on a CPU, no more. As a consequence a CPU is either busy with one job, or idle All CPUs are the same When the first dimension of the flow is non null, the second dimension is non null All jobs are the same

minimizing the overall processing time is equivalent to maximizing the bandwidth because all jobs are equal.

These approaches are interesting for comparison purposes. The requirement that jobs must be handled one after the other (on line algorithm), implies that it is not possible to optimize the allocation as if all jobs requests were known in advance. So the algorithm proposed will be the simplest cross optimization algorithm one can think of: "Select as much as processing and network resource as possible in order to minimize the job's processing time".

4.3 Methodology

The objective in an industrial context is to implement an optimizing algorithm that can handle all situations. In this context, such an algorithm must be able to determine resource scheduling for the jobs on the fly. I.e determine when and which resources should be used by each job. The objective of such an algorithm would be to minimize the overall job processing time and/or the job rejection rate. Classical techniques to handle this kind of problem can be applied to produce different results. Exact results could be produced with an Integer Linear Programming formulation, approximate results with meta-heuristics. Heuristic techniques will be used, although the optimum found is not the best one, their simplicity and speed of execution make them ideal candidates for an implementation

in an industrial product. This chapter studies the benefit of the interaction between a Grid scheduler and an NMS to co-allocate resource for jobs. Jobs are composed of a data transfer and a computing task. This chapter limits its study to jobs that can not be delayed: either they are accepted and they start immediately or they are rejected. A cross optimization algorithm *XO* for Grid schedulers to better co-allocate resources minimizing the job completion time seen by the end-user is proposed. The objective is to evaluate the impact of this XO on the total job processing time and compare it with an algorithm that models a simple bandwidth on demand service. An extension of the algorithm proposed could be implemented for planned jobs request, i.e. jobs that can wait before their execution start. Appendix 4.6 is a description of this extension. The Grid scheduler has to select for each job a computing node and network resources. Three solutions are proposed: *Legacy*, *CCB*, *XO*. The legacy solution is the overlaid solution: the network is used without reservation, with the best-effort service class. *CCB* is Connection with crank-back, it means that the Grid scheduler does not have the knowledge of the network (virtual) topology nor the amount of reservable bandwidth on each link, it only interacts with a bandwidth on demand service provided by the network operator. When a computing site can not be reached, another request is sent until an available computing site is found and reachable.

To simplify the complexity of comparing and evaluating the performance of each algorithm and yet to evaluate the effect of cross optimization, the simulation tools will not rely on a discrete event based simulator. Instead, a tool that would compute the number of accepted jobs and the average processing time for each algorithm was built. This tool takes a fixed number of jobs as an input parameter. As a consequence it was possible to compute the number of accepted jobs leaving time dependent parameters for a second study. The complexity of the problem and the number of parameters in the model requires this step by step approach. In order to better identify the impact of each parameter taken independently and to understand more deeply the overall system. These first results should be used to design and propose a more complex solution.

Jobs

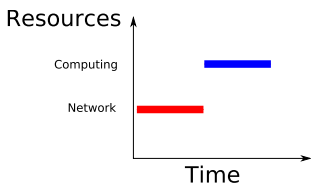


Figure 4.3: Resources use pattern

Jobs are a sequence of two sub-tasks: a data transfer and an execution of some code contained in the data transfer. To evaluate the benefit of *XO*, these simple tasks seem to be a good representation of basic batch job execution processes. A job request j will be characterized by a source node, an amount of data to transfer $S(j)$, an execution duration

estimation on a reference machine $D(j)$, an access rate $A_r(j)$ independent and identically distributed (i.i.d.) random variable. The distribution of A_r is uniform between between 10Mbit/s and 10Gbit/s. Each job requires one and only one CPU. The amount of data to transfer (resp. execution time estimation) is an independent and identically distributed (i.i.d.) variable following a uniform distribution. Jobs will be modelled by the coupled

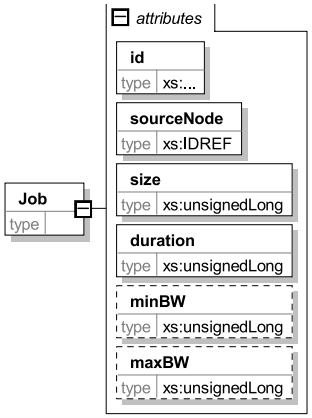


Figure 4.4: Job schema

file size and execution time. It requires the user to have an estimate of its job execution time. Such information can be easily obtained after having run the program a few times. Runtime on different machine can be estimated using benchmarks [61, 62, 63]. Otherwise, it is also possible to consider that the time given by the user in the job description, is an upper boundary of the execution time. Incentive mechanisms could be used the make the user improve its job execution duration estimation. The model described here implies a job composed of two sub tasks. A job composed of more sub tasks like stage-in, execution, stage-out will be treated as two independent jobs: the first job would be composed of the stage-in and execution sub tasks and the second job of the stage-out sub task. In parallel computing jobs require more than one CPU. They are composed of at least several threads or process that work together in parallel. A job description [42] can contain explicitly the number of CPU required for its execution. The model described could be easily extended to take it into account. Job's characteristics, i.e. file size, duration, access rate will be generated according to different distributions.

Network

The network will be modelled by a graph composed of nodes and links. Links will be given a capacity following two distributions: uniform or discrete. The discrete law is obtained by a discretization of a positive Gaussian at 2.5Gbit/s, 10Gbit/s, n x10Gbit/s. Diagram 4.5 shows the network data model. Generating random topologies that model current existing network is a complex problem by itself and has been studied [64, 144, 48] for a long time. Waxman [155] random graphs will be used to generate topologies. Kuipers [93] explains very well the class of Waxman graphs. "The class of Waxman graphs belongs to the class

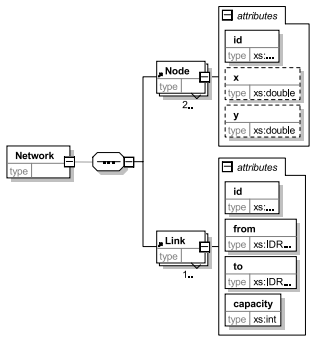


Figure 4.5: Network model

of random graphs, where the probability of existence of a link between two nodes decays exponentially with the geographic distance between those two nodes. Such graphs are often chosen because of their resemblance to actual network topologies." More formally, the probability p_{ij} to have a link between node i and node j, is a function of the distance given by the following formula. The probability of not having a link between those two nodes is $1 - p_{ij}$. The probability of link appearance follows a binomial law of parameter p_{ij}.

$$p_{ij} = \alpha \mathbf{e}^{-\frac{\mathbf{d(i,j)}}{\beta}} \tag{4.1}$$

"The idea of relating the probability of a link between node i and node j to some function of the distance between those nodes stems from the correspondence with realistic telecommunication networks." The farther two nodes lie separated, the smaller the need for a direct link between them. Parameters α and β influences the average node degree [150, 109] and density. Diagram 4.6 gives an example of two Waxman graphs of different theoritical average degree for a 20 node network. To generate a Waxman graph

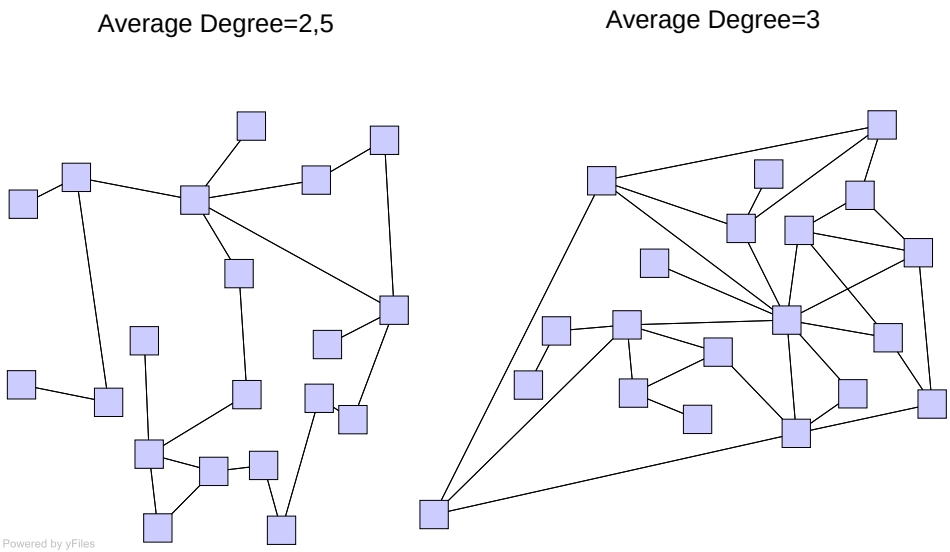


Figure 4.6: Waxman Graphs

of n nodes, first nodes are geographically uniformly distributed in a plane. Then, p_{ij} is computed and links are created according to a binomial random variable of parameter p_{ij}.

Clusters

Several clusters spread across the network are attached to one network node. Each cluster has a given number of processing units (CPU). Each processing unit can be allocated to no more than one job. Within a cluster, all processing units have the same processing power. CPU processing power is uniformly distributed amongst clusters.

The remaining paragraphs briefly describe the algorithms used to model the three different interactions between the NMS and the Grid scheduler.

4.3.1 Legacy

This allocation algorithm models no interaction between the Grid scheduler and the NMS. It considers a network with no reservation capability and no admission control. All jobs are accepted in the network as long as they have been allocated a computing resource. The purpose of this model is to represent an today's most frequent situation. The idea of the algorithm is very simple: jobs are processed in a given order; CPUs are ordered by their decreasing processing power. The first job is assigned to the first CPU and so on. As a result, each job is assigned to the first available CPU ordered by decreasing processing power. Once the execution site for each job is known, the traffic matrix of the network can be determined. For each couple (source, destination) the number of jobs that has to flow in the network is known. The network routing is assumed to be static: a route is predetermined before all jobs arrive in the network and this route will not change during the whole execution of the algorithm. Two variants will be investigated; the first variant is when the routing is either a MinHop or MaxBW routing. MinHop (resp MaxBW) routing is a routing that minimize the number of hops (resp maximize the bandwidth) between each couple (source, destination). Problems that it raises are:

- How is the bandwidth going to be shared on a route between different jobs?
- What happens if congestion occurs?

To answer the first question, suppose that each job is entitled to an equal amount of the available bandwidth. So if a route of capacity C must share C between p jobs, each jobs is entitled to C/p. However, not all jobs are capable of emitting at the rate C/p. So slow emitting rate jobs should free some bandwidth for the fast emitting rate jobs. The residual bandwidth should be shared equally among jobs that are not running at their maximum emitting rate. However, some links are shared between two or more routes, so it

is possible to have congestion on shared links. If congestion occurs, the access rate of jobs is decreased by a multiplicative factor (less than 1) and this mechanism is iterated until no link is overloaded. However the network shall not be under-utilized, so the decreasing factor should be choosen in order to reduce congestion but not too fast. Algorithm 1 and 2 determine the CPU, the network path and the amount of bandwidth for each job.

Algorithm 1: Legacy allocation algorithm

Require: A network graph $\mathcal{G}$
Require: Set of clusters $\mathcal{C}$
Require: Set of jobs $\mathcal{J}$
Let r_{ij} be the route between node i and j
Let n_{ij} be the number of jobs on r_{ij}
$\forall(i,j) n_{ij} = 0$
Sort clusters $\mathcal{C}$ by decreasing processing power
for all jobs J of $\mathcal{J}$ **do**
    Assign one processing unit of the first element c of $\mathcal{C}$
    if c has no more available processing unit **then**
        Remove c from $\mathcal{C}$
    end if
    Let i be the source node of J
    Let j be the attached node of c
    Let $R(J) = r_{ij}$ be the route of J
    $n_{ij} = n_{ij} + 1$
end for
repeat
    $R = DetermineBW(\mathcal{J}, r, n)$
    Let $MaxLoad$ be the load of the most congested link l
    for all jobs J using l **do**
        Reduce J's maximum emitting rate of 80%
    end for
until $MaxLoad > 1$

Once CPUs have been assigned to each jobs, the number of jobs each route must support is known. For each route, the bandwidth each job can use is computed according to the mechanism described earlier. Then the most congested link is detected and the congestion avoidance mechanism is applied to jobs that use this link. It means that the access rate of these jobs is reduced and the whole process is iterated with new emitting rates for these jobs until no link is congested. The congestion avoidance mechanism described here is not the one used in TCP, it will be discussed in the next section. The impact of the decreasing factor on the utilization ratio of the network could also be studied. What is the ideal decreasing factor? What is the impact on the most loaded link? Does this algorithm manage to get a network with the most loaded link load close to 1? After several tests, 80% seemed to be a good factor. Results of these tests are not shown here.

Algorithm 2: Job's used bandwidth determination

Require: Set of jobs $\mathcal{J}$
Require: Routing table R
Require: Allocation table n
for all Routes r of R, $n_{ij} > 0$ **do**
  let J_r be the jobs using r
  Let $Fair$ be r's capacity divided by n_{ij}
  repeat
    Allocate $Fair$ to jobs of J_r with undetermined bandwidth
    if Some jobs can't use $Fair$ because of their maximum emitting rate **then**
      Allocate them their maximum emitting rate
      Share the remaining capacity $Capa$ among the remaining jobs $Others$
      $Fair = Capa/Others$
    end if
  until all jobs of J_r's bandwidth is determined
end for

4.3.2 Connection with CrankBack (CCB)

In this model, the fact that the NMS will not provide any information regarding network current utilization is taken into account. The Grid scheduler can only use a bandwidth on demand service provided by the network operator. Furthermore, this model will take into account or not a crank-back mechanism. In the no crank-back variant, if it is impossible to reach the best available CPU for a given job, then the job will not be processed. The algorithm will skip to the next one. In the crank-back variant, if it is impossible to reach the best available CPU for a given job, then the algorithm will try to reach the second best CPU and so on until a CPU is reached, or none.

The algorithm is very simple, it will process all jobs sequentially. For each job, it will select the most powerful CPU. Then it will request the NMS to set up a connectivity of the maximum bandwidth available respecting its emitting rate limit. If no path is available, then, the computing department will skip to the next job and loop. The algorithm just described is the no crank-back variant, because a very simple "cross-optimization" could be to try to set up a connexion with the second best processing site and so on until a it is possible to set up a connection towards a processing site, or none. It's the crank-back variant.

4.3.3 Cross Optimisation (XO)

One of the objective would be to minimize the sum of all processing times for all jobs, where processing time is given by Equation 4.2. Our final objective is to have an in-line algorithm. Meta-heuristics will not be used here to find a global optimum, nor any

Algorithm 3: CCB

Require: A network graph $\mathcal{G}$
Require: Set of clusters $\mathcal{C}$
Require: Set of jobs $\mathcal{J}$
Extend $\mathcal{G}$ to include job nodes and clusters nodes
for all Job J of $\mathcal{J}$ **do**
  Sort $\mathcal{C}$ by decreasing power
  for all Cluster c of $\mathcal{C}$ **do**
    if a CPU is available on c **then**
      Let i be the source node of J
      Let j be the attached node of c
      Let r_{ij} be the set of maximum Bandwidth routes between i and j
      if r_{ij} is empty **then**
        Do Nothing and skip to the next job (no crank-back variant)
        Do Nothing and skip to the next cluster (crank-back variant)
      else
        Allocate J on the first element of r_{ij}
      end if
    end if
  end for
end for

feedback based ideas. The algorithm proposed will not minimize the average waiting time. Instead it minimizes the job's waiting time without taking into account other jobs, but using as many resources as possible among available ones. So for each job resources will be selected in order to minimize its processing time, as given by Equation 4.2.

$$\sum_{j \in Jobs} \frac{S_j}{BW_j} + \frac{D_j}{POW_j} \tag{4.2}$$

Where S_j is the size in Mbit to transfer in the network of job j, BW_j is the reserved bandwidth for job j, D_j is the expected duration of job j on a reference machine, POW_j is the power factor of the CPU allocated for j. In other words, this algorithm models a computing service with objectives such as "get this done as soon as possible". To perform this optimization, this algorithm requires a deep exchange of information between the Grid scheduler and the NMS.

The algorithm handles jobs sequentially. Each job is characterized by its file size and its execution time. First all potential execution destinations are found, those are the destination where there is still a CPU available. Then for each of these destinations, the available routes that maximize the bandwidth between the source (taking into account its access rate) and the destination are found. Then for each destination cluster, the time it would take to run the job on this cluster via the best available route is computed. And the cluster and the route that minimize this time are selected. If several route are possible the route that also minimize the number of hops is selected. Once the best route and the

best cpu for a job are found, network and computing resources are reserved and the next job is processed. Obviously, it is possible that some jobs will not be processed, it will happen when there is not enough resources whether network or processing. To implement

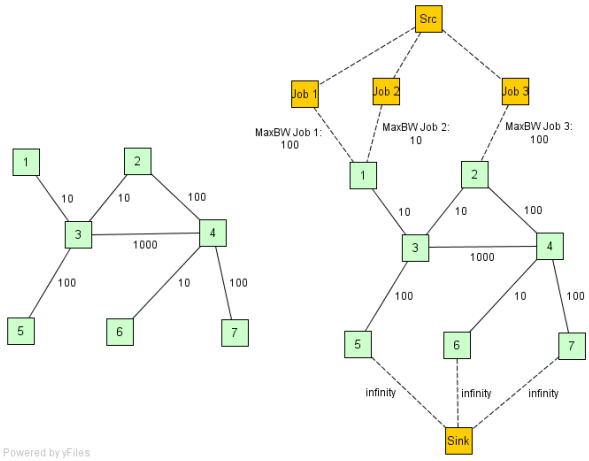


Figure 4.7: Extended Graph example

this algorithm, an extended network graph is used to take into account jobs and clusters. Ie. New links and new nodes will be added to the original network topology to model a job, its access rate and a cluster. More precisely, a new node and a new link between the new node and the source node of the job will be added. The link capacity will be the maximum emitting rate of the job. A similar approach is used for clusters. Furthermore, a super source node connected to all new job nodes and a super sink node connected to all new cluster node can be created. The capacity of links connected to the super source or the super sink is infinite. Diagram 4.7 shows an example of an Extended Graph.

It should be noted that the following code is pseudo-code, it differs from the real implementation. To simplify notations and avoid technical programming details, two global variables, `routeOfJob[Job i]` and `ClusterOfJob[Job i]` which are two tables containing the route and the cluster of Job i will be used. A route will be implemented as an object that has a list of edges and a capacity.

Step 1 is important because new created link's capacity are set to the job's maximum emitting rate.

Links with Multi-criteria routing (Pareto set) RMB1C Let's first recall the definition of the Pareto optimality. In a set $S = a_i = (x, y)_i = (x_i, y_i)$ of elements a_i. a_j dominates a_i if equation 4.3 holds.

$$a_i \leq a_j <=> x_i \leq x_j and y_i \leq y_j \tag{4.3}$$

The Pareto set is the set of elements a_i that are not dominated.

Algorithm 4: XO

Require: A network graph $\mathcal{G}$
Require: Set of clusters $\mathcal{C}$
Require: Set of jobs $\mathcal{J}$
1: Extend $\mathcal{G}$ to include job nodes and clusters nodes
2: **for all** Job J of $\mathcal{J}$ **do**
3: Let R_J the set of maximum bandwith routes towards the super sink
4: **for all** Route r of R_J **do**
5: Compute the execution time of J if it were allocated on r
6: **end for**
7: Allocate J on the route of minimum execution time
8: **end for**

There is a link between *XO* and multi-criteria routing. The objective of the multi-criteria routing is to find the Pareto set of optimal routes between a source and a destination. Once this set of optimal routes is computed, a route is chosen is the set according to some criteria, e.g. by computing a weighted average on the different components of the route. *XO* is computing a weighted average except that weights are not defined to meet the network operator's objectives, but are dependent on the user's job characteristics. And the weights will vary from one job to the other. Finding all Pareto optimal routes between the super source and the super sink is equivalent to step 3, that is to say selecting all potential destination and maximum bandwidth routes towards each one of them. Selecting the best route according to a weighting function among the Pareto optimal routes is equivalent to step 7, the one that will minimize the job processing time.

4.4 Analytical results

All scenarios are run on a random topology. This section tries to predict performance results of the different algorithms. The first question is how many jobs can be allocated in the system composed of network and clusters? The second question is what is the average job processing time?

Nb Job and routing coefficient

First, the simplest case when the "network" is composed of just one link and two nodes and that jobs are not file transfer but bandwidth requests is considered. Let C be the random variable (r.v.) of the link capacity, A_r the r.v. of the bandwidth request (Access Rate). The number of satisfied bandwidth requests on that link is given by $\lceil \frac{E(C)}{E(A_r)} \rceil$. It's an application of Wald's identity [75] (page 233). Note that the last bandwidth request might be granted the remaining bandwidth and not A_r. The quantity $\frac{E(C)}{E(A_r)}$ is called

the normalized link capacity. When the p links are available between the two nodes, the same results applies except that C is replaced by pC. A proof of that is given here, let's recall Wald's identity. Wald's identity is applied with the stopping time N defined by the number of jobs on one link.

Theorem 1 (Wald identity) *Let (X_i) a family of i.i.d. real r.v. such as $E(X_1) < \infty$ and T a stopping time for the σ-field $\mathcal{F}_n = \sigma(X_i; i \leq n)$. If $E(T) < \infty$ the following equation holds*

$$E(\sum_{i=1}^{T(\omega)} X_i) = E(X_1)E(T)$$

The number of jobs is the minimum between the number of CPU and the number of jobs the network can support. The following will consider that the number of jobs is not limited by the number of CPU.

Proposition 1 (The Number of jobs is a stopping time) *Let C be the capacity of the link. C is a random variable. The number of jobs N,* XO, CCB *and* CnoCB *can allocate on the link is solution of*

$$\sum_{k=1}^{N} A_{r_k} > C \text{ and } \sum_{k=1}^{N-1} A_{r_k} < C \tag{4.4}$$

Said differently N is a stopping time, it is the first entrance time for random variable $\sum_{k=1}^{n} A_{r_k} - C$ in the interval $]0, +\infty[$.

Indeed, the reserved bandwidth for the $N-1$ jobs allocated will be equal to the maximum emitting rate, it is only the last job that will reserve the residual capacity of the link which is by definition of the last allocated job, smaller or equal to A_{r_N}.

When the network composed of several links and several nodes, the number of satisfied bandwidth requests between two nodes is less obvious to compute. It depends highly on the number of nodes, the number of links, the routing algorithm and the node degree. The problem is even more complex if job requests have different origins and potentially different destinations.

Lee [98] studied the maximum flow problem on random graph between two arbitrary nodes, i.e. between one source and one destination. Actually he studied the maximum flow along links of unit capacity (called maximum unitary flow), as a consequence he really studied the number of link disjoint paths p between two nodes. Thanks to the max-flow/Min-cut theorem, the maximum unitary flow equals the number of links p of the minimum cut. Lee's result can be summarized by the following Theorem.

Theorem 2 (Lee's law) *Average maximum unitary flow is proportional to the smallest of source's degree and destination's degree.*

This result is true for the max flow routing and is just an application of Max-flow/Min-cut to Lee's results and Wald's identity. It could be interesting to generalize Lee's results and thus his law for s sources and d destinations and to any routing algorithm. This is the tentative of the following conjecture.

Conjecture 1 *Given a random graph $\mathcal{G}$, a routing algorithm X, the number of bandwidth requests satisfied between s sources and d destinations is proportional to the normalized link capacity. The proportionality coefficient is called the routing coefficient of X between s sources and d destinations in $\mathcal{G}$. i.e.*

$$N = \kappa_{X_{sd}}(\mathcal{G})\lceil\frac{E(C)}{E(A_r)}\rceil \tag{4.5}$$

If this is true, the equivalent capacity expressed in Mbit/s of an allocation algorithm X between s sources and d destinations can be defined as $\kappa_{X_{sd}}(\mathcal{G})E(C)$. While N is the equivalent capacity expressed in bandwidth requests units. If Wald's identity is applied to the minimum cut and Lee's law. It's the proof that the routing coefficient of the allocation given by MaxFlow between one source and one destination is the minimum of the source and the destination's degree. Another study shall be done to decouple in the routing coefficient the influence of a structure coefficient and of the allocation algorithm. The next section will verify if Conjecture 1 is true for *XO*, *CCB* and *CnoCB* when job request can be considered equal to bandwidth requests of bandwidth A_r. This second assumption is hypothesis 1.

Hypothesis 1 *Satisfied job request's reserved bandwidth distribution for* XO, CCB *and* CnoCB *is equal to A_r's distribution.*

Job processing time

If

1. hypothesis 1 holds,
2. on the set of accepted jobs the job computation duration's distribution is the same as the distribution of all of the job requests
3. and the transfer size follows the same distribution as the job request's in general
4. the allocated CPU processing power follows the same distribution as the CPU processing power in general

then, the job processing time is given by:

Proposition 2 *when* (S, D, BW, POW) *follows a continuous uniform distribution,*

$$E(T) = \frac{m_S}{l_{BW}} \ln \frac{BW_{max}}{BW_{min}} + \frac{m_D}{l_{POW}} \ln \frac{POW_{max}}{POW_{min}} \tag{4.6}$$

where $m_X = E(X) = (X_{min} + X_{max})/2$ *and* $l_X = X_{max} - X_{min}$.

The computation of the average of the job processing time $T = S/BW + D/POW$ when (S, D, BW, POW) follows a continuous uniform distribution gives the result. It should be noted that assumptions made do not differentiate allocation algorithms. This model was used to formalize the difference between the different algorithm, but it cannot be solved analytically and the formulation does not give any interesting results.

4.5 Results

This section will presents the main results on cross optimization. First the normalized link capacity is verified to be an important parameter. Then conjecture 1 is verified. The influence of the number of CPU is then analyzed. Once the number of accepted jobs is known, the last sections verifies the average processing time prediction given by formula 4.6.

4.5.1 Normalized link capacity

The main performance measure for an allocation algorithm is the number of jobs it can allocate on resources. Of course, the more resources that are available the bigger the number of accepted jobs is. The less resources jobs require, the more that are accepted. So the number of accepted jobs increases with the average link capacity, number of CPU but decrease with the access rate. The previous section showed the importance on one link of the normalized link capacity. Several simulations were conducted to show that it is also the key parameter in a network. Leaving all other parameters unmodified, varying $E(A_r)$ and $E(C)$ while keeping a constant ratio, it was observed that the mean number of accepted jobs was the same. Diagram 4.8 illustrate this.

4.5.2 Number of accepted Jobs

Diagram 4.9 shows the impact of the number of network resources on the number of accepted jobs. It also shows the influence of the number of node and average node degree.

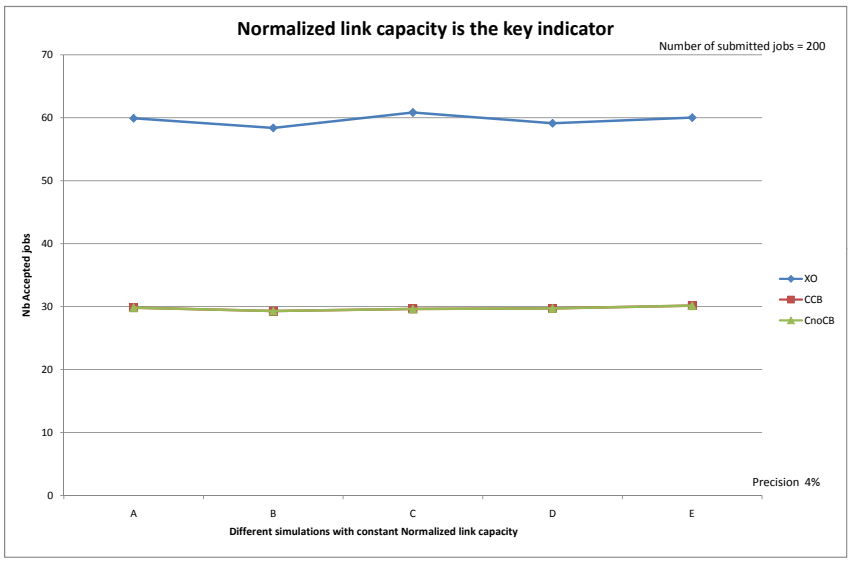


Figure 4.8: Normalized link capacity is a key parameter

It was obtained submitting 1000 jobs in 10, 20 and 30 node networks for two average degree 2.5 and 3. The precision of these graphs is given by the relative confidence interval size: 2%. The total number of CPU is 2000, i.e. the ratio number of CPU to number over submitted jobs is 2. So in this graph, the network is the bottleneck resource. Similar curves were obtained for CPU ratio of 1 and 0.5. Although these curves were plotted for constant network link capacity, similar curves were also obtained for the discrete link capacity distribution. The main difference between all curves is that the routing coefficient varies either because of the allocation algorithm or because of structure parameters (degree, number of nodes, link distribution, CPU ratio). A more precise analysis should be conducted to evaluate the impact of the number of node, the average node degree, the number of clusters in the network, the number of source nodes, the repartition of jobs among source nodes and the repartition of CPUs among clusters. Results shown here were obtained when all jobs (resp CPUs) are uniformly distributed among source nodes (resp clusters). A network node that is not attached to a cluster is a source node.

The main finding is that the benefit of *XO* over *CCB* is negligible, it is below the precision of these graphs roughly 2% and that conjecture 1 clearly holds.

4.5.3 Number of CPU influence

Diagram 4.10 shows the influence of the total number of CPU on the number of accepted jobs. It was obtained for a normalized link capacity of 16.

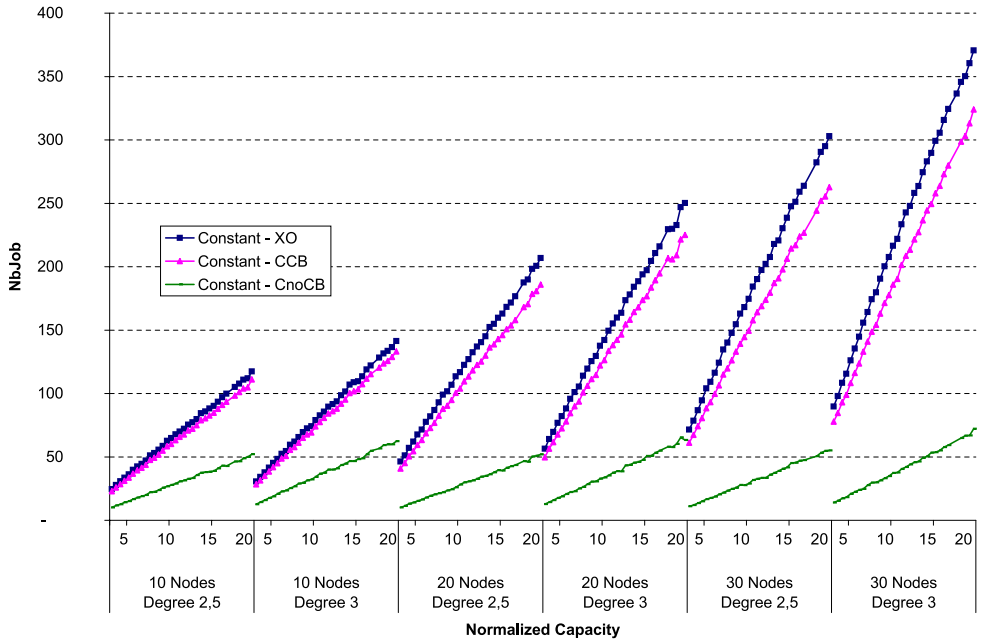


Figure 4.9: Number of accepted jobs

All curves were plotted with the same average link capacity and same access rate distribution. The CPUs' power is uniformly distributed amongst the clusters but constant within a given cluster. Similar curves were obtained when the network link capacity is distributed. It was observed that the number of accepted jobs is smaller than when network link capacity is constant. This is natural: the distribution increases the asymmetry of the link capacity and induces a penalty because a route's bandwidth is the minimum of its links' capacity. So the equivalent capacity in Mbit/s will be smaller.

XO's number of accepted jobs is bigger than *CCB*'s and *CnoCB*'s but it is only slightly better than *CCB*. *XO* allocates jobs based on the set of network resources and processing resources that minimize each job's total processing time. There is absolutely no reason that would make *XO* select the same resources for two consecutive jobs. So, *XO*'s utilization of resources is better. It balances load more efficiently than *CCB* and *CnoCB*. "Connexion without Crankback" or *CnoCB* first selects the best processing site, then tries to see if network resources are available towards this processing site. If no network resources are available to reach a site, it stops. It does not use all available resources like *CCB* does by selecting the second best processing site and so on. As a consequence, *CCB*'s number of accepted jobs is much bigger than *CnoCB*'s. *CCB* and *CnoCB* perform similarly when all CPUs are identical because processing sites are assigned jobs one after the other in the same order for both algorithms. *XO* and *CCB*'s curves are either constant or they saturate. Constant curves are obtained when there is not enough network resources and saturating curves are obtained when first the CPUs and then the network are limiting the number of accepted jobs. In the diagram, small networks can not accept more than 200 jobs whatever the normalized link capacity (see Diagram 4.9). This is reflected in Diagram 4.10 by the constant curves for 10 and 20 nodes networks. The 30 node network

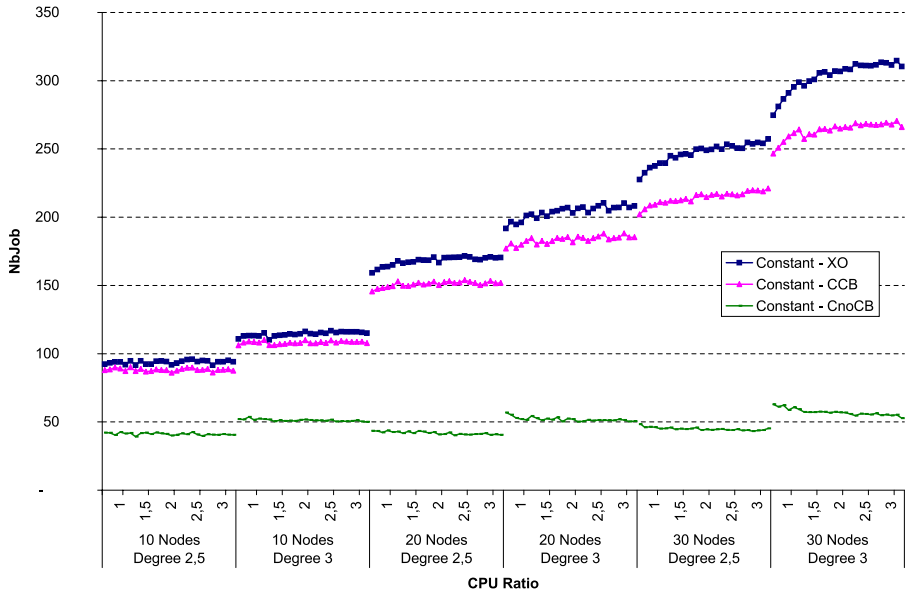


Figure 4.10: Number of CPU influence

These curves were obtained with the following parameters:

Number of Nodes	10, 20 and 30
Average Degree	2.5 and 3 (for all node number)
Average Link capacity	80Gbit/s
Number of Jobs	1000
Average Job Size	50Gb
Job size span	10Gb
Average Duration	5000s
Duration span	1000s
Number of Processing Sites	3, 6 and 9 (respectively)
Total number of CPU	$1000x$
Average CPU Power Coefficient	1000
CPU Power Coefficient Span	500
Access Rate minimum value	10Mbit/s
Access Rate maximum value	10Gbit/s
Link capacity Distribution Type	Constant
Other distribution type	Uniform

of average degree 3 of normalized capacity 16 accepts more than 300 jobs when there are enough CPUs but less than 300 when the number of CPUs is limiting.

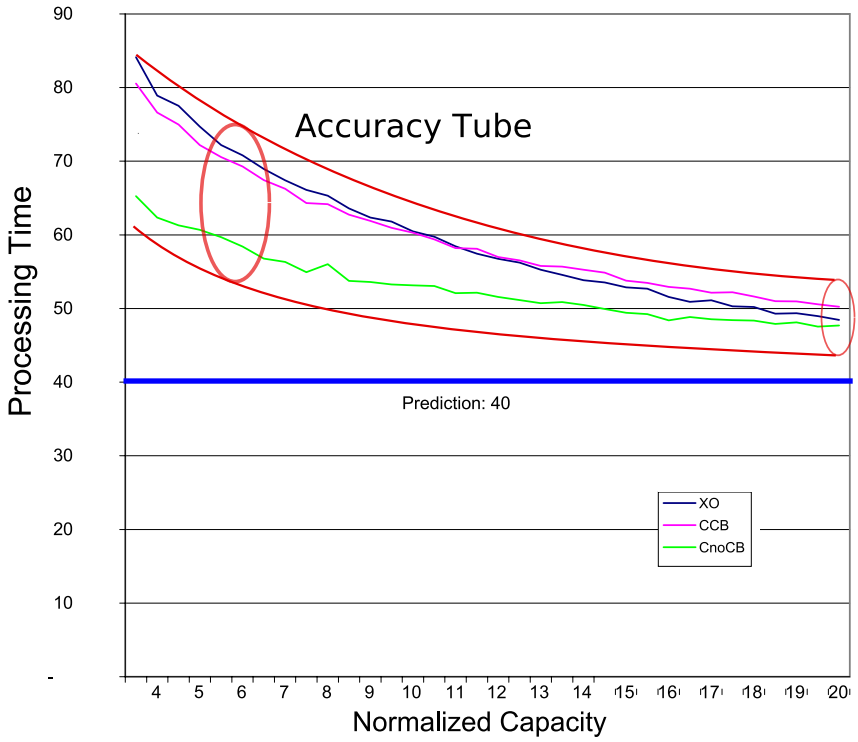


Figure 4.11: Average processing time

4.5.4 Processing Time

Diagram 4.11 shows the average processing time for allocated jobs for the three different algorithms as a function of the normalized link capacity. Curves *LegacyMaxBW* and *LegacyMinHop* are not plotted. They are way outside of the scope, more than a factor 100. It clearly shows the benefit of reservations (*XO*, *CCB* and *CnoCB*) over the best effort approach on the average processing time. *LegacyMaxBW* and *LegacyMinHop* produces the same results, but of course, reservations limit the number of accepted jobs. The difference between *XO* and *CCB* is negligible. The accuracy of these plots is 15%. Although not plotted here, the order of job processing impacts the average processing time, although the difference is less than the accuracy of the graphs. All curves seem to converge towards the predicted value. Their shape can be partly explained because of the validity of the approximation of hypothesis 1.

4.5.5 Validity of Hypothesis 1

For both link capacity distributions, the approximation is more valid when the normalized link capacity increases. The error is less than 8% in volume for the discrete distribution and less than 4% for the constant link distribution. The idea behind this is that the number of jobs that can not use the maximum emitting rate is proportional to the number

of link disjoint paths and is dependent on the structure of the network graph and the allocation algorithm. In a way similar to [98] although no study has been conducted to prove it, but curves plotted in 4.12 show the dependence on the normalized link capacity is inversely proportional, which is the expected trend shape.

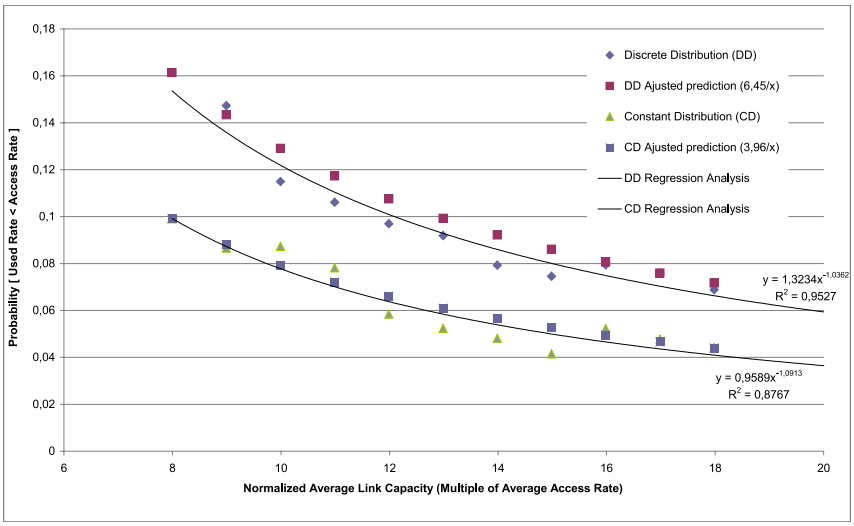


Figure 4.12: Approximation's error

4.6 Conclusions

One of the objective of these simulations was to determine the benefit of a cross optimization over a traditional Bandwidth on Demand service. The key finding is that on networks composed of less than 20 nodes, there is no need to go beyond a bandwidth on demand service, communicating the network topology and reservable bandwidth is not necessary. However, for bigger networks in nodes and link capacity, a cross optimization increases the number of accepted jobs by 20%. A side result, but important one, was to verify that the number of accepted jobs in a network is proportional to the normalized link capacity. The proportionality coefficient was called the routing coefficient and depends on the allocation algorithm, the routing algorithm and structure parameters. A lower bound of the average processing time can be analytically computed. When the normalized link capacity increases, the observed average processing time converges towards the computed lower bound.

The number of jobs computed here is when the network was initially empty and jobs are added until the network is full. A time based discrete event simulation was not conducted, i.e. a simulation such as when jobs are computed, they free resources that can be used by new jobs. Now that the system's equivalent capacity and average job processing time can be computed, future work could try to generalize the results to estimate the performance of an on-line system with a discrete event simulation. An interesting study would be to

check the following statement: the overall system is equivalent to a queuing system of length the computed equivalent capacity and average service time the average job request processing time.

List of Patents and Publications

The following is a list of articles where I am the first author:

- "Grid services over IMS" [117], Broadnets 2006 proceedings published by IEEE Communication Society
- "Grid services over IMS" [118], Published by International Engineering Consortium (IEC) in the report "Business Models and Drivers for Next-Generation IMS Services", 2007
- "Dynamic SLA Negotiation with WS-Agreement" [119], published by the Network of Excelence COREGRID in the Technical Report 82
- "Dynamic SLA Negotiation with WS-Agreement" [120], published in the proceedings of Web Information Systems and Technologies (WEBIST 2008) conference, Best papers will be published in Springer's Lecture Notes in Computer Science
- "Co-allocation & cross optimization of network and computing resources for distributed applications", to submit [116]
- "Performance comparison of negotiation protocol", to submit

The following is a list of patents where I am the first author:

- "Dispositif de contrôle de l'établissement de sessions" [111], patent No EP1845693
- "Elément de type PCE pour le calcul de chemins de connexion prévisionnels dans un réseau de transport" [112, 113], patent No FR2895625
- "Noeud de réseau de transport, à adjonction de données temporelles à des données d'ingénierie de trafic" [114], patent No EP1802076
- "Procédé d'allocation de ressources pour un logiciel de gestion de ressources distribuées" [115], INPI patent application No 0756827

Patent application where I am co-author:

- "Method of providing a gird network application over a transport network" [151], OEB patent application No 07290821.3-1525

Conclusions and Future work

Co-allocation of network and computing resources is possible only if network technology is capable of providing some sort of resource reservation. Inherent to resource reservation, the concept of guaranteed QoS has been progressively introduced in IP networks. Following the failure of the IntServ approach, the DiffServ approach has been a second attempt in this direction but without explicit resource reservation. It is only with MPLS and its successive extensions (MPλS and GMPLS) that guaranteed QoS may be provided today by means of explicit resource reservation.

Grid applications require resource virtualization. During these last ten years, the number and the variety of applications that could benefit of resource virtualization has grown-up dramatically. If the first investigations in this matter have been applied to very prospective domains limited to the research community, resource virtualization will certainly be in the next decade at the heart of numerous advanced commercial applications. In that sense, Grids constitute a huge potential market for telecom operators. A new type of entity known as the Resource Broker will also provide new business opportunities for both the computing and the telecom industry. Virtualization satisfying strong QoS guarantees cannot be achieved without co-allocation. The emergence of (G)MPLS standards enabling guaranteed QoS and traffic engineering has been a key factor for resource co-allocation. Today, several problems remain to be solved for the application of GMPLS to Grid services, that is to resource virtualization. First, we have to determine how GMPLS can be integrated to the Grid Resource Management systems. A second problem is related to the QoS guarantees provisioning which is not limited to an efficient management of network resources like in traditional networks. Grid services also need an efficient management of computing and storage resources distributed over the network. In other terms, Grids require the development of new cross optimization tools applied to both the network, the computing and the networking resources.

These questions have been addressed in the context of this thesis. In Chapter 1, we have provided a state of the art in this domain. Thus, we have described most important research projects between 2004 and 2007 to tackle the co-allocation problem. In Chapter 2, we have presented the main architectures to integrate network and computing resources. Existing approaches have been described: Web service and Grid GMPLS. We proposed a new approach based on IMS and SIP. We also proposed some extensions to the existing control plane to take into account time based reservation. Two publications [117, 118]

and four patents [111, 112, 114, 151] have been accepted for the contributions.

In Chapter 3, we have studied the protocols under development and standardization in order to provide QoS guarantees under both computing and networking resources optimization. More specifically, the protocols for the SLA negotiation and creation have been analyzed. The SLA creation protocol, "WS-Agreement" proposed by the Open Grid Forum (OGF) has been presented and analysed. With O. Waldrich, we proposed an SLA negotiation protocol based on WS-Agreement. We have proposed two phase commit extensions that are required by some applications. This was described in a Technical Report of the Network of Excellence CoreGrid [119] and in a conference paper[120]. We have proposed a queueing model to evaluate the performance of these extensions and we have conducted several discrete time simulations to compare the performance of a two phase commit protocol with the legacy protocol. We have shown that for a given rate of generated job requests, the two phase commit extensions needs to double the memory required by the Grid scheduler.

In Chapter 4, wa have analyzed how far should the exchange of information go between the Grid scheduler and the NMS. Would resources be better utilized, would jobs be executed faster if the Grid scheduler had access to the network's topology and to the available bandwidth on each link? We proposed a cross optimizing algorithm to answer this question. We developed a simulation model to evaluate the performance of our algorithm compared to the one achieved by a bandwidth on-demand service. One of the main original results of this chapter is the fact that the number of jobs accepted in the system is proportional to the normalized link capacity. We have underlined that it is possible to define the system equivalent capacity expressed in number of jobs, or the network equivalent capacity expressed in Mbit/s. The other finding is that on networks composed of less than 20 nodes, there is no need to go beyond a bandwidth on demand service, communicating the network topology and reservable bandwidth is not necessary. However, for bigger networks in nodes and link capacity, a cross optimization increases the number of accepted jobs by 20%. These analytical results have been validated by means of computer simulations. The system's equivalent capacity determined in chapter 4 may facilitate the design of new cross optimizing heuristics. The results presented in this last chapter of the thesis are at the origin of one publication [116] and of one patent [115].

Future work on cross optimization could be done. Due to time constraints, we could not verify in a more dynamic environment that the Grid composed is equivalent to a queue of length the equivalent capacity and average service time the average job request processing time. Furthermore the structure coefficient that appear in the equivalent capacity dependence on the average node degree could be analyzed when multiple sources are involved. This thesis can be viewed as a first step towards cross-optimization in Grid environment.

Annexe A – Planned job request cross optimizer

The idea of the proposed algorithm is to be able to allocate network resources and computing resources in a way that minimize the completion time of each job. The main difference with the previous sections is that job request are planned job requests. For each job, computational and network resources will be used in the future. In other words, the algorithm proposed is a cross-optimizing algorithm for planned jobs requests. The algorithm takes as input a job request and processes each job request sequentially. For each job request, the algorithm selects network and computational resources and decides when they would be reserved. The algorithm supports several variants in order to broaden its applicability and improve its efficiency. The main idea is that within job requests constaints, it's possible to reduce the completion time by compensating the "power" of the network with the "power" of computational resources.

A job request is composed of a job description and job constraints, see diagram 13. Several job description are possible, the algorithm described below takes as input the following parameters. The algorithm can be extended to take into account additional constraints, see section 4.6. Most complex problems can be reduced to the algorithm but with additional constraints.

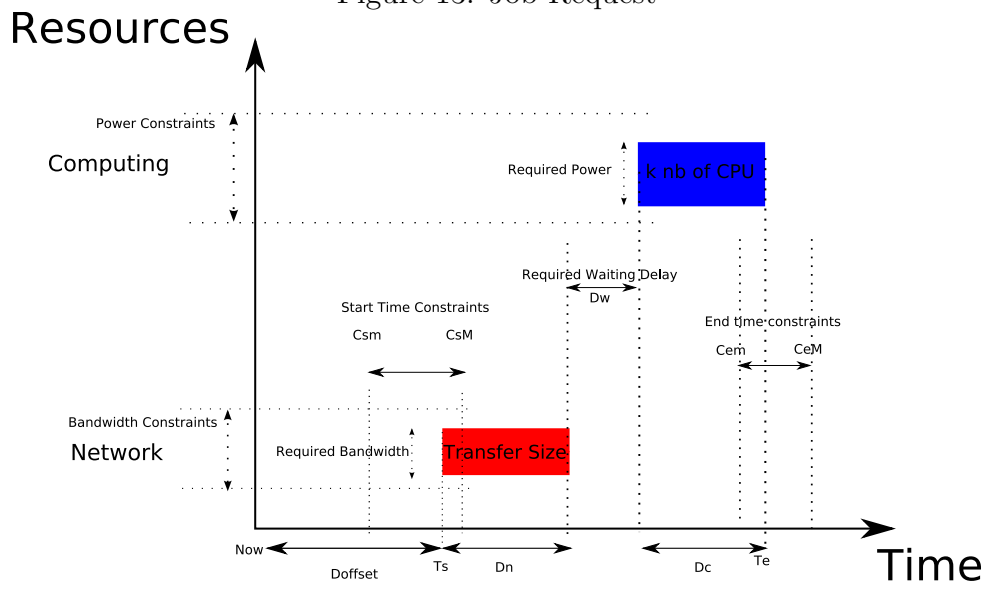
Input parameters

Diagram 13 uses the following convention, D_x for a duration corresponding to x, i.e. D_n is the duration of the network use, Ty for a time, i.e. T_s is the desired start time, C_{x_m} for a lower bound constraint (minimum), C_{x_M} for an upper bound constraints (Maximum).

The Job description contains (refer to fig 13):

- Source Node s
- Start Time Offset or Offset Duration, D_{offset}. Hence the minimum starting time

Figure 13: Job Request



would be $T_{s_m} = now + D_{offset}$

- Required Bandwidth BW_{req}
- Transfer Duration or D_n, the transfer time required if the bandwidth required is provided
- Required Waiting Delay or D_w, the delay the job must wait between the computing task and the transfer task
- Required Number of CPU or k, the number of CPU required by the job
- Required Processing power Pow_{req}
- Processing duration or D_c, the processing duration if the number of CPUs is provided on a machine of the required processing power

One of the variants describes how to handle a job description which does not require an explicit bandwidth but simply an amount of data to transfer.

The job constraints contains:

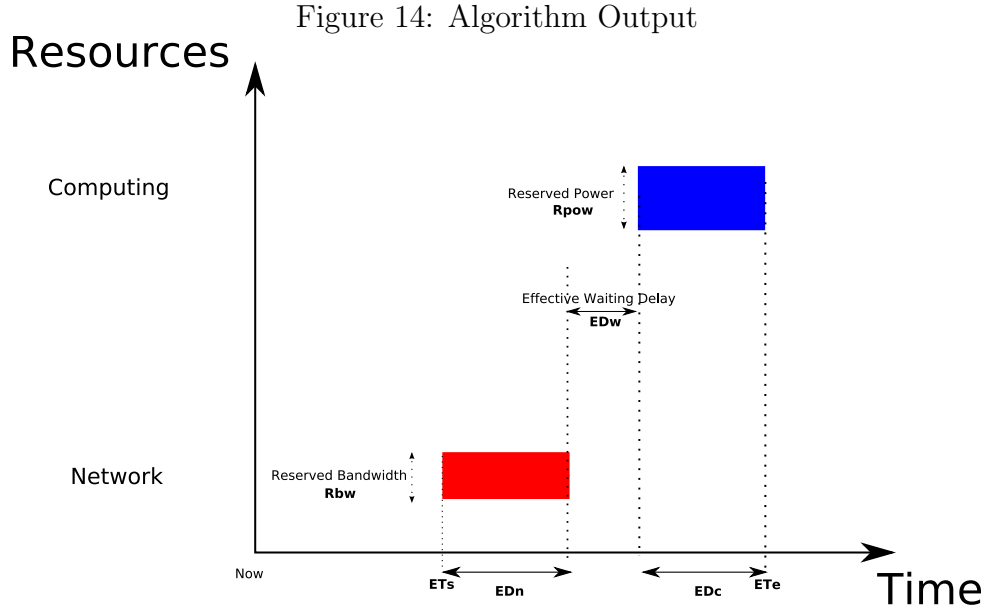
- Start Time Constraints T_{s_m} and T_{s_M}
- End Time Constraints or Completion Time Constraints, C_{e_m} and C_{e_M}
- Bandwidth Constraints or BW_m and BW_M
- Power Constraints or Pow_m and Pow_M

- Number of CPU constraints although today there is no use for such constraints, Maybe in the future such constraints would be used
- Delay constraints

Output parameters

The output parameters are given in bold in diagram 14. The letter E used as a prefix in variable name stands for Estimated or Effective time and the letter R for Reserved.

Figure 14: Algorithm Output



- Network resources identification
- Processing resources identification
- Reserved Bandwidth $BW_r = R_{bw}$
- Estimated start time E_{T_s}
- Effective transfer Duration E_{D_n}, n stands for network
- Effective Waiting Delay E_{D_w}
- Reserved Power R_{pow}
- Effective computation Duration E_{D_c}
- Estimated End Time E_{T_e}

Main Algorithm

The algorithm will process job requests one after the other, each job request has a description and constraints as in the paragraph above. For all jobs the algorithm returns a mapping job request - resources as described in the paragraph above. The algorithm cycles continuously whilst it waits to receive another job request. The algorithm maintains a sorted list $\mathcal{L}$ that contains all start and end times of all reservations on all resources. In other words, every instant a resource is reserved or freed is in $\mathcal{L}$. At the origin, $\mathcal{L}$ is empty, meaning that all resources are freed. For all elements t_i of $\mathcal{L}$, the algorithm keeps in memory a resource availability map $\mathcal{M}$ that contains the amount of available network and processing resources during the slot $[t_i, t_{i+1}]$ or $[t_i, \infty[$ (if there is just one element in $\mathcal{L}$. I.e. the amount of available bandwidth on each link and available number of CPU on each processing site and of the processing power unit.

Once resources have been selected, times E_{T_s}, $E_{T_s} + E_{D_n}$, $E_{T_s} + E_{D_n} + E_{D_w}$ and E_{T_e} must be inserted in $\mathcal{L}$ and the new amount of available resources at these times must be updated.

The algorithm Routes builder finds destinations and routes in a graph $\mathcal{G}'$ between a source node given in the job description and all the destinations. The initial destinations are all network nodes linked to a processing site.

If a processing site is able to move a job from one CPU to another (within its domain), it has no impact on this algorithm.

Architectural options

The proposed algorithm and variants (see below) can be implemented in at least two different kind of architectures. The first architecture communicates at frequent interval with resources to build and maintain $\mathcal{M}$, while the second architecture communicates when needed with the processing sites. For instance, instead of calling *CouldFinishAtSite*, the main algorithm could send a *CouldFinishAtSite* request to the processing site *site*. It would then be the responsability of the processing site to answer with the soonest processing starting time. Both architectures can be combined to manage processing sites that communicates resources availability and those that do not.

Algorithm 5: Planned job request cross-optimiser: First Transfer Then Compute

Require: A job request composed of its description and constraints $\mathcal{J} = (\mathcal{D}, \mathcal{C})$
Require: A network graph $\mathcal{G}$
Require: A list of Processing sites $\mathcal{P}$
Require: A list $\mathcal{L}$ of all start and end times of all reservations on all resources
1: $T_s \leftarrow now + D_{offset}$ {now is the current time}
2: Find the slot S_s of $\mathcal{L}$ that contains T_s
3: **repeat**
4: Remove from $\mathcal{G}$ links that don't satisfy bandwidth requirements for the required duration

$$\mathcal{G}' = TrimGraph(\mathcal{G}, T_s, T_s + D_n, BW_{req})$$

5: Let's build the list of routes towards potential destinations, starting from node s

$$(\mathcal{R}outes, \mathcal{D}est) \leftarrow RoutesBuilder(\mathcal{G}', \mathcal{J})$$

6: **if** $\mathcal{R}outes$ is Not empty **then**
7: **for all** route of $\mathcal{R}outes$ **do**
8: Let BW_r be the bandwidth reserved for this route {It can be different from BW_{req} in the variants of the algorithm}
9: **for all** Processing sites *site* attached to a destination of $\mathcal{D}est$ **do**
10: Compute the estimated termination time E_{T_e} if the job were executed on *site* using formula: $E_{T_e} = CouldFinishAtSite(T_s + D_n \frac{BW_{req}}{BW_r} + D_w, site, \mathcal{J})$
11: **end for**
12: **end for**
13: Return the route and the site whose estimated termination time E_{T_e} is the smallest satisfying constraints
14: **else** {$\mathcal{R}outes$ is empty}
15: Move S_s to the next slot, and Let T_s be the begining of the new slot
16: **end if**
17: **until** a start time is found or T_s is outside constraints {$T_s < T_{s_m}$ or $T_s > T_{s_M}$}

Variants

Bandwidth constraints

One of the main variants of the algorithm is to be able to find resources when the job request has bandwidth constraints. I.e. A bandwidth BW_{req} is specified but also BW_m and BW_M. In this situation, line 4 of algorithm 5 must be replaced by:

Let $\mathcal{G}' = TrimGraph(\mathcal{G}, T_s, T_s + D_n \frac{BW_{req}}{BW_m}, BW_m)$

Algorithm 6: Routes Builder

Require: A set $\mathcal{D}est$ of potential destinations {In the worst case $\mathcal{D}est$ contains all destinations}
Require: A network graph $\mathcal{G}'$
Require: A job request $\mathcal{J}$
1: Let $\mathcal{R}outes$ the list of potential routes assotiated to these destinations
2: **for all** Destination d of $\mathcal{D}est$ **do**
3: Add to $\mathcal{R}outes$ one route among the routes of maximum bandwidth between s and d in $\mathcal{G}'$
4: **if** No route is found between s and d **then**
5: Remove d from $\mathcal{D}est$
6: **end if**
7: **end for**
8: Returns $(\mathcal{R}outes, \mathcal{D}est)$

Algorithm 7: CouldFinishAtSite

Require: A processing site *site*
Require: A job request $\mathcal{J}$
Require: A minimum starting time T_s { The input parameter T_s should be used instead of information derived from $\mathcal{J}$}
1: Let Pow be the processing power of the CPUs of *site*
2: Find the slot S_{c_s} of $\mathcal{L}$ that contains T_s
3: **repeat**
4: Find the first element t_k of $\mathcal{L}$ bigger or equal than $E_{T_e} = T_s + D_c \frac{Pow_{req}}{Pow}$
5: Let S_{c_e} be the slot that contains t_k and E_{T_e} or the slot just before t_k if they are equal
6: **if** For all Slots between S_{c_s} and S_{c_e} The number of CPU available is bigger than the number required by $\mathcal{J}$ **then**
7: Returns E_{T_e}
8: **end if**
9: Make S_{c_s} be the next slot in $\mathcal{L}$, and Let T_s be the begining of that slot
10: **until** a start time is found or T_s is outside constraints $\{T_s < T_{s_m}$ or $T_s > T_{s_M}\}$

Of course, in line 3 of algorithm 6, the selected route's bandwidth must not exceed BW_M. If that is the case, the reserved bandwidth on this route will be $BW_r = BW_M$.

Transfer x Mbit rather than get y Mbit/s

The algorithm could be further extended to take into account job requests that do not specify a bandwidth requirement but simply an amount of data to transfer $Q_{transfer}$. Said differently, the algorithm can be extended to handle elastic job requests. An elastic

Algorithm 8: TrimGraph

Require: A network graph $\mathcal{G}$
Require: A starting time T_s and a termination time T_e
Require: A trimming threshold BW_{trim}
1: Find the slot S_s of $\mathcal{L}$ that contains T_s
2: Let S_{n_e} be the slot that contains T_e or the last slot
3: Create a graph $\mathcal{G}' \leftarrow \mathcal{G}$
4: **for all** slots between S_s and S_{n_e} and links **do**
5: **if** Available bandwidth on a link is lower than BW_{trim} **then**
6: Remove the link from $\mathcal{G}'$
7: **end if**
8: **end for**
9: Return $\mathcal{G}'$

job request is a job request without specifying BW_{req} and bandwidth constraints. The following argues that designing a planned job request scheduler for elastic job requests is equivalent to the bandwidth constrained problem.

Indeed, all jobs can not be transfered beyond the access rate of their originating network, which gives an upper bound to the bandwidth. A lower bound to the bandwidth is given by the job time constraints. The idea is to be able to estimate the transfer termination time constraint C_{t_M}. Starting from the job termination time constraint, C_{e_M} and doing a retro planning such that whatever the processing resource selected, the termination time constraint will be satisfied. To do so, we have to assume the slowest processing resource is selected, let $Pow_{slowest}$ be its processing power. Then,

$$C_{t_M} = C_{e_M} - \frac{Pow_{req}}{Pow_{slowest}} D_c - D_w$$

And the minimum bandwidth required to finish the transfer at C_{t_M} is:

$$BW_m = \frac{Q_{transfer}}{C_{t_M} - T_{s_m}}$$

Of course, as time elapses, the minimum bandwidth required to finish the transfer increases. This is what happens during the execution of the algorithm, as the algorithm walks through $\mathcal{L}$, the required bandwidth to respect the deadline increases.

So elastic job requests can be handled by algorithm 9. The purpose of this algorithm is to create an equivalent job request based on the elastic job request modifying BW_{req} and job constraints, as explained before.

Algorithm 9: Planned Elastic Job request cross-optimizer

Require: An elastic job request $\mathcal{J}$

Require: Same input as algorithm 5 "*Planned job request cross-optimizer*"

1: Let $Pow_{slowest}$ be the processing power of the slowest computational resource

2: Compute the termination time constraint C_{t_M} with:

$$C_{t_M} = C_{e_M} - \frac{Pow_{req}}{Pow_{slowest}} D_c - D_w$$

3: Create a new job $\mathcal{J}' = \mathcal{J}$

4: Modify $\mathcal{J}'$'s minimum bandwidth required with:

$$BW_m(\mathcal{J}') = \frac{Q_{transfer}}{C_{t_M} - T_{s_m}}$$

5: Let $BW_{req}(\mathcal{J}') = BW_m(\mathcal{J}')$

6: Call algorithm 5 with $\mathcal{J}'$ for input

Compute first, transfer second

The job described in diagram 13 is composed of two tasks: first a transfer task followed by a computing task. If those tasks were inversed, i.e. first a computing task followed by a transfer task, a variant (see algorithm 10) of algorithm 5 should be used to provide the cross optimisation. First the job description is slightly changed to include a destination site d. Instead of trying to find first potential routes and then computing the estimated job termination time, the algorithm should try to estimate when the processing task can be terminated as soon as possible and then see when the network can terminate the transfer.

Of course, if we want to take into account bandwidth constraints (and elastic job requests), line 3 of algorithm 11 must be modified like line 4 of algorithm 5.

How to take into account the number of CPU constraints?

If the job requires a number of CPUs, the function *CouldFinishAtSite* must be modified to take into account a filtering based on the number of CPUs. To do it, the map $\mathcal{M}$ has to take into account the number of available CPUs.

Algorithm 10: Planned job request cross-optimiser: First Compute Then Transfer

Require: A job request composed of its description and constraints $\mathcal{J} = (\mathcal{D}, \mathcal{C})$
Require: A network graph $\mathcal{G}$
Require: A list of Processing sites $\mathcal{P}$
Require: A list $\mathcal{L}$ of all start and end times of all reservations on all resources
1: $T_s \leftarrow now + D_{offset}$ {now is the current time}
2: **for all** Processing sites $site$ **do**
3: Compute the estimated computing termination time $E_{T_{ce}}$ if the job were executed on $site$:

$$E_{T_{ce}(site)} = CouldFinishAtSite(T_s, site, \mathcal{J})$$

4: Compute the estimated termination time E_{T_e} and memorize the route used for the transfer

$$(E_{T_{ce}}(site), Route(site)) = CouldFinishTransfer(E_{T_{ce}}(site) + D_w, s, d, \mathcal{J}, \mathcal{G})$$

5: **end for**
6: **return** the processing site and the route that have the smallest $E_{T_{ce}}(site)$

Algorithm 11: CouldFinishTransfer

Require: A start time T_s
Require: A source and destination node pair (s, d)
Require: A job $\mathcal{J}$
Require: A graph $\mathcal{G}$
1: Find the slot S_s of $\mathcal{L}$ that contains T_s
2: **repeat**
3: Let $\mathcal{G}' = TrimGraph(\mathcal{G}, T_s, T_s + D_n, BW_{req})$
4: In the graph $\mathcal{G}'$ find a route between s and d using a maximum bandwidth route algorithm
5: **if** A route is found **then**
6: Let BW_r be the reserved bandwidth on the route
7: Return $T_s + D_n \frac{BW_{req}}{BW_r}$ and the route selected
8: **else**
9: Move S_s to the next slot, and Let T_s be the beginning of the new slot
10: **end if**
11: **until** a start time is found or T_s is outside constraints $\{T_s < T_{s_m}$ or $T_s > T_{s_M}\}$

How to take into account delay constraints?

Once the transfer is realized, the job has to wait for D_w on the processing site. The first thing to do it to make sure the processing site has sufficient local storage capability for

the waiting duration plus the processing duration, i.e.

$$D_w + D_c \frac{Pow_{req}}{Pow_r}$$

Additional storage requirement could also be taken into account in the job description.

How to minimize the search time inside $\mathcal{L}$?

$\mathcal{L}$ could be split in several lists, one for each resource, hence reducing the number of entries per individual list. These lists should be stored as hashtables to reduce the search time.

How to improve the list of potential destinations?

First the list of potential destination is the list of all destinations attached to a processing site.

How to customize the routing algorithm?

This route can be either, the shortest path between s and d, satisfying BW_{req} or the maximum bandwidth route between s and d or any route satisfying at least BW_{req} selected taking into account load-balancing constraints. Such load-balancing constraints could be used in the following way: selecting a random route of bandwidth BW_{req} among routes satisfying at least BW_{req}, or selecting a random route of a random bandwidth between bandwidth constraints among routes satisfying bandwidth constraints. To do so, line 3 must be replaced by

> Add to $\mathcal{R}outes$ the routes selected by the Traffic Engineering routing algorithm (TE-Algo)

Such TE-Algo is not specified here.

How to improve algorithm speed when $\mathcal{R}outes$ is empty?

When $\mathcal{R}outes$ is empty, it means the network is not connected anymore. The next T_s time to investigate is the end time of the slot that makes the network graph between the source s and all potential destinations not connected.

How to take into account path switching capability?

Already described in [56].

Annexe B – Simulation tools

The simulator to evaluate and compare the performance of various algorithms is composed of several programs. Most programs were written in C++, several shell scripts, several VBA programs and a few XSLT programs. All C++ programs are multi-platform, they were compiled with Eclipse and MS VC++, they run on both platforms.

TopologyGenerator

The first program is a topology generator. Since we are working with random graphs, we need a random topology generator. The topology will be composed of the network graph, several jobs and clusters. The program takes model parameters as input and writes one or several topologies according to the model. The output is one or several XML files. The data model for the XML file is represented in diagram 15. An XML document is automatically parsed by a DOM parser and automatically represented as an C++ class using an XML-C++ data binding tool [5].

The simulator generates a waxman graph but it excludes all graphs that are not connected [109].

CreateMapping

CreateMapping is a program that takes as input a topology and computes the different mapping, i.e. the resource allocation for each algorithm mentioned in chapter 4. A mapping has the following XML data model. To each job, it associates a route. The route number is the same as the job number. I.e. route 92 is the route used by job 92. The mapping data model is shown in diagram 16.

Diagram 17 shows an example simulation result file.

We used C++ programming techniques like templates and some design patterns [76].

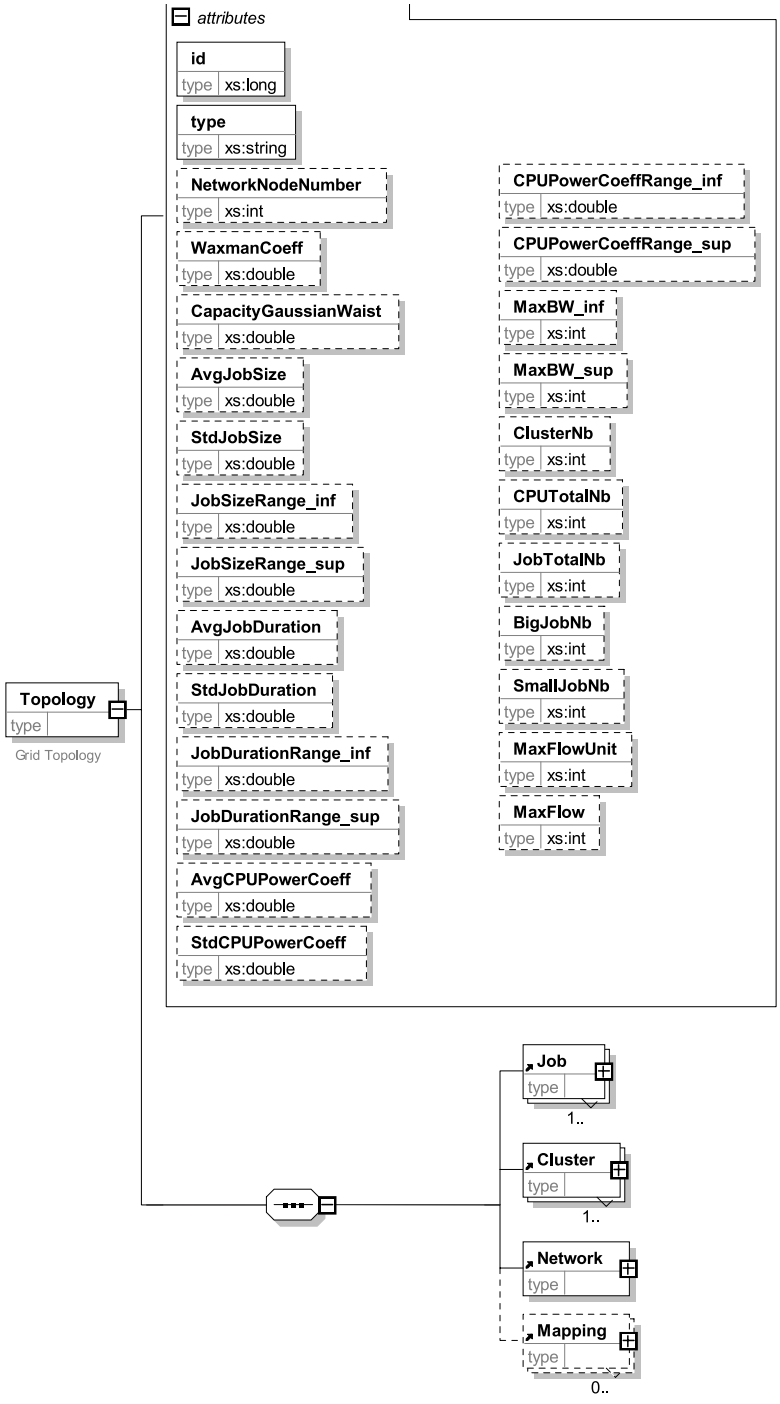


Figure 15: Topology model

We used four templates one for each algorithm: *XO*, Connection with CranckBack, Connection without CranckBack, LegacyMaxBW and LegacyMinHop. The template parameter is an iterator describing how to walk through the jobs.

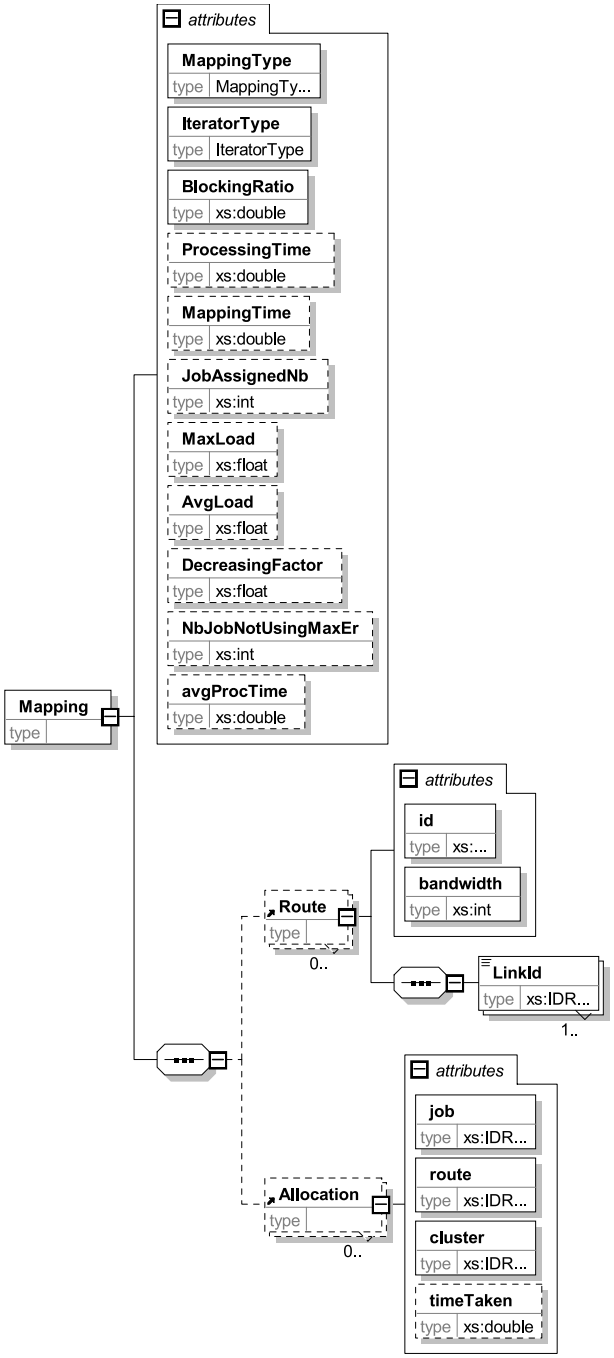


Figure 16: Mapping output model

Topology Visualization tool

We wrote a simple XSLT stylesheet to transform our XML file representing our graphs to convert it to a GraphML compliant file. This way, we could automatically visualize the topology generated with any GraphML editor.

Mapping (45)

	MappingType	IteratorType	ProcessingTime	MappingTime	JobAssignedNb	MaxLoad	AvgLoad	DecreasingFactor	Route	Allocation
1	XO	EgoistFCFS	281565.97157042	0.02	56				Route (56)	Allocation (56)
2	XO	IncreasingDuration	248083.5224639	0.01	52				Route (52)	Allocation (52)
3	XO	DecreasingDuration	303079.190164	0.02	52				Route (52)	Allocation (52)
4	XO	IncreasingSize	315611.60807049	0.03	53				Route (53)	Allocation (53)
5	XO	DecreasingSize	262656.49039969	0.02	52				Route (52)	Allocation (52)
6	XO	IncreasingDominance	273811.52794457	0.03	54				Route (54)	Allocation (54)
7	XO	DecreasingDominance	273601.29539034	0.02	54				Route (54)	Allocation (54)
8	XO	MHLDominance	266143.0108208	0.03	53				Route (53)	Allocation (53)
9	XO	MLHDominance	266143.0108208	0.02	53				Route (53)	Allocation (53)
10	LegacyMinHop	EgoistFCFS	521646.11355009	0.01	100	0.6005	0.13925	0.8	Route (100)	Allocation (100)
11	LegacyMinHop	IncreasingDuration	507785.74607812	0.02	100	0.99375	0.25372	0.8	Route (100)	Allocation (100)
12	LegacyMinHop	DecreasingDuration	509830.4199685	0.01	100	0.9957	0.2237475	0.8	Route (100)	Allocation (100)
13	LegacyMinHop	IncreasingSize	507174.15597975	0.01	100	0.98095	0.2381662	0.8	Route (100)	Allocation (100)
14	LegacyMinHop	DecreasingSize	511684.64094836	0.01	100	0.89695	0.21233	0.8	Route (100)	Allocation (100)
15	LegacyMinHop	IncreasingDominance	518532.68806186	0.02	100	0.87485	0.1692587	0.8	Route (100)	Allocation (100)
16	LegacyMinHop	DecreasingDominance	516633.39648263	0.02	100	0.8088	0.1716013	0.8	Route (100)	Allocation (100)
17	LegacyMinHop	MHLDominance	516829.40362142	0.02	100	0.63985	0.147125	0.8	Route (100)	Allocation (100)
18	LegacyMinHop	MLHDominance	516829.40362142	0.02	100	0.63985	0.147125	0.8	Route (100)	Allocation (100)
19	LegacyMaxBW	EgoistFCFS	523361.32406082	0.01	100	0.58515	0.1238763	0.8	Route (100)	Allocation (100)

Figure 17: Simulation result file example

Data Aggregation & Analysis

We decided not to embed a stop test in our simulation to get our results but to compute confidence interval a posteriori. That is to say, we simulated a fixed number of topologies, then we computed confidence intervals, and we iterated when needed.

Bibliography

[1] Amazon elastic compute cloud. aws.amazon.com/ec2. (document), 1.2.1

[2] Amazon elastic compute cloud api reference. http://docs.amazonwebservices.com/AmazonEC2/gsg/2007-01-19/. (document), 1.2.1

[3] Canet 4. http://www.canarie.ca/canet4. 1.5.5

[4] Cluster computing on demand. http://clusterondemand.com. (document), 1.2.1

[5] Codalogic lmx. www.tech-know-ware.com/lmx/. 4.6

[6] Datasynapse. http://www.datasynapse.com. (document), 1.4.1

[7] Gara project. http://datatag.web.cern.ch/datatag/. (document), 1.5, 1.5.3

[8] Generalized multi-protocol label switching (gmpls) architecture. Technical report. 1.5

[9] Global file system. http://en.wikipedia.org/wiki/Global_File_System. 1.4.1

[10] Global file systems. http://en.wikipedia.org/wiki/Global_filesystem. 1.4.1

[11] Globus project. http://www.globus.org. 1.5.3

[12] Grid resource scheduler project. http://www.cs.ucl.ac.uk/staff/S.Bhatti/grs/index.html. (document), 1.5, 1.5.2

[13] Gridway. http://www.gridway.org/. 1.4.2

[14] Ip multimedia subsystem (ims); stage 2. TS 23.228. (document), 2.5

[15] List of file systems. http://en.wikipedia.org/wiki/List_of_file_systems. 1.4.1

[16] Nortel drac. http://www.nortel.com/drac. (document), 1.5, 1.5.12

[17] Open grid forum. http://www.ogf.org. 1.4.3

[18] Platform lsf. http://www.platform.com. (document), 1.4.1

[19] Portable batch system. http://www.pbsgridworks.com. (document), 1.4.1

[20] Qbone bandwdith broker. http://qbone.internet2.edu/bb/bboutline2.html. (document), 1.5, 1.5.6

[21] Service level agreement. http://en.wikipedia.org/wiki/Service_Level_Agreement. (document)

[22] Seti@home. http://setiathome.berkeley.edu/. (document)

[23] Sun grid compute utility. http://www.network.com/. (document), 1.2.1

[24] Sun n1 grid engine. http://www.sun.com/software/gridware/. (document), 1.4.1

[25] Third generation partnership project. http://www.3gpp.org/. (document), 2.5

[26] Tivoli workload scheduler loadleveler. http://www-03.ibm.com/systems/clusters/software/loadleveler.html. (document), 1.4.1

[27] User controlled light paths (uclp). http://www.canarie.ca/canet4/uclp/. (document), 1.5, 1.5.5

[28] Viola. http://www.viola-testbed.de/. (document), 1.5, 1.5.13

[29] Network resource reservation software and interfaces: a state of the art survey. EU Deliverable DJRA4.1 Addendum, EGEE, 2004. 1.5

[30] Open service access (osa); parlay x web services. ES 202-391, March 2005. 2.5.1

[31] Amps - design specification. Technical Report GN2-04-153v4, GEANT2, 2006. 1.5.7

[32] End-to-end specification for bandwidth allocation and reservation. EU Deliverable Draft MJRA4.5, EGEE, 2006. (document), 1.5, 1.5.7

[33] Specification of interfaces for bandwidth reservation service. EU Deliverable DJRA4.1, EGEE, 2006. 1.5, 1.5.7, 1.5.7

[34] Grid optical user network interface (g.ouni). GHPN Draft version 0.4, 2007. (document), 2.4

[35] A path computation element (pce)-based architecture. RFC 4655, August 2006. (document), 2.3.2

[36] Ason/gmpls extension for reservation and time based automatic bandwidth service. expired draft draft-yong-ccamp-ason-gmpls-autobw-service-00.txt, November 2006. (document), 2.3

[37] I. Akogrimo. Project. Akogrimo– Access to Knowledge through the Grid in a mobile World. http://www.mobilegrids.org. (document), 1.5, 1.5.9

[38] A. Amoroso and K. Marzullo. Multiple job scheduling in a connection-limited data parallel system. *Parallel and Distributed Systems, IEEE Transactions on*, 17(2):125–134, 2006. 4.2.2

[39] D. Anderson. BOINC: a system for public-resource computing and storage. *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10, 2004. (document)

[40] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002. (document)

[41] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). GWD-R (Proposed Recommendation), Open Grid Forum, 2007. (document), 3.1.2, 3.2.3

[42] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) Specification V1. 0. *Grid Forum Document GFD*, 56, 2005. 1.4.5, 3.3.1, 4.3

[43] T. Banks. Web Services Resource Framework (WSRF) - v1. 2. *OASIS Specification*, 2006. 1.4.2

[44] C. Barz, F. Hommes, W. Moll, M. Pilz, C. Rosche, and J. Schon. *ARGON - Allocation and reservation in Grid-enabled optic networks, VIOLA Bericht B2.4.1.* Bonn University, Bonn, Germany, August 2005. 2.2.3

[45] L. Battestilli, A. Hutanu, G. Karmous-Edwards, D. S. Katz, J. MacLaren, J. Mambretti, J. H. Moore, S.-J. Park, H. G. Perros, K. Syam Sundar, S. Tanwir, S. R. Thorpe, and Y. Xin. Enlightened computing: An architecture for co-scheduling and co-allocating network, comput, and other grid resources for high-end applications. (document), 1.5, 1.5.14

[46] L. Berger et al. Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions. Technical report, RFC 3473, January 2003, 2003. 1.5.1

[47] B. Bhargava. *Concurrency and Reliability in Distributed Database Systems*. Van Nostrand Reinhold, 1987. (document), 3.1.3

[48] B. Bollobás. *Random Graphs*. Cambridge University Press, 2001. (document), 4.3

[49] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocl (SOAP) 1.1. (document), 1.3

[50] C. Briquet and P.-A. de Marneffe. Grid resource negotiation: survey with a machine learning perspective. In *PDCN'06: Proceedings of the 24th IASTED international conference on Parallel and distributed computing and networks*, pages 17–22, Anaheim, CA, USA, 2006. ACTA Press. (document), 3.1

[51] R. Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing, PhD Thesis.* Monash University, Melbourne, Australia, 2002. (document), 1.2.2, 3.1

[52] F. Callegati, W. Cerroni, A. Campi, G. Zervas, R. Nejabati, and D. Simeonidou. Application Aware Optical Burst Switching Test-bed with SIP Based Session Control. *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on*, pages 1–6, 2007. (document), 2.5

[53] A. Campi. Sip for grid networks, presentation, 2007. (document), 2.5

[54] A. Campi, W. Cerroni, F. Callegati, G. Zervas, R. Nejabati, and D. Simeonidou. SIP Based OBS Networks for Grid Computing. *Conference on Optical Network Design and Modelling, ONDM, Athens, Greece, May*, 2007. (document), 2.5

[55] S. Cantor, I. Kemp, N. Philpott, and E. Maler. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2. 0. *Committee Draft*, 4:14, 2005. 1.4.2

[56] R. Cohan, N. Fazlollahi, and D. Starobinski. Graded channel reservation with path switching in ultra high capacity networks. In *GridNets 2006. 3rd international workshop on networks for grid applications. Broadnets Proceedings. IEEE*, 2006. 4.6

[57] C. Curti, T. Ferrari, L. Gommans, S. van Oudenaarde, E. Ronchieri, F. Giacomini, and C. Vistoli. On advance reservation of heterogeneous network paths. *Future Generation Computer Systems*, 21(4):525–538, 2005. (document), 1.5, 1.5.4

[58] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. Snap : A protocol for negotiation of service level agreements and coordinated resource management in distributed systems. In *Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing, Edinburgh, Scotland*, July 2002. (document), 3.1.2

[59] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. *8th Workshop on Job Scheduling Strategies for Parallel Processing*, 2002. (document)

[60] R. Das, J. Hanson, J. Kephart, and G. Tesauro. Agent-human interactions in the continuous double auction. *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), August*, pages 4–10, 2001. 3.1.1

[61] J. Dongarra. The LINPACK Benchmark: An Explanation. *Proceedings of the 1st International Conference on Supercomputing*, pages 456–474, 1987. (document), 4.3

[62] J. Dongarra, P. Luszczek, and A. Petitet. The LINPACK Benchmark: past, present and future. *Concurrency and Computation Practice and Experience*, 15(9):803–820, 2003. (document), 4.3

[63] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. Stewart. *LINPACK Users' Guide*. 1979. (document), 4.3

[64] P. Erdos and A. Renyi. On random graphs. *Publ. Math. Debrecen*, 6(290), 1959. (document), 4.3

[65] D. Erwin. Unicore: a grid computing environment. *Concurrency and Computation: Practice and Experience*, 14(13-15):1395–1410, 2002. (document), 1.4.3, 1.4.3

[66] D. Erwin. UNICORE Plus Final Report-Uniform Interface to Computing Resources. *UNICORE Forum eV*, pages 3–00, 2003. (document), 1.4.3, 1.5.13

[67] R. Fielding. *Representational state transfer (REST). Chapter 5 in Architectural Styles and the Design of Networkbased Software Architectures*. PhD thesis, Ph. D. Thesis, University of California, Irvine, CA, 2000. (document), 1.3

[68] L. Ford and D. Fulkerson. *Flows in networks*. Princeton University Press Princeton, NJ, 1962. 4.2.2

[69] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006. (document), 1.4.2, 1.5.3

[70] I. Foster and C. Kesselman. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1998. (document)

[71] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supportsadvance reservations and co-allocation. *Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on*, pages 27–36, 1999. (document), 1.5, 1.5.3

[72] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum, June*, 22:2002, 2002. (document)

[73] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200, 2001. (document)

[74] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, et al. The Open Grid Services Architecture. *Global Grid Forum, GFD-I.030*. (document), 1.4.5

[75] R. Gallagher. *Discrete Stochastic Processes*. Springer, 1996. (document), 4.4

[76] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series)*. Addison-Wesley Professional, 1995. (document), 1.5.13, 4.6

[77] M. Gondran and M. Minoux. Graphes et algorithmes. *Collection de la Direction des Études et Recherches dŠÉlectricité de France,*, 1995. 4.2.2

[78] T. Goss-Walter, R. Letz, T. Kentemich, H. Hoppe, and P. Wieder. An Analysis of the UNICORE Security Model. *Global Grid Forum*. 1.4.3

[79] J. Gray and L. Lamport. Consensus on transaction commit. *ACM Transactions on Database Systems (TODS)*, 31(1):133–160, 2006. 1.5.14

[80] L. Green. Service level negotiation in a heterogeneous telecommunication environment. In *Proceeding International Conference on Computing, Communications and Control Technologies (CCCT04), Austin, TX, USA*, August 2004. (document), 3.1.2

[81] C. Groves, M. Pantaleo, T. Anderson, and T. Taylor. RFC3525: Gateway Control Protocol Version 1. *Internet RFCs*, 2003. 2.5.1

[82] R. Guerin and A. Orda. Networks with advance reservations: the routing perspective. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 118–127 vol.1, 2000. (document)

[83] A. Gulbrandsen, P. Vixie, and L. Esibov. RFC2782: A DNS RR for specifying the location of services (DNS SRV). *Internet RFCs*, 2000. 2.5.1

[84] V. Gurbani, L. Jagadeesan, and V. Mendiratta. Characterizing session initiation protocol (sip) network performance and reliability. In *Proceedings of International service availability symposium*, April 2005. (document), 3.3

[85] M. Handley and V. Jacobson. SDP: Session Description Protocol, RFC 2327. *Internet Engineering Task Force*, 1998. (document), 2.5.1

[86] E. He, X. Wang, and J. Leigh. A Flexible Advance Reservation Model for Multi-Domain WDM Optical Networks. *IEEE GRIDNETS*, 2006. (document)

[87] J. C. Hull. *Options, Futures and Other Derivatives (6th Edition)*. Prentice Hall, 2005. 3.1.1

[88] N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2), March 2001. (document), 3.1.2

[89] S. Kannan, M. Roberts, P. Mayes, D. Brelsford, and J. Skovira. Workload management with loadleveler. *IBM International Technical Support Organization, November*, 2001. (document), 1.4.1

[90] D. Katz, K. Kompella, and D. Yeung. RFC3630: Traffic Engineering (TE) Extensions to OSPF Version 2. *Internet RFCs*, 2003. 1.5.1

[91] A. Keller and H. Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1):57–81, 2003. (document)

[92] W. Kohler. A survey of techniques for synchronization and recovery in decentralized computer systems. *ACM Computing Surveys*, 13(2), June 1981. (document), 3.1.3

[93] F. Kuipers. *Quality of Service Routing in the Internet: Theory, Complexity and Algorithms*. Delft University Press, 2004. 4.3

[94] Y. Kulbak and D. Bickson. The eMule Protocol Specification. *eMule project, http://sourceforge.net*. (document)

[95] D. Kuo, M. Parkin, and J. Brooke. A framework & negotiation protocol for service contracts. In *Proceedings of the 2006 IEEE International Conference on Services Computing (SCC 2006)*, pages 253–256, 2006. (document), 3.1

[96] D. Kuo, M. Parkin, and J. Brooke. Negotiating contracts on the grid. In *Exploiting the Knowledge Economy - Issues, Applications, Case Studies, Volume 3, Proceedings of the eChallenges 2006 (e-2006) Conference*. IOS Press, 2006. (document), 3.1

[97] L. Lamport. Paxos made simple. *ACM SIGACT News*, 32(4):18–25, 2001. 1.5.14

[98] D. Lee and H. Rieger. Maximum flow and topological structure of complex networks. *EUROPHYSICS LETTERS*, 73(3):471–477, 2006. (document), 4.4, 4.5.5

[99] T. Lehman, J. Sobieski, and B. Jabbari. DRAGON: A framework for service provisioning in heterogeneous grid networks. *Communications Magazine, IEEE*, 44(3):84–90, 2006. (document), 1.5, 1.5.8

[100] M. Lei, S. Vrbsky, and X. Hong. A dynamic data grid replication strategy to minimize the data missed. In *GridNets 2006. 3rd international workshop on networks for grid applications. Broadnets Proceedings. IEEE*, 2006. 4.2

[101] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. *IBM Corporation*, 2002. (document)

[102] J. MacLaren, B. Rouge, and M. Mc Keown. HARC: A Highly-Available Robust Co-scheduler. *e-print http://www.realitygrid.org/publications/HARC.pdf*. 1.5.14

[103] T. Magedanz, D. Witaszek, and K. Knuettel. The IMS Playground@ Fokus-An Open Testebed for Next Generation Network Multimedia Services. *Proceedings of First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM'05), IEEE Computer Society, Feb*, pages 2–11, 2005. 2.5.1

[104] G. Markidis, A. Tzanakaki, N. Ciulli, G. Carrozzo, D. Simeonidou, R. Nejabati, and G. Zervas. EU Integrated Project PHOSPHORUS: Grid-GMPLS Control Plane for the Support of Grid Network Services. *Transparent Optical Networks, 2007. ICTON'07. 9th International Conference on*, 3, 2007. (document), 1.5, 1.5.11

[105] D. Menasce and E. Casalicchio. A framework for resource allocation in grid computing. *Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on*, pages 259–267, 2004. (document)

[106] D. Menasce and E. Casalicchio. QoS in grid computing. *Internet Computing, IEEE*, 8(4):85–87, 2004. (document)

[107] R. Menday and P. Wieder. GRIP: The Evolution of UNICORE towards a Service-Oriented Grid. *Proc. of the 3rd Cracow Grid Workshop (CGWŠ03)*, pages 142–150, 2003. 1.4.3

[108] R. Montero, E. Huedo, and I. Llorente. Grid Scheduling Infrastructures based on the GridWay Meta-scheduler. *TCSC Newsletter*, 8(2), 2006. 1.4.2, 2.2.1

[109] M. Naldi. Connectivity of Waxman topology models. *Computer Communications*, 29(1):24–31, 2005. (document), 4.3, 4.6

[110] M. Oszu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1991. (document), 3.1.3

[111] A. Pichot. Dispositif de contrôle de l'établissement de sessions. Technical report, European Patent Office. (document), 4.6, 4.6

[112] A. Pichot. Elément de type pce pour le calcul de chemins de connexion previsionnels dans un réseau de transport. Technical report, INPI. (document), 4.6, 4.6

[113] A. Pichot. Elément de type pce pour le calcul de chemins de connexion previsionnels dans un réseau de transport. Technical report, World Intellectual Property Organization. 4.6

[114] A. Pichot. Noeud de réseau de transport, à adjonction de données temporelles à des données d'ingénierie de trafic. Technical report, European Patent Office. (document), 4.6, 4.6

[115] A. Pichot. Procédé d'allocation de ressources pour un logiciel de gestion de ressources distribuées. Technical report, INPI. (document), 4.6, 4.6

[116] A. Pichot. Co-allocation & cross optimization of network and computing resources for distributed applications. *to submit*, 2008. (document), 4.6, 4.6

[117] A. Pichot and O. Audouin. Grid services over IP Multimedia Subsystem. *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–7, 2006. (document), 4.6, 4.6

[118] A. Pichot and O. Audouin. Grid over ip multimedia subsystem. *Business Models and Drivers for Next-Generation IMS Services*, 2007. (document), 4.6, 4.6

[119] A. Pichot, P. Wieder, O. Wäldrich, and W. Ziegler. Dynamic sla-negotiation based on ws-agreement. Technical Report, TR 82, CoreGrid, 2007. (document), 3.5, 4.6, 4.6

[120] A. Pichot, P. Wieder, O. Wäldrich, and W. Ziegler. Dynamic sla negotiation based on ws-agreement. In *Proceedings of the 4th International conference on Web Information Systems and Technologies (WEBIST 2008)*, May 2008. (document), 3.5, 4.6, 4.6

[121] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005. (document)

[122] M. Rambadt and P. Wieder. UNICORE–Globus Interoperability: Getting the Best of Both Worlds. *Proc. of*, 11. (document), 1.4.6

[123] M. Rambadt and P. Wieder. UNICORE–Globus: Interoperability of Grid Infrastructures. *Cray User Group Summit 2002 Proceedings*, 2002. (document), 1.4.6

[124] W. Reinhardt. Advance Resource Reservation and its Impact on Reservation Protocols. *Proceedings of Broadband Island*, 95:28–35, 1995. (document)

[125] Y. Rekhter, T. Li, et al. A Border Gateway Protocol 4 (BGP-4). Technical report, RFC 1771, March 1995. 1.5.1

[126] K. Reynolds. The Double Auction. *Agorics, Inc*, 1996. 3.1.1

[127] A. Roach. RFC 3265: Session Initiation Protocol (SIP)-Specific Event Notification. dynamicsoft, June 2002. *Standards Track*. (document), 2.5.2

[128] O. Rodeh and A. Teperman. zFS-a scalable distributed file system using object disks. *Mass Storage Systems and Technologies, 2003.(MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*, pages 207–218, 2003. 1.4.1

[129] J. Rosenberg and H. Schulzrinne. RFC3264: An Offer/Answer Model with Session Description Protocol (SDP). *Internet RFCs*, 2002. (document), 2.5.2

[130] J. Rosenberg, H. Schulzrinne, and P. Kyzivat. Caller Preferences for the Session Initiation Protocol (SIP) RFC 3841. *IETF, Aug*, 2004. (document), 2.5.2

[131] J. Rosenberg, H. Schulzrinne, and P. Kyzivat. Indicating user agent capabilities in the session initiation protocol (SIP). Internet Engineering Task Force. Technical report, RFC 3840 (Aug.), 2004. (document), 2.5.2

[132] L. Sadeghioon, R. Nejabati, and D. Simeonidou. GMPLS Extensions for a User-Centric and Grid Enabled Optical Network Control Plane. *Transparent Optical Networks, 2006 International Conference on*, 3, 2006. (document), 2.4

[133] S. Salsano. Sip based qos negotiation protocol. *Online proceedings of EEQoSŠ05*, 2005. (document), 1.5, 1.5.10

[134] V. Sander. Networking issues for grid infrastructure. GFD 37, 2004. (document)

[135] V. Sander. Networking issues for grid infrastructure. GFD 37, November 2004. (document), 2.1.2, 2.4

[136] J. M. Schopf. *Ten actions when Grid scheduling: the user as a Grid scheduler*, pages 15–23. Kluwer Academic Publishers, Norwell, MA, USA, 2004. (document), 3

[137] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002. 4.2.2

[138] A. Schrijver. *Combinatorial Optimization, Polyhedra and Efficiency*. Springer-Verlag, 2003. 4.2.2

[139] M. Shakun, editor. *Group Decision and Negotiation*. Springer Netherlands, 2002. (document), 3.1

[140] W. Shen, H. H. Ghenniwa, and C. Wang. Adaptive negotiation for agent-based grid computing. In *Proceedings of AAMAS2002 workshop on agentcities: Challenges in Open Agent Environments*, pages 32–36, Bologna, Italy, 2002. (document), 3.1.2

[141] M. Shirts, C. Snow, E. Sorin, and B. Zagrovic. Atomistic Protein Folding Simulations on the Submillisecond Time Scale Using Worldwide Distributed Computing. *Biopolymers*, 68:91–109, 2003. (document)

[142] D. Skeen. Nonblocking commit protocols. In *Proceedings of ACM SIGMOD Int'l Conf. Management of Data*, June 1981. 3.1.3, 3.1.4

[143] H. Smit and T. Li. Intermediate System to Intermediate System (ISIS) Extensions for Traffic Engineering (TE). *Internet Engineering Task Force, RFC3784, May*, 2004. 1.5.1

[144] R. Solomonoff and A. Rapoport. Connectivity of random nets. *Bulletin of Mathematical Biology*, 13(2):107–117, 1951. (document), 4.3

[145] S. Soltis, G. Erickson, K. Preslan, M. OŠKeefe, and T. Ruwart. The Global File System: A File System for Shared Disk Storage. *IEEE Transactions on Parallel and Distributed Systems*, 1997. 1.4.1

[146] A. Takefusa, M. Hayashi, N. Nagatsu, H. Nakada, T. Kudoh, T. Miyamoto, T. Otani, H. Tanaka, M. Suzuki, Y. Sameshima, et al. G-lambda: Coordination of a Grid scheduler and lambda path service over GMPLS. *Future Generation Computer Systems*, 22(8):868–875, 2006. (document), 1.5, 1.5.15

[147] S. Thorpe, D. Stevenson, and G. Edwards. Using Just-in-Time to Enable Optical Networking for Grids. *Proceedings of Broadnets*, 2004. 1.6

[148] F. Travostino, J. Mambretti, and G. Karmous-Edwards. *Grid Networks: Enabling Grids with Advanced Communication Technology*. Wiley, 2006. 1.5

[149] S. Vadhiyar and J. Dongarra. A metascheduler for the Grid. *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pages 343–351, 2002. 1.4.1

[150] P. Van Mieghem. PATHS IN THE SIMPLE RANDOM GRAPH AND THE WAXMAN GRAPH. *Probability in the Engineering and Informational Sciences*, 15(04):535–555, 2002. (document), 4.3

[151] D. Verchere, A. Pichot, B. Bela, O. Audouin, and al. Method of providing a gird network application over a transport network. Technical report, European Patent Office. (document), 4.6, 4.6

[152] O. Waldrich, P. Wieder, and W. Ziegler. A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources. *Parallel Processing and Applied Mathematics, LNCS*, 3911:782–791. (document), 1.4.1, 1.4.3, 1.5, 1.5.13, 2.2.1, 2.2.3

[153] O. Wäldrich, P. Wieder, and W. Ziegler. A meta-scheduling service for co-allocating arbitrary types of resources. In R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Wasniewski, editors, *Proceedings of the Second Grid Resource Management Workshop (GRMWS 05) in conjunction with Parallel Processing and Applied Mathematics: 6th International Conference (PPAM 2005)*, volume 3911 of *Lecture Notes in Computer Science*, pages 782–791. Springer, 2005. ISBN: 3-540-34141-2. 3.2.3

[154] O. Wäldrich and W. Ziegler. A ws-agreement based negotiation protocol. In *Draft paper published for Open Grid Forum 18 GRAAP-WG*, September 2006. (document), 3.2.3

[155] B. Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622, 1988. (document), 4.3

[156] V. Welch et al. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective, 2004. 1.4.2

[157] L. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, and H. Wittig. Issues of reserving resources in advance. *Proceedings of NOSSDAV, Lecture Notes in Computer Science*, pages 27–37. (document)