Allocation Distribuée de Requête dans les Réseaux de Capteur Sans Fil

Bing Han

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Thèse

présentée pour obtenir le grade de docteur

de l'Institut Télécom, TELECOM ParisTech

Spécialité : Informatique et Réseaux

Bing HAN

Allocation Distribuée de Requête dans les Réseaux de Capteur Sans Fil

Soutenue le 07 septembre 2009 devant le jury composé de

Annie Gravey	TELECOM Bretagne	Président
Jean-Yves Le Boudec	EPFL	Rapporteurs
David Simplot-Ryl	INRIA Lille - Nord Europe	Rapporteurs
Marcelo Dias De Amorim	CNRS LIP6	Examinateurs
Daniel Kofman	TELECOM ParisTech	Directeur de thèse
Gwendal Simon	TELECOM Bretagne	Directeur de thèse

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

PhD Thesis

submitted in partial fulfillment of the requirement for the degree of Doctor of Philosophy

in the Institut Télécom, TELECOM ParisTech

In: Computers and Networks

Bing HAN

Distributed Query Allocation in Wireless Sensor Networks

Presented 07 September 2009 before the committee composed of

Annie Gravey	TELECOM Bretagne	President
Jean-Yves Le Boudec	EPFL	Reviewer
David Simplot-Ryl	INRIA Lille - Nord Europe	Reviewer
Marcelo Dias De Amorim	CNRS LIP6	Examiner
Daniel Kofman	TELECOM ParisTech	Superviser
Gwendal Simon	TELECOM Bretagne	Superviser

To
my beloved parents
and
Emilie,

À
mes chères parents
et
Emilie,

Acknowledgement

The work presented in this thesis has been carried out in both Paris and Brest, where I met many brilliant scientists, colleagues and friends. I believe they deserve, for their continuous supports which are indispensable in realizing this thesis, my most profound gratitude.

I would like to thank first of all my supervisor Daniel Kofman, who offered me the great opportunity to work in the Computer Science Department of TELECOM ParisTech. He also introduced me to the Computer Engineering Department of TELECOM Bretagne, where I met my supervisor Annie Gravey to whom I am equally indebted for her warmly reception and insightful discussions and instructions on the research works.

I would like to express my sincere appreciation to Gwendal Simon, who has been instructing me on this thesis at both macro and micro levels. His eagerness for excellence in the research works and his encouragements at difficult times both have great influences on my research works. I will never forget the days we worked intensively on the papers and the discussions we had that inspired me a lot of new ideas. Many thanks to Jimmy Leblet and YiPing Chen, the collaboration with them has has been pleasant and fruitful.

It is my honor that the jury members accepted the invitation and spent their precious time helping me to improve this thesis. I wish equally to express my gratitude to them.

My sincere appreciation to all members of both Computer Engineering Department of TELECOM Bretagne and the Network and Computer Science Department of TELECOM ParisTech, especially Myriam Morcel, Sophie Bérenger, Hélène Melkebeke, Hayette Soussou and Armelle Lannuzel who perfectly take care of my frequent demands on following the administrative procedures so that I have been able to concentrate to the most extend on my research works.

I am also very much obliged to Jean-Yves Floch and Patrick Clément who provided their endless help on the working platforms, computers, networks, etc., to Yannis Haralambous who, as an awful LaTex expert, gave me many instructions on my first LaTex document, to Claude Chaudet and Maria Teresa Segarra, the discussions with them are always very interesting, to Lin Chen, Xiaoyun Xue, Lusheng Wang, Yaning Liu, Stefano Secci, Erwing Sanchez Sanchez, Minh Thanh Ngo, Tuan Dung N Guyen, An Phung-Khqc and Gabriela Athea Orez for the pleasant days with them.

I am indebted to my parents Yuyou Han and Yahui Liu, who tried all their best in ameliorating my living and studying conditions, gave me their full confidence and encouraged me during the most difficult times. I am indebted to my wife Yun (Emilie) Yang, whose endless patience and unconditional love pushes me forward. I enjoyed very much many surprises she gave me and tasting her desserts is really a fantastic experience.

Finally, I would like to thank Olivier Pothier, Catherine Lamy, Gregoire Pau and unknown contributors to the "ENST these" LaTex template with which this thesis is prepared.

Résumé

Introduction Générale

Le réseau de capteurs sans-fil (Wireless Sensor Network, WSN) est un réseau sans-fil composé des nœuds de capteurs distribuées charger de recueillir des informations sur le monde physique. L'idée de WSN est de créer un système pour recueillir, traiter et représenter l'information qui relie l'homme avec le monde physique. Beaucoup d'efforts ont été faits au cours des dernières années à la fois sur les aspects théoriques et applicatifs de WSNs et nous pouvons nous attendre à ce que le déploiement du WSN à grande échelle soit possible dans un avenir proche. Cette thèse se concentre sur le WSN à grande échelle qui peut potentiellement servir à de nombreux utilisateurs dans une architecture ouverte, où chaque utilisateur dispose d'un contact direct avec les capteurs. Nous allons nous référer à cette architecture de WSN avec utilisateur mobile. WSN avec utilisateur mobile est une solution prometteuse pour de nombreuses applications. Plusieurs utilisateurs du réseau peuvent agir comme collecteurs de données travaillant ensemble pour un travail en commun ou ils peuvent également être les utilisateurs finaux qui n'ont pas de contact direct entre eux. Dans les deux cas, l'équité parmi ces utilisateurs est une question importante.

Nous avons d'abord enquêté sur des questions d'équité avec un modèle de requête simple. Dans ce modèle, plusieurs utilisateurs font une requête sur des capteurs situés dans une région encerclée. La requête cercle est supposée avoir un diamètre variable et centrée sur l'utilisateur. Un problème d'optimisation distribuée avec des contraintes de congestion est formulée. Un algorithme heuristique est mis au point pour rapprocher la solution optimale. Ensuite, un problème similaire avec un modèle de requête discret est également étudié. Dans le modèle de requête discret, la requête est mesurée par le nombre de sauts qu'elle est diffusée. Cette variation rend le problème combinatoire et NP-difficile. Un problème sac à dos multidimensionnel et avec de multiples choix est utilisé pour modéliser le problème. Le lexicographique max-min d'équité et la couverture maximale de requêtes sont étudiés pour objectifs. Des solutions distribuées sont proposées et leurs performances sont démontrées par des simulations. Parce que la spécification ZigBee et la norme IEEE 802.15.4 sont deux normes de fait des réseaux de capteurs sans fil, nous étudions les problèmes d'un réseau de capteurs sans fil basé sur de telles technologies. Des propriétés particulières de la structure arborescente ZigBee sont exploitées à garder l'algorithme entièrement local ainsi qu'à limiter les communications dans le réseau. L'efficacité des algorithmes est démontrée par simulations.

Alors que tous les sujets abordés à ce jour sont liés à la répartition de la capacité dans les réseaux de capteurs sans fil avec de multiples utilisateurs, le problème combinatoire derrière la version discrète, problème de sac à dos multidimensionnel et à choix multiples, est très intéressant et mérite une attention particulière. L'incohérence du temps pour obtenir la solution entre les problème avec des paramètres similaires et du fort contraste entre les petits problèmes difficiles et des grands problèmes faciles sont notre motivations pour de plus amples enquêtes sur le problème lui-même. En premier, nous proposons des méthodes pour générer des

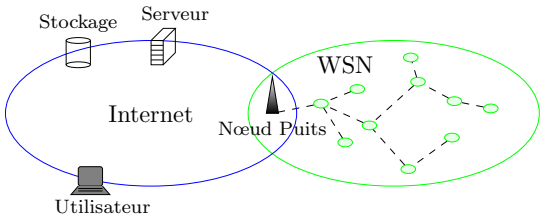


Figure 1: Un réseau de capteurs sans fil avec un nœud-puits fixe.

problèmes avec différentes propriétés, nous essayons de résoudre plusieurs groupes de cas, avec l'algorithme actuel/outils. Des propriétés particulières qui rendent difficiles les cas ont été identifiées.

État d'art des Réseaux de Capteur

Le réseau de capteurs sans fil est considéré comme une méthode prometteuse pour de nombreuses applications, y compris à la fois les applications militaires traditionnelles et les nouveaux émergés dans les domaines scientifiques et civiles. Pour le premier, il s'agit notamment de surveiller le champ de bataille, la frontière, les cibles mobiles, *etc.* Pour la suite, surveillance de l'environnement, de volcan, des animaux sauvages ou végétales sont des domaines d'application typiques dans l'étude scientifique, tandis que d'autres applications pour la surveillance de sécurité d'architecture, mauvais fonctionnement, de la santé des personnes et la circulation automobile, ou à faciliter au cours de risque tels que les tremblements de terre, d'inondation ou d'incendie. Évidemment, on est incapable d'énumérer toutes ces applications car elles sont en pleine expansion dans de nombreuses régions où les (filaire) capteurs sont employés à des fins de contrôle.

La conception de réseaux de capteurs sans fil dépend fortement de son application. En fait, le choix des capteurs à bord, de la taille du nœud, le type de nœud-puits, le mécanisme de communication, la structure du réseau et les logiciels doivent être choisis avec soin pour répondre aux exigences particulières de la demande. Nous allons nous concentrer uniquement sur l'aspect mise en réseau de l'ensemble du système et d'identifier trois types d'architectures du réseau WSN. Si les deux premiers ont trouvé leurs applications dans le monde réel, le troisième vient juste d'émerger.

WSN avec nœud-puits fixe Dans certains WSN applications militaires et scientifiques, les capteurs sont installés manuellement aux positions soigneusement conçues pour former un réseau statique. Un nœud-puits est également installé à une position fixe et des données sont envoyées à partir de capteurs au serveur central par l'intermédiaire des nœud-puits. Nous appelons cela l'architecture de réseau WSN avec nœud-puits fixe, comme le montre la Figure 1. Un problème évident de WSN avec nœud-puits fixe est le goulot d'étranglement formé autour du nœud-puits, à la suite de l'agrégation des données de tous les capteurs. Ce goulot d'étranglement consume d'énergie sur les capteurs autour du nœud-puits beaucoup plus vite que

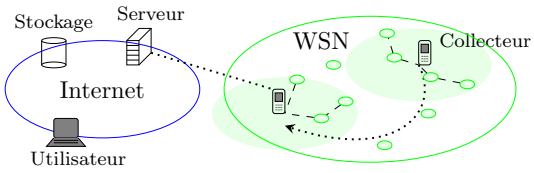


Figure 2: Un réseau de capteurs sans fil avec un nœud nœud-puits mobile.

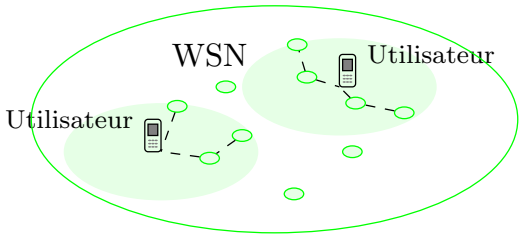


Figure 3: Un réseau de capteurs sans fil avec des utilisateurs mobiles.

les autres capteurs et, enfin, quand ces capteurs sont épuisés, le reste du réseau est séparé du nœud-puits. Installation de plusieurs nœud-puits dans le réseau permet d'atténuer les goulots d'étranglement, cependant, cette façon de faire augmente à la fois les investissements et la complexité du réseau. En WSN avec nœud-puits fixe, le client ou l'utilisateur accède aux services fournis par le WSN à travers le back-end serveur.

WSN avec nœud-puits mobile Dans certaines circonstances, il est impossible ou inutile d'installer le capteur à une position exactement contrôlée et, parfois, les capteurs ne peuvent pas former un réseau connecté au nœud-puits. Par exemple, une densité faible de capteurs déployés est suffisant pour la tâche donc la distance entre les capteurs est au-delà de leur distance de transmission maximum. En revanche, un réseau est inutile et coûteux. Dans ce cas là, les chercheurs ont proposé d'utiliser certains dispositifs mobiles pour la collecte des données provenant des capteurs lorsque les dispositifs se déplacent à l'intérieur de la gamme de transmission de capteurs. Nous avons appelé ces appareils mobiles des nœud-puits mobiles et de l'architecture de WSN avec nœud-puits mobile. Les nœud-puits mobiles sont également connus comme les mulets de données ou les collecteurs de données dans la littérature. Les données générées par les capteurs sont stockées temporairement avant qu'un nœud-puits mobile se déplace à proximité, puis transféré au nœud-puits mobile, et apporté au serveur pour le traitement et, enfin, fourni aux utilisateurs. Dans le WSN avec nœud-puits mobile, le nœud-puits mobile est une partie du déploiement de réseau, conçu, déployé et géré par l'opérateur du WSN. WSN avec nœud-puits mobile atténue le goulot d'étranglement dans le WSN avec nœud-puits fixe. Figure 2 est une illustration brève de l'architecture WSN avec nœud-puits mobile.

WSN avec utilisateur mobile Dans les applications WSN au grand public, le nombre d'utilisateurs potentiels du réseau sera très grand. Les architectures actuelles

de réseau WSN avec les nœud-puits fixes ou les nœud-puits mobiles rencontrent des difficultés d'évolutivité du service pour les utilisateurs via les back-end serveurs. En outre, les utilisateurs de ce type de services sont généralement intéressés par certaines informations sur leur environnement ambiant, c'est-à-dire les utilisateurs sont intéressés par ce qui se passe à proximité. Ce type d'information comprend la température, bruit, parking, intensité du trafic, la disponibilité de certains produits dans les supermarchés, *etc.* Dans ce cas là, un scénario plus efficaces de la récupération de l'information pourrait être de laisser les utilisateurs communiquer directement avec les capteurs collecter des informations avec les types de capteurs qui l'entourent. Nous avons pour nom de cette architecture le WSN avec utilisateur mobile, comme illustrée par la Figure 3. Nous soulignons également que le WSN avec utilisateur mobile permet aux utilisateurs de profiter de plus de liberté lors de l'interrogation des capteurs. Pour les requêtes portant sur une vaste zone géographique, une transmission des données à la manière multi-saut est nécessaire.

WSN avec utilisateur mobile présente de nombreux avantages par rapport aux deux architectures, notamment pour certains types de demandes. D'une part, de nombreux types d'informations ont seulement valeur en temps réel et il n'est donc pas nécessaire de les stocker pour un traitement ou pour des fins de vérification. Par conséquent, les données peuvent être fournies aux utilisateurs par transmission directe, sans l'aval du serveur au préalable. D'autre part, les capteurs dans ce scénario ne peuvent être déclenchés que par une demande de l'utilisateur. S'il n'y a pas de requête à traiter, les capteurs peuvent rester dans le mode consommation minimisée. En outre, la communication se fait uniquement entre les utilisateurs et les capteurs autour d'eux, les capteurs ne participent pas à la requête qui n'a pas d'effet sur eux. En revanche, dans un WSN avec nœud-puits fixe, les capteurs sans tâches de détection doivent aussi communiquer afin d'aider à la transmission de données pour les autres, consommer de l'énergie.

WSN avec utilisateur mobile diffère de WSN avec nœud-puits fixe ou mobile dans une manière très importante, c'est-à-dire que le WSN avec nœud-puits qu'il soit fixe ou mobile n'est pas le consommateur final de l'information produite par les capteurs. Ils agissent au nom de l'exploitant du WSN la collecte de données provenant des capteurs, la transmission des données de back-end serveurs, les passerelles de commandes à partir des serveurs de capteurs. Le dispositif utilisé est sélectionné à la conception, leur scénario de requête est pré-programmée, leur position ou la trajectoire est prévue, tous par l'opérateur. Par conséquent, le réseau est un système fermé fournissant des services uniquement par certains back-end serveurs. En revanche, l'architecture de WSN avec utilisateur mobile est plus ouverte pour les utilisateurs d'une manière que les utilisateurs puissent recueillir directement des données provenant des capteurs avec leurs appareils mobiles tels que des téléphones portables ou les PDA. Toutefois, il convient de noter que le WSN avec utilisateur mobile et les deux autres architectures ne sont pas mutuellement exclusives. Au lieu de cela, les utilisateurs mobiles peuvent co-exister avec un nœud-puits fixe ou mobile, ce qui donne une architecture hybride et plus flexible.

Motivations et objectifs Cette thèse a été motivée par la collecte d'information in-site appliquées à des applications du service public à grande échelle. Nous nous sommes intéressés en particulier à l'architecture de WSN avec utilisateur mobile. Ces applications présentent de plus en plus d'intérêts à la fois industriels et académiques. Une architecture à plusieurs nœud-puits de mobile a été proposée dans [CM06]. Dans cette architecture, les téléphones cellulaires sont équipés de plusieurs interfaces sans fil: un réseau de base de communication mobile et l'autre pour communiquer avec d'autres périphériques via les connexions sans fil courte distance, tels que Bluetooth et ZigBee. Sur la base de ces téléphones cellulaires multi-radio, il est possible de fournir des services de WSN utiles pour le grand public. Nous soulignons ici plusieurs d'entre eux avec seulement leurs scénarios de base et les caractéristiques distinctes.

- Système d'information Omniprésent: Capteurs sans fil peuvent être déployés le long des rues de la ville et les parkings afin que les pilotes puissent accéder à l'information en temps réel sur le trafic à venir et les parkings disponibles à proximité avec leurs téléphones cellulaires. Apparemment, on se soucie plus généralement des informations de trafic sur une petite région, ou d'information du parking autour de la destination.

- Système Contre-Émergence: Un autre domaine d'application pour un WSN avec utilisateur mobile peut être le recueil d'information pendent des opérations d'émergence. Nœuds de capteurs sans fil peuvent être déployées à l'intérieur et autour d'un feu afin de faciliter l'opération. Avec certains dispositifs portables, les pompiers sont en mesure de recueillir diverses informations provenant de capteurs qui les entourent, afin de prendre des bonnes actions afin qu'elles puissent se tenir à l'écoute des explosions dangereuses, éviter d'être pris au piège ou à localiser les victimes.

Nous insistons sur plusieurs caractéristiques de ces types d'applications et le WSN avec utilisateur mobile ci-dessous:

- Le réseau de capteurs sans fil dans ce cas est plus orienté vers l'utilisateur que les applications militaires ou scientifiques. Pour le système d'information omniprésent mentionné ci-dessus, les utilisateurs ont accès aux services par un contrat, devenant ainsi les clients du réseau. En conséquence, il est nécessaire que les fournisseurs de services satisfont leurs clients. Pour les systèmes contre-urgence, satisfaire les besoins fondamentaux de chaque agent est important même s'il est plus probable qu'ils appartiennent à la même organisation.

- Les utilisateurs ne sont pas gérés par les opérateurs du réseau. S'il est vrai que les appareils en service doivent répondre à certains besoins fonctionnels, certains contrats entre les utilisateurs et l'opérateur sont nécessaires, les profils utilisateur, tels que des mode d'accès, la position, *etc.* ne peuvent pas être contrôlés par l'opérateur.

- Les utilisateurs sont autonomes et ils n'ont généralement pas de contact direct les uns avec les autres. Au contraire, ils sont habituellement en concurrence les uns avec les autres pour les ressources du réseau.

En vertu d'un tel environnement autonome, avec la contrainte de ressources stricte, de multiples utilisateurs doivent partager les ressources communes de manière satisfaisante en vue d'exploiter le réseau dans un état optimal en fonction de la population d'utilisateurs et de la répartition géographique. En particulier, on voudrait empêcher que certains utilisateurs soient bloqués parce qu'ils entrent dans le réseau après que les autres utilisateurs ont consommé toutes les ressources disponibles. Cela exige d'enquêter sur une série de problèmes d'optimisation, qui sont au centre des préoccupations de cette thèse.

Cette thèse vise les problèmes d'optimisation de réseau de l'origine du WSN avec utilisateur mobile défini précédemment. Les problèmes seront élaborés avec la programmation mathématique et résolues de manière analytique et algorithmique. Deux objectifs d'optimisation seront étudiés, l'un est de maximiser la satisfaction de tous les utilisateurs dans le réseau et l'autre est de donner à chaque utilisateur une partie équitable du service fourni par le réseau. Les deux objectifs d'optimisation soulignent davantage le côté utilisateur. Ce point de vue est considéré comme l'une des nouveautés de cette thèse. Lorsque le WSN doit fournir les services à ses clients, d'optimiser des paramètres au côté client, par exemple, leur gamme de requête, est tout à fait logique. En conséquence, les paramètres au côté capteur qui sont considérés comme critiques dans la littérature courante, comme l'efficacité de la consommation d'énergie et de la durée de vie du réseau, sont d'importance secondaire dans cette thèse. Nous estimons que la limite des ressources sur le capteur seulement comme des contraintes dans la formulation des problèmes d'optimisation. La ressource en cours d'examen est la bande passante effective du capteur.

De toute évidence, pour une architecture de réseau, tels que le WSN avec utilisateur mobile que nous étudions, toute solution doit être capable de s'adapter à la dynamique de réseau. Par exemple, un utilisateur rejoint ou quitte, le changement de sa position, la quantité d'information il requit, *etc.* Cela exige essentiellement une solution distribuée qui permet la coopération entre les utilisateurs et les capteurs à la réalisation de certains objectifs d'optimisation. Par conséquent, cette thèse se concentrera principalement sur la formulation des problèmes d'optimisation et les solutions distribuées.

En résumé, cette thèse a pour objectif de proposer des algorithmes distribués pour les utilisateurs et les capteurs dans un WSN afin d'allouer un rayon optimal de requête pour chaque utilisateur de façon adaptative et évolutive.

Allocation de Requête dans un WSN avec Utilisateur Mobile

Dans certains application contre-émergence, le réseau de capteur sans fil pourrait considérablement aider les équipes d'intervention par notification à certains événements. Une application typique de ce réseau pourrait être un système de surveillance avec des pompiers équipés de dispositifs portables qui collectent les données à partir d'une zone de combustion afin de déterminer un périmètre de sécurité, tandis que d'autres opèrent sur le foyer et sont en alerte en temps réel sur les risques d'explosions à proximité. Les pompiers ont envoyé des demandes pour recueillir des données provenant des capteurs à l'intérieur d'un domaine spécifique.

Ainsi, ils pourraient être considérés comme les utilisateurs mobiles.

Ce système de surveillance doit être digne de confiance, afin que tous les événements dans une zone contrôlée doivent être signalés à l'utilisateur d'interrogation, et devraient être adaptables, puisqu'il y a des changements de nombre d'utilisateurs au fil du temps. En outre, il est souhaité pour chaque utilisateur de contrôler la plus grande zone possible pour assurer la sécurité individuelle. Toutefois, étant donné que le nœud de capteur a une bande passante limitée, de multiples requêtes pourraient congestionner les capteurs dans certaines zones, en fonction de la position des utilisateurs, la taille de la zone, *etc.* En outre, si les capteurs sont déjà saturés par les requêtes, les utilisateurs récemment arrivés seront conservés dans un état de faim et de l'impossibilité de récupérer les informations. Au contraire, si chaque pompier accepte de diminuer la taille de sa zone de requête, cela peut permettre aux nouveaux venus d'utiliser le réseau. Ainsi, l'équité entre les utilisateurs partageant le réseau est préférable. En conséquence, les utilisateurs ont tendance à augmenter leurs rayons de requête tant que des règles d'équité sont conservés. Nous supposons que les utilisateurs ne sont pas coordonnés par les liens de communication directs.

Un Modèle de Requête Continu

Une requête émise par un utilisateur appartient à une zone supposée être un disque de rayon variable centré sur l'utilisateur. Chaque capteur à l'intérieur de la zone a demandé de générer certaines quantité de données par seconde en réponse à la requête. Un capteur peut être couvert par plusieurs requêtes.

Le modèle de la charge de trafic que nous utiliserons a été proposé dans [GK04] et étendu dans [LH05]. Avec ce modèle de trafic, nous sommes en mesure d'obtenir une représentation analytique du trafic généré par un capteur de n'importe quelle position dans le réseau. C'est une fonction de la position du nœud et de l'utilisateur qui interroge le capteur, ainsi que d'autres paramètres, montré comme suit:

$$\delta(d_{ki}) = \begin{cases} \frac{br^2}{(r^*)^2} & k \in r_i^* \\ b + \frac{2br^2}{\pi(r^*)^2} \arcsin\left(\frac{r^*}{d_{ki}}\right) - \frac{2bd_{ki}^2}{\pi(r^*)^2} \arcsin\left(\frac{r^*}{d_{ki}}\right) & k \in Q_{r_i b} \setminus r_i^* \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

où r^* signifie la distance maximum de transmission d'un capteur, r_i^* signifie la disque de rayon r^* centrée au capteur i et $Q_{r_i b}$ signifie la région couverte par la requête de rayon r_i d'utilisateur i avec b comme le trafic demandé sur chaque capteur.

Configuration Équité Max-min: Une configuration est définie par un ensemble de paire rayon-utilisateur: $C = (r_i | i \in S)$ et nous disons que C est une configuration faisable s'il n'y a pas de congestion dans le réseau. Nous nous sommes intéressés à l'équité max-min, c'est à dire que C possède des propriétés telles que les rayons plus petits sont maximisés, les rayons deuxièmes plus petits sont maximisés et ainsi de suite. Une règle de base est que l'augmentation d'une requête ne devrait pas diminuer d'autres requêtes déjà plus petits. De toute évidence, la limite de la capacité doit être maintenue. Notez que dans le contexte que nous examinons, le trafic b est fixe et chaque utilisateur s'est intéressé à maximiser son rayon de requête afin de maximiser la sécurité individuelle.

Un Modèle de Requête Discrète

La requête est mesurée par le nombre de sauts tels qu'une requête de rayon j couvre tous voisins de j sauts de l'utilisateur.

Nous avons supposé que tous les nœuds inclus dans la requête de l'utilisateur doivent fournir une certaine quantité de ressources. Notez que si un nœud k est dans une requête, alors tous les nœuds le long du chemin de transmission de données sont dans la même requête et ils consomment des ressources en transmettant les données à partir de k. Comme nous avons également assumé qu'un mécanisme de routage avec les chemins plus courts est en cours d'utilisation, le montant de ressource des nœuds le long de la route doit fournir, en raison de l'impact de l'utilisateur i, est égal ou supérieur à ce que le nœud k doit fournir, si le flux de données est vers l'utilisateur. A noter que nous supposons qu'il n'y a pas de compression ou d'agrégation des données sur les chemins.

Chaque nœud k admet des ressources quantifiables comme c_k qu'il est en mesure de fournir à certains utilisateurs. Le montant de ressource sur le nœud k que l'utilisateur i consomme lorsque son rayon de requête est fixé à j est désignée comme w_{ijk}. Si le nœud i n'appartient pas à V_{j_ib}, nous avons $w_{ijk} = 0$.

Nous n'utilisons pas la fonction de trafic pour cette étude. Au lieu de cela, nous allons mettre en place des mécanismes pour chaque nœud pour mesurer le trafic réel.

Nous définissons l'allocation de requête avec équité max-min (MMF) de la même manière que nous l'avons fait pour le modèle continu. Par ailleurs, nous définissons également un autre objectif d'optimisation afin de maximiser la somme des rayons de requêtes de tous les utilisateurs (MNU).

Nous modélisons le problème avec un problème sac à dos. Chaque requête d'utilisateurs i correspond à un élément à être sélectionné et la valeur des éléments pourrait être le nombre de saut de 0 au diamètre de G noté D_G. Étant donné que chaque utilisateur est autorisé à choisir une seule valeur pour sa requête à la fois, les éléments peuvent être considérés comme regroupés en m classes et chaque classe correspond à un utilisateur. Ainsi, nous avons la contrainte de choix multiples que exactement un élément doit être choisi au sein de chaque classe. Une variable binaire x_{ij} est associée à la gamme de requête j de l'utilisateur i où $x_{ij} = 1$ indique que l'utilisateur i définit ses requête gamme j, et $x_{ij} = 0$ autrement. Le montant des ressources fournies par un nœud k à un utilisateur i quand i prend chacune de ses requêtes possible $j \in (0, 1, \ldots, D_G)$ sera considéré comme un ensemble $(w_{i0k}, w_{i1k}, \ldots, w_{id_Gk})$, et chaque w_{ijk} pourrait être associé à la $i$ème dimension du poids d'un élément j de la classe i. Chaque nœud k forme une dimension de contrainte avec ses ressources disponibles c_k. Comme il y a beaucoup de nœuds dans le système, nous avons enfin une contrainte multi-dimensionnelle, c'est-à-dire, un MMKP. Le problème d'allocation de requête discret de maximiser la somme de l'utilité de chaque utilisateur, est exactement le classique MMKP. Il pourrait être résolu par des algorithmes MMKP. Cependant, tous ces algorithmes examinent la résolution du problème dans un mode central. Étant donné la nature des problèmes qui ont motivé ce travail, un algorithme distribué est préférable. Nous allons pro-

poser un tel algorithme. En outre, nous avons également formulé le problème MMKP avec l'équité max-min. Cet objectif est nouveau et intéressant car il démontre la possibilité de formuler différents objectifs d'optimisation dans un cadre de MMKP.

Solution du Modèle Continu

Calculer le rayon d'équité max-min. Si la requête prend une valeur continue, nous sommes en mesure de le calculer en fonction de l'équité max-min dans un réseau avec seulement deux utilisateurs. Nous considérons que deux utilisateurs de i et i' sont dans le réseau.

Dans un WSN avec des nœuds à la distribution Poisson et un paradigme convergent de communication, le goulot est l'utilisateur lui-même [MDMLN03]. Nous disons que la requête est globalement maximisée quand la bande passante de tous les nœuds dans les r_i est saturée par le trafic exclusivement dédié à i. Si on fait abstraction de la bande passante consommée par le protocole, la requête est globalement maximisée quand $\pi r_i \lambda_0 b = W$. Lorsque le débit de données demandé (b) et la densité de capteurs (λ_0) sont fixes, nous allons naturellement obtenir:

$$r_{max} = \sqrt{\frac{W}{\pi b \lambda_0}} \tag{2}$$

Dans ce cas, le montant maximum de données qu'un utilisateur peut recevoir est limité par sa bande passante et les deux requêtes pourraient être maximisées avec la même rayon r_{max}. De toute évidence, la configuration $C = (r_i, r_{i'})$ avec $r_i = r_{i'} = r_{max}$ est max-min équité. On peut obtenir les mêmes résultats pour les cas suivants.

Deux utilisateurs pas si lointains: Dans ce cas là, deux utilisateurs peuvent fixer les requêtes à r_c et chaque utilisateur peut déterminer r_c en communiquant avec un capteur de sa requête. Ainsi, la configuration max-min pour ce cas est $C = (r_i, r_{i'})$ où $r_i = r_{i'} = r_c$, et:

$$r_c = \sqrt{\frac{\left(\frac{W}{\pi (r^*)^2 \lambda_0} - b + t_1\right)}{t_2}}, \tag{3}$$

où:

$$t_1 = \frac{2b(d-r^*)^2}{\pi (r^*)^2} \arcsin\left(\frac{r^*}{d-r^*}\right) \tag{4}$$

$$t_2 = \frac{b}{(r^*)^2} + \frac{2b}{\pi (r^*)^2} \arcsin\left(\frac{r^*}{d-r^*}\right) \tag{5}$$

Deux utilisateurs à proximité: La configuration max-min est $C = (r_i, r_{i'})$ avec

$$r_c = \sqrt{\frac{W}{2\pi \lambda_0 b}}. \tag{6}$$

Ça complète l'analyse de la configuration max-min pour le cas de deux utilisateurs.

Un algorithme distribué L'algorithme se compose de deux phases. Le premier, généralement appelé démarrage lent, est utilisée jusqu'à ce qu'une approximation d'un rayon réalisable est obtenue. Le rayon commence à une valeur unitaire et augmente selon une certaine stratégie jusqu'à ce que l'utilisateur soit prévenu par un message `<saturated>`. La fonction `initRadius` gère l'augmentation initiale de rayon. Le `initRadius` est fondé sur une croissance exponentielle, c'est-à-dire le rayon de requête est initialisé à 1 et doublé à chaque fois qu'il est appelé. L'idée est de détecter le plus rapidement possible le goulot d'étranglement de capteurs.

À la réception du premier message `<saturated>`, le rayon est fixé à une valeur plus basse en fonction de certains critères. Le nouveau rayon est obtenu par `resetRadius`. Le `resetRadius` emploie le résultat d'analyse pour les deux utilisateurs présentés ci-dessus, et la fonction retourne un rayon proche de la solution optimale dans la plupart des cas. Dans la deuxième phase, chaque utilisateur essaie d'augmenter son rayon de requête périodiquement, afin d'explorer le rayon optimal. Ceci est pris en charge par la fonction `increaseRadius`. Le `increaseRadius` fait une fonction croissante linéaire du rayon après chaque intervalle de temps prédéfini par une étape de longueur prédéfinie. Finalement, l'utilisateur est prévenu par un message `<saturated>`, puis il diminue à nouveau son rayon et entre dans un nouveau cycle.

Solution du Modèle Discrète

L'idée de base des algorithmes est de résoudre un problème localisé beaucoup plus petit à chaque nœud. À cette fin, chaque nœud garde la trace de tout le trafic passant par lui-même. Avec cette information, quand un nœud est saturé par plusieurs requêtes d'utilisateurs, il est en mesure de formuler un MMKP locale avec seulement les utilisateurs concernant et une unique contrainte. Ce problème est généralement beaucoup plus petit et pourrait être résolu rapidement. Puis le nœud avertit les utilisateurs en relation avec la solution. Éventuellement, l'utilisateur adopte une des solutions multiples en tant que son nouveau rayon de requête. Dans le reste de cette section, nous donnons d'abord des discussions détaillées sur les techniques clés que nous emploierons, puis nous proposons un algorithme unifié pour les deux objectifs d'optimisation MMF et MNU.

Solution du MCKP Local

Lorsque le nœud est saturé, il formule un petit problème MMKP avec ses mesures du trafic local et une seule contrainte, ou de manière équivalente, un problème MCKP. Ce MCKP doit être résolu d'une manière centralisée ainsi des algorithmes exacts ou heuristiques peuvent être exploités. Divers algorithmes ont été proposés pour résoudre un MCKP avec l'objectif de maximisation de la somme d'utilité de tous les utilisateurs. Nous adoptons un outil GLPK pour résoudre ce problème MCKP dans le cadre d'un algorithme distribué que nous allons proposer pour le problème de MNU. En revanche, aucun algorithme n'a été proposé pour un MCKP avec l'objectif MMF. Depuis que le MCKP est un cas particulier de MMKP, il est possible d'appliquer directement l'algorithme exact que nous allons proposer à résoudre ce MCKP. Toutefois, le coût de calcul augmente de façon exponentielle d'un

Algorithm 1: Local MCKP-MMF

input : $S' \subseteq S$, U, $M(k)$, c_k
output: MCKP-MMF configuration C

for $i \in S'$ **do**
  $C \leftarrow \{S_i = (j_i \leftarrow 0, s_i \leftarrow \mathsf{active})\}$
end
for $j \leftarrow 1$ *to* $|U|$ **do**
  $A \leftarrow \{S_i : s_i = \mathsf{active}\}$
  if $A = \emptyset$ **then** break
  `sort` $(A, t_{.j})$
  for $a \leftarrow 1$ *to* $|A|$ **do**
    $C' \leftarrow C$, $j'_a \leftarrow j$
    if `feasible` (C') **then** $j_a \leftarrow j$
    else
      for $a' \leftarrow a$ *to* $|A|$ **do** $s_{a'} \leftarrow \mathsf{stop}$
      break
    end
  end
end
Return C

algorithme exact, même si le nombre d'utilisateurs utilisant un nœud congestionné est raisonnablement petit. Dans ce qui suit, nous proposons une heuristique d'un tel problème MCKP. Cette heuristique sera intégrée dans l'algorithme distribué.

Adaptation Dynamique du Rayon de Requête

Après que le problème MCKP local est résolu, le nœud envoie le résultat à tous les utilisateurs concernés afin de leur indiquer leurs nouveaux rayons de requêtes qui sont censés se conformer à la contrainte sur ce nœud. Toutefois, un utilisateur peut recevoir ces notifications multiples à partir de plusieurs nœud. Ainsi, afin de satisfaire la contrainte la plus stricte, un utilisateur doit adapter son rayon de requête en fonction de la valeur la plus petite dans toutes les notifications. L'effet secondaire de cette politique est que la requête tend à diminuer sur le long terme et un utilisateur peut ne pas être en mesure de connaître son rayon de requête optimal en raison des informations incomplètes dont il dispose. Pour atténuer cet effet secondaire et aider les utilisateurs à sauter d'un optima local affectés par l'algorithme 1, chaque utilisateur devrait augmenter son rayon de requête périodiquement.

Un Algorithme Unifié

Maintenant que toutes les conditions préalables sont discutées, nous sommes en mesure de donner les algorithmes principaux à la fois pour les problèmes de MNU et MMF. Comme indiqué au début de la section, les deux algorithmes peuvent être décrits dans un cadre générique, comme indiqué dans l'algorithme 2.

Algorithm 2: Distributed Heuristic

Sink Part : Run at user i
`send` $< \texttt{level}, i, 1 >$
while $no <$ `adjust-level` $>$ $message$ **do**
  $level \leftarrow$ `initLevel()`
  `send` $< \texttt{level}, i, level >$
end
$level \leftarrow$ `adjustLevel()`
while true **do**
  while $no <$ `adjust-level` $>$ $message$ **do**
    $level \leftarrow$ `increaseLevel()`
    `send` $< \texttt{modify-level}, i, level >$
  end
  $level \leftarrow$ `adjustLevel()`
  `send` $< \texttt{modify-level}, i, level >$
end

Sensor Part: Run at sensor k
while true **do**
  if `congested()` **then**
    $C \leftarrow$ `solveMCKP()`
    for $\forall i : j_i \in C$ **do**
      `send` $< \texttt{adjust-level}, j_i >$ to i;
    end
  end
end

Le Problème Derrière: Instance Difficile du MMKP

Nous avons formulé la version discrète du problème d'allocation de requête pour un WSN avec utilisateur mobile avec un problème sac à dos multidimensionnel aux choix multiples. Nous nous concentrons maintenant sur ce problème combinatoire lui-même.

MMKP a de nombreuses applications. Il a été utilisé pour modéliser le problème de gestion de la qualité de service (QoS) dans les réseaux informatiques [LLRS99] et les problèmes de contrôle adaptatif d'admission dans le systèmes multimédia [SIH05, Kha98, KLMA02]. Divers autres problèmes d'allocation des ressources peuvent également être directement représentés par le MMKP [KPP04, PHD05].

Nous étudions la relation entre les différents paramètres d'un MMKP telles que le profit, le poids et la capacité afin d'identifier les facteurs clés qui rendent une instance difficile. En outre, les cas où les éléments sont non corrélés, faiblement corrélés et fortement corrélés au sein de chaque classe, entre les classes et à travers de multiples dimensions sont examinés. À notre connaissance, aucun travail n'a été signalé dans la littérature. Une méthode systématique pour générer les instances de MMKP est proposée. Plusieurs groupes de cas obtenues avec cette méthode sont

évalués avec un algorithme exact et les outils GLPK [glp] et CPLEX [cpl]. Les expériences montrent que de nombreux cas sont de plusieurs ordres de grandeur plus difficiles que ceux traditionnellement utilisés en terme de temps de calcul. Ces instances dures du MMKP ont généralement une capacité moyenne et une forte corrélation entre le poids et le profit. Les expériences suggèrent également que les cas avec les profits similaires entre les classes et avec une forte corrélation entre le poids et le profit sont difficiles à résoudre.

Noter qu'il est très important de tester les algorithmes afin de connaître leurs performances en pratique. Lorsque les algorithmes sont pour attaquer à un problème particulier, les instances idéales pour l'évaluation des performances sont ceux de traces du monde réel. Toutefois, comme les MMKPs proviennent habituellement des contextes applicatifs diversifiés, les instances typiques à partir d'un certain domaine ne peuvent guère être raisonnables pour les autres. En outre, il n'existe pas de rapport systématique sur les instances de MMKP dans la littérature. En revanche, les cas de test peuvent être générés pour couvrir les instances de type d'une gamme beaucoup plus large. En conséquence, les instances générées jouent un rôle important dans l'évaluation comparative des algorithmes et ont été utilisées dans les recherches de KP et MMKP.

Les chercheurs ont proposé une librairie des instances de MMKP [OR-]. Bien que ces instances ont été largement utilisées dans la littérature, nos résultats de calcul montrent qu'ils ne sont pas suffisants pour démontrer les performances des algorithmes. Table 1 présente le temps utilisé pour résoudre les six premières instances dans la librairie avec CPLEX, GLPK et l'algorithme BBLP [Kha98, KLMA02]. Ici, nous insistons sur le temps utilisé à travers les instances. Notamment, les instances I3 et I4 prennent plus de temps que I5 et I6, malgré qu'ils soient plus petits que les seconds. Cela implique effectivement que non seulement la taille d'une instance, mais aussi la structure d'une instance joue un rôle très important dans le temps de résolution.

Table 1: Temps (second) utilisé pour résoudre les instances dans "OR benchmark library".

Inst	m	n	l	CPLEX	GLPK	BBLP
I1	5	5	5	0.005	0.028	0.016
I2	10	5	5	0.006	0.029	0.033
I3	15	10	10	1.983	16.036	67.260
I4	20	10	10	31.045	1383.251	1532.059
I5	25	10	10	0.018	0.046	0.660
I6	30	10	10	0.204	0.190	2.369

Nouvelles Méthodes pour Générer les Instances de MMKP

Générer les Profits

Afin de sélectionner les profits des éléments dans chaque classe i, nous avons d'abord limité les profits avec deux paramètres $p_i^{\min}$ et $p_i^{\max}$ et choisi des valeurs dans cette intervalle. Cela pourrait se faire de diverses manières et ici, nous définissons quelques fonctions génératrices pour les profits.

Fonction Génératrice Uniforme Profits aléatoires sont naturelles dans de nombreux problèmes et sont largement utilisés dans la littérature. Dans une fonction génératrice uniforme, nous tirons les profits de manière uniforme et aléatoire dans un intervalle. On note la fonction génératrice uniforme:

$$p_{ij} = \mathcal{U}\left(p_i^{\min}, p_i^{\max}\right). \tag{7}$$

Fonction Génératrice Linéaire Éléments avec les profits linéaires sont moins étudiés dans la littérature. Toutefois, ce type de valeur est en fait assez commun. Par exemple, dans le problème de QoS adaptative [AHHS05], les niveaux de qualité de service sont habituellement représentés sur les profits d'éléments et de valeurs entières consécutives. Aussi dans le problème d'allocation de requête multi-sauts [HLS09], les rayons de requêtes sont mis en correspondance avec les profits et se mesurent en nombre de sauts et ils prennent aussi des valeurs entières consécutives.

Dans la fonction génératrice linéaire, nous assignons p_{ij} avec une fonction linéaire de l'indice j, *i.e.*

$$p_{ij} = \frac{j-1}{n_i - 1}\left(p_i^{\max} - p_i^{\min}\right) + p_i^{\min}. \tag{8}$$

Pour plus de clarté, nous utilisons une notation courte pour cette fonction génératrice linéaire comme suit:

$$p_{ij} = \mathcal{L}\left(p_i^{\min}, p_i^{\max}\right). \tag{9}$$

Application des Fonctions Génératrices Les fonctions génératrices doivent être appliquées sur chaque classe. Évidemment, on peut appliquer la même fonction pour toutes les classes ou modifier les fonctions de chaque classe. Pour une fonction génératrice uniforme, même quand elle est appliquée à toutes les classes avec les mêmes paramètres, la nature aléatoire de la fonction va donner différentes valeurs pour les profits dans les différentes classes. Au contraire, lorsque la fonction génératrice linéaire est appliquée à toutes les classes avec les mêmes paramètres, toutes les classes auront le même vecteur de profit pour leurs éléments. Au lieu d'appliquer la même fonction génératrice à toutes les classes, nous proposons en outre deux façons d'utiliser les fonctions génératrices. La première est de reproduire le vecteur généré par une fonction génératrice uniforme sur toutes les classes.

C'est typiquement le cas lorsque plusieurs utilisateurs (classes) peuvent accéder aux mêmes ensembles d'objets (éléments) avec plus ou moins de qualité du service (profits), mais le coût pour y accéder diffère (poids). On désigne explicitement les profits générés par cette manière:

$$p_{ij} = \mathcal{R}\left(\mathcal{U}\left(p_1^{\min}, p_1^{\max}\right)\right). \tag{10}$$

Ici, $\mathcal{R}$ signifie Reproduire le premier vecteur de profit généré pour d'autres classes. La deuxième façon d'appliquer les fonctions de production est de prendre en compte l'indice de classe i au moment de décider de l'intervalle à partir de laquelle les valeurs sont prises pour chaque classe, *e.g.* $\mathcal{U}(10(i-1), 10i)$ ou $\mathcal{L}(10(i-1), 10i)$. Lorsque la fonction génératrice uniforme est appliquée de cette façon, les profits dans chaque classe sont toujours choisis aléatoirement, mais les profits des différentes classes sont dispersés dans différents intervalles. Bien que la fonction génératrice linéaire est appliquée, les profits sont linéairement attribués à des intervalles différents. On note cette application particulière des fonctions génératrices comme:

$$p_{ij} = \mathcal{C}(F), \tag{11}$$

où F est une fonction génératrice avec des paramètres différents pour différentes classes et $\mathcal{C}$ signifie que la fonction est dépendante d'un classe.

Générer les Poids

Pour générer les poids, on peut appliquer une certaine corrélation sur la fonction génératrice pour chaque dimension. En particulier, nous définissons les fonctions génératrices suivantes.

Fonction Génératrice Non-Corrélée Dans une fonction génératrice non-corrélée, nous attribuons simplement les poids de manière uniforme et aléatoire dans un intervalle:

$$w_{ijk} = \mathcal{U}\left(w_{ik}^{\min}, w_{ik}^{\max}\right). \tag{12}$$

Fonction Génératrice Faiblement Corrélée Cette fonction génératrice est motivée par des résultats sur les instances du KP [Pis05]. La motivation est de générer les poids corrélés aux profits, mais toujours avec un degré de liberté pour chaque dimension. Dans notre proposition, les poids sont attribuées en fonction de:

$$w_{ijk} = \mathcal{U}\left(\max\left(0, p_{ij} - \frac{p_i^{\max}}{\delta}\right), p_{ij} + \frac{p_i^{\max}}{\delta}\right). \tag{13}$$

Nous allons utiliser la notation suivante:

$$w_{ijk} = \mathcal{W}(\delta). \tag{14}$$

Fonction Génératrice Fortement Corrélée Fonction génératrice fortement corrélée est également motivée par les résultats précédents où la corrélation entre les profits et les poids est forte:

$$w_{ijk} = p_{ij} + \frac{p_i^{\max}}{\delta}. \tag{15}$$

Nous utilisons la notation suivante pour cette fonction:

$$w_{ijk} = \mathcal{S}(\delta). \tag{16}$$

Fonction Génératrice Fortement Inversée Corrélée Pour une fonction génératrice fortement inversée corrélée, les poids sont attribuées en fonction de:

$$w_{ijk} = p_i^{\max} - \frac{p_{ij}}{\delta}, \tag{17}$$

et sera dénommée:

$$w_{ijk} = \mathcal{I}(\delta). \tag{18}$$

Notez que la fonction génératrice fortement inversée corrélée n'est pas intéressante pour être utilisé seule. Les cas intéressants se produisent lorsque les deux fonctions, fortement corrélée et fortement inversée corrélée coexistent sur les différentes dimensions de poids. Intuitivement, ces cas sont difficiles à résoudre parce qu'un choix attentif entre les poids à travers plusieurs dimensions doit être fait. Bien que nous n'avons pas de connaissance de problèmes réalistes de ce type de MMKP, ils sont encore intéressants d'un point de vue théorique.

Application des Fonctions Génératrices Similaires aux fonctions génératrices pour les profits, on pourrait appliquer la même fonction génératrice avec les mêmes paramètres à toutes les dimensions. Mais il est également possible d'appliquer la même fonction avec des paramètres différents ou même des fonctions différentes pour les dimensions. En plus d'appliquer simplement la même fonction génératrice avec les mêmes paramètres sur toutes les dimensions, ici nous vous proposons deux façons d'appliquer les fonctions génératrices pour les poids. Le premier est d'inclure l'indice de dimension k comme un paramètre dans la fonction génératrice. Par exemple, pour une fonction génératrice non-corrélée, les poids d'une dimension k peuvent être choisis dans un intervalle qui dépend de k, ou pour une fonction faiblement, fortement et fortement inversée corrélée, le paramètre δ peut être choisi en fonction de k. Il est commode d'utiliser une notation comme suit:

$$w_{ijk} = \mathcal{D}(F), \tag{19}$$

où F peut être, par exemple, $\mathcal{U}(1, 10k)$ pour une fonction génératrice uniforme, ou $\mathcal{W}(k+5)$ et $\mathcal{S}(k+5)$ pour une fonction génératrice faiblement corrélée et fortement corrélée, respectivement. Ici, $\mathcal{D}$, signifie que les fonctions génératrices sont Dimension-dépendantes. Nous pourrions également appliquer différentes fonctions

génératrices pour différentes dimensions. Par exemple, nous allons générer des instances avec la fonction génératrice fortement inversée corrélée pour quelques dimensions et la fonction génératrice fortement corrélée pour les autres. Dans ce cas, on note:

$$w_{ijk} = \mathcal{D}(F_1 F_2 \dots), \tag{20}$$

où $F_1, F_2, \dots$ sont les fonctions génératrices que nous utilisons.

Générer les Capacités du Sac à dos

Enfin, les capacités du sac sont générées pour les classes et les dimensions. Nous générons une série de S instances et l'instance h a une capacité c_k^h pour dimension k, où $h = 1, 2, \dots, S$. La capacité est dispersée à dans un intervalle à partir du poids minimum et maximum:

$$c_k^h = \frac{h}{S+1} \left(\sum_{i=1}^{m} w_{ik}^{\max} - \sum_{i=1}^{m} w_{ik}^{\min} \right) + \sum_{i=1}^{m} w_{ik}^{\min}. \tag{21}$$

Le paramètre h sera également appelé le *niveau de capacité* des instances de la série. Notez que tous les S instances n'ont pas besoin d'avoir les mêmes éléments (les profits et les poids). Toutefois, afin d'enquêter sur l'impact du niveau de capacité sur le temps de la solution, nous générons tous les S instances dans la même série avec le même profit et les valeurs de poids. En conséquence, les instances dans une série diffèrent les unes des autres que par leurs capacités.

Plusieurs catégories des instances de MMKP ont été produites pour montrer les instances difficiles. Des expériences sur ces instances difficiles avec un algorithme exact et les outils ont également révélé une structure particulière du problème. Brièvement, l'instance est difficile à résoudre lorsque toutes les classes contiennent le même profit et les poids sont corrélés avec les profits. D'ailleurs, une dimension de poids fortement corrélée est en mesure de rendre les instances difficiles. Enfin, certaines catégories des instances sont très difficiles pour l'algorithme BBLP et les outils GLPK et CPLEX, même de nombreux algorithmes avancés de branchement et coupe sont employés par les deux outils génériques. Les propriétés structurelles spéciales des instances méritent une enquête plus approfondie.

Conclusion

Les réseaux de capteurs sans fil deviennent une réalité. Le déploiement à grande échelle qui fournit l'information au site en temps réel pour les utilisateurs mobiles pourrait être envisagé dans l'avenir proche. L'accès direct des utilisateurs mobiles dans les nœuds de capteurs simplifie l'architecture du réseau et est capable de limiter le trafic en local, c'est critique pour un réseau évolutif. L'équité et l'efficacité doivent être simultanément pris en compte pour optimiser le fonctionnement de ces réseaux. Alors que la plupart des études de littérature mettent l'accent sur les nœuds de capteurs, notre vision est que l'équité des utilisateurs est particulièrement importante lorsque les utilisateurs sont des clients du service fourni par le réseau. Suite à cette

vision, nous étudions des questions d'équité dans le réseau de capteurs sans fil du point de vue d'un utilisateur.

Nous avons identifié et étudié le problème d'allocation équitable des requêtes pour un WSN avec utilisateur mobile. Plusieurs questions connexes, *i.e.* la capacité des réseaux sans fil ad-hoc et des réseaux de capteurs sans fil, la couche MAC et la couche réseau pour les réseaux de capteurs sans fil, les caractères du problème sac à dos multidimensionnel aux choix multiples et ses algorithmes et les définitions de l'équité sont brièvement étudiés. Cette partie de l'étude sondage nous a fourni une bonne connaissance sur la base de laquelle les aspects suivants du problème de l'allocation de requête ont été étudiés.

(i) Le problème d'allocation de requête à l'équité Max-Min dans un WSN est défini et discuté. L'analyse est basée sur un modèle de requête continue et un modèle de trafic. En vertu de ces hypothèses, la région de requête d'un utilisateur est limitée par la bande passante des capteurs et des utilisateurs. Ainsi, les utilisateurs ont à coopérer avec les capteurs pour atteindre les résultats souhaités. L'expression explicite de requête à l'équité max-min, pour le cas où seulement deux utilisateurs existent dans le réseau, est dérivée. Et le problème au cas où plusieurs utilisateurs sont dans le réseau est résolu avec un algorithme heuristique distribué. Nos simulations montrent l'efficacité de l'algorithme proposé.

(ii) L'allocation équitable des requêtes entre les utilisateurs est également étudiée avec un modèle discret. Dans ce cas, la valeur discrète du rayon de la requête ne promet plus de l'existence d'une solution équitable max-min. Ainsi, l'équité max-min lexicographique doit être exploitée. Nous avons également constaté qu'il est commode de représenter le problème de l'équité max-min lexicographique sur le MMKP. En outre, l'objectif traditionnel de l'optimisation, ce qui maximise la somme de l'utilité de tous les éléments dans le sac, pourrait également être utilisé pour maximiser la somme du rayon de requêtes des utilisateurs. Sur la base de ces observations, nous proposons une formulation unifiée pour les deux problèmes. Cette formulation, d'une part, est en mesure de formuler différents problèmes d'optimisation de l'origine d'un WSN avec utilisateur mobile, et d'autre part, implique la mise en œuvre d'un algorithme uniforme et simple pour résoudre les deux problèmes. Les simulations ont été menées pour évaluer la performance de l'algorithme. Différentes propriétés entre les deux objectifs d'optimisation sont discutées.

(iii) Pour que l'étude ci-dessus soit pratiquement significative, nous étudions la faisabilité de reformuler le problème et mettre en œuvre nos solutions dans un WSN basé sur IEEE 802.15.4/ZigBee. Le mode d'arbre du ZigBee est envisagé en raison de son efficacité. En outre, l'attribution des adresses et la routage utilisée par l'arbre ZigBee nous permettent un calcul plein localisé. L'algorithme distribué que nous avons proposé est efficace pour approcher de la solution optimale et pour contrôler la congestion.

(iv) Le MMKP a été utilisé pour formuler le problème d'allocation de requête pour un réseau de capteurs sans fil avec utilisateur mobile avec un modèle de requête discrète. De nombreuses expériences ont montré des propriétés particulières des instances du MMKP, *i.e.* le temps utilisé pour résoudre des instances peut varier beaucoup tel que les instances plus petites prennent beaucoup plus de temps que

les grandes. Dans la dernière partie de cette thèse, nous avons étudié cette question par expériences. Une méthode systématique pour générer les instances MMKP est proposée et plusieurs groupes d'instances qui représentent une variété de types de corrélation entre les paramètres du problème sont générés. Ces instances sont testées avec l'algorithme BBLP ainsi que deux outils d'optimisation, le GLPK et le CPLEX. Les résultats montrent que les profits linéaires et la corrélation forte entre le poids et le profit font des instances très difficiles pour l'algorithme BBLP et les outils, même si certains mécanismes avancés de la programmation en nombre entier sont intégrés dans les deux outils.

Limites et Perspectives

L'étude de cette thèse a été basée sur la vision que le WSN sera déployé à grande échelle pour fournir des services en temps réel à de multiples utilisateurs mobiles qui sont en mesure d'accéder directement aux nœuds de capteurs. Outre les problèmes d'équité étudiés dans cette thèse, il existe plusieurs autres questions discutables de cette architecture du réseau très particulière.

Applicabilité La première question est de savoir si oui ou non le WSN avec utilisateur mobile est avéré utile et sera déployé. Depuis les expériences des applications déployées, de nombreux réseaux de capteurs sont d'une taille très limitée en ce qui concerne le nombre de nœuds (*e.g.* une dizaine de nœuds) et la zone géographique qu'ils couvrent (des centaines de mètres carrés). D'un point de vue technique, les difficultés majeure pour un grand réseau de capteurs sont la connectivité et l'évolutivité. Alors que de nombreux chercheurs sont occupés à résoudre ces problèmes, d'autres remettent en question les besoins d'un grand réseau connecté en permanence. L'argument c'est que de nombreux petits WSN indépendants réunis ensemble seront suffisants pour des services omniprésentes. Avec ce débat à l'esprit, nous trouvons que le WSN avec utilisateur mobile peut satisfaire les deux parties. D'une part, il pourrait s'étendre à un réseau avec un grand nombre de nœuds couvrant une zone géographique très vaste, tandis que d'autre part, il n'a pas besoin d'être entièrement connecté comme les utilisateurs mobiles sont toujours en mesure de récupérer des données de nœuds de capteurs qui sont suffisamment proches. En conséquence, nous avons un fort sentiment que le WSN avec utilisateur mobile au moins offre une voie prometteuse pour de futures applications omniprésentes de détection.

Réseau ou Service? Une autre question intéressante est devrions-nous séparer ou intégrer le réseau avec le service? Traditionnellement, les réseaux de capteurs sont responsables de la collecte des données brutes (probablement avec un traitement très limité dans le réseau), tandis que les données sont fournies aux utilisateurs via un serveur back-end. En conséquence, la plupart des recherches sont de l'aspect réseau sans tenir compte des utilisateurs. Nous avons étudié l'aspect d'utilisateurs avec des limites du réseau comme des contraintes. Bien que l'architecture WSN avec utilisateur mobile est originale et réalisable pour les applications où la signification

des données recueillies ont une forte dépendance à l'égard spatial et temporel, *i.e.* les données sont significatives pour une courte période de temps et fortement liée à l'endroit où les données sont récupérées et elles ne doivent pas être stockés en général, il est *infaisable* pour les scénarios où les données doivent être conservées. Pour ce dernier cas, néanmoins, il est possible de soutenir les utilisateurs mobiles avec une architecture traditionnelle.

Multicast et Traitement dans Réseaux Notre formulation du problème et les algorithmes fonctionne avec un routage du plus court chemin ou un routage avec arbre hiérarchique et nous ne considérons que le routage unicast avec aucun traitement dans les réseaux. En réalité, beaucoup de routages ou protocoles de collecte de données existent et certains ont un mécanisme inhérent de l'agrégation de données afin de réduire le trafic. Pour les protocoles de routage multicast qui envoient une seule copie de données vers des destinations multiples, ou des protocoles de routage avec l'agrégation dans les réseaux, notre modèle n'est plus valide. En conséquence, un nouveau modèle du trafic est nécessaire en vertu de cette situation qui pourraient être intéressante pour les travaux futurs.

Bande Passante Variable Nous avons basé notre étude sur les résultats théoriques de la capacité du WSN qui ont leurs propres limites. Notamment, les résultats sont obtenus comme une valeur asymptotique lorsque le nombre de capteurs dans une zone déterminée devient infini. En réalité, la bande passante effective entre chaque paire d'émetteur et le récepteur dépend de nombreux facteurs, *e.g.* le régime de la modulation, la puissance de transmission, l'état de transmission des paires de proximité, le mécanisme d'accès au médium, *etc.* La formulation actuelle du problème avec une bande passante constante partagée n'est pas capable de tenir compte de ces facteurs. Alors que le modèle complexe est nécessaire pour gérer plus de détails, les solutions actuelles pourraient être étendues, sans trop d'effort. La mesure proposée dans cette thèse est capable de s'adapter à la dynamique de la bande passante. Étudier la performance des algorithmes d'allocation dynamique de requête sous situation de la bande passante est une autre perspective sur laquelle le travail actuel pourrait être prolongé.

Mobilité Même si nous avons beaucoup insisté sur les utilisateurs mobiles, nous n'avons pas abordé les problèmes posés par la mobilité dans cette thèse, *i.e.* tous les modèles, l'analyse, les algorithmes sont basés sur des capteurs et les utilisateurs statiques. Cela empêche une application directe des mécanismes proposés à des scénarios réels. Pour que cette étude soit plus pratique, nous allons étendre nos propositions à un environnement utilisateur mobile, qui peut ne pas être trivial lorsque la convergence des algorithmes distribués doit être mise en œuvre.

Sécurité L'accès direct des utilisateurs mobiles aux nœuds de capteurs peut soulever des questions de sécurité. De toute évidence, un contrat de service doit être exécuté entre les utilisateurs et l'opérateur du réseau. Une authentification simple protégée

par un mécanisme de cryptage léger doit être mise en œuvre à la fois pour les capteurs et les utilisateurs.

Pourquoi MMKP est Dur? Enfin, compte tenu du problème MMKP lui-même, l'étude empirique présentée dans cette thèse n'est qu'une première étape vers la compréhension de ses propriétés structurelles. Nous avons montré que les profits linéaires et corrélation forte entre les profits et les poids font une instance de MMKP dure, des recherches plus approfondies sont nécessaires pour dire pourquoi. Bien que l'algorithme exact BBLP et le GLPK et CPLEX sont des outils utilisés pour montrer ces instances difficiles, des expériences supplémentaires de ces instances avec les heuristiques existantes peuvent fournir plus d'indices à la question mentionnée ci-dessus. Bien que certaines expériences ont déjà montré que certaines heuristiques sont parfois très efficaces pour résoudre des instances difficiles avec des propriétés spéciales, ce sujet mérite une étude en profondeur.

Abstract

A Wireless Sensor Network (WSN) is a wireless network consisting of spatially distributed sensor nodes to gather information from the physical world. The promising idea of WSN is to build an information gathering, processing and representing system that connects human with the physical world. Lots of efforts have been made in recent years on both the theoretical and applicative aspects of WSNs and we can expect that large scale WSN deployment will be possible in the near future. This thesis focuses on the large scale WSN which may serve potentially many users in an open architecture where each user has direct contact with the sensors. We will refer to this architecture as the mobile user WSN. Large scale mobile user WSN provides a promising solution to many applications. Multiple users in the network could act as data collectors working together for a common task or they could also be end users which have no direct contact between each other. In either case, fairness among these users is an important issue in practice.

We first investigate the fairness issues with a simple query model. In this model, multiple users query the sensors located within a circled area. The query circle is assumed to have a continuous variable diameter and centered at the querying user. Distributed optimization problem with congestion constraints is formulated and heuristic algorithm is developed to approximate the optimal solution. Next, a similar problem with a discrete query model is further investigated. In the discrete query model, the query range is measured by hop numbers. This variation makes the problem to be combinatorial and NP-hard. Multidimensional multiple choice knapsack problem is used to model the problem with both lexicographical max-min fairness and maximal query cover objectives. Distributed solutions are proposed and their performance are demonstrated by simulations. Since the ZigBee specification and IEEE 802.15.4 standard are two *de facto* standards of the wireless sensor networks, we study our problems for a wireless sensor network based on such technologies. Special properties of the ZigBee cluster tree structure are exploited to keep the algorithm fully local thus only limited communications are involved in the proposed distributed algorithms. Efficiency of the algorithms are demonstrated by extensive simulations.

While all the subjects discussed so far are related with the fair capacity sharing problem found in wireless sensor networks with multiple users, the combinatorial problem behind the discrete version, namely the multidimensional multiple choice knapsack problem, is very interesting and deserves special research efforts. The inconsistency of the solution times of instances with similar parameters and the sharp contrast between hard small problems and easy large problems motivated our further investigation on the problem itself. By first proposing methods to generate problem instances with different properties, we try to solve several groups of instances with the current algorithm/solvers. Special properties that make the instances hard have been identified.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Histories make men wise; poems, witty; the mathematics, subtle; natural philosophy, deep; moral, grave; logic and rhetoric, able to contend."

– *Francis Bacon*

In this chapter, we first motivate our study by providing a brief retrospection on the evolution of the wireless sensor networks, with special emphasis on their network architecture driven by the application scenarios. Then the main objectives and contributions of the thesis are introduced, followed by a brief outline of the thesis organization at the end of this chapter.

1.1 A Brief History

Wireless Sensor Network has emerged from the traditional sensor network. The first research on the sensor network originated during the cold war, when sensor systems were developed and used for airspace and under water surveillance for hostile aircrafts and submarines. At that time, the major techniques consist of radar and sonar and the whole system consists of multiple sensory units (radar or sonar) connected to a single central processing facility. From the 1980's, sensor system has found its way to evolve towards a distributed network system. A representative is the Distributed Sensor Network (DSN) project sponsored by the Defense Advanced Research Projects Agency (DARPA) of the U.S. This project has identified four key technologies for the sensor network: the sensor technologies, communication technologies, processing algorithms and distributed software technologies. These key technologies have defined the sensor network to be a multidisciplinary research field. As sensor network has its deep root in the military applications and the Department of Defense (DoD) of the U.S. realized its importance in the future battlefield, most of the research works were sponsored by the DoD of the U.S. Only after the year

2000, with the rapid advances of two key technologies: one is the Micro-Electro-Mechanical System (MEMS) which has greatly reduced the size and price of sensors of various sorts and the other is the wide application of the Very-Large-Scale Integration (VLSI) which has made smaller and more powerful processors and wireless transceivers widely available, the sensor network paved its way to a fully networked and autonomous system and finally gained its current name as the Wireless Sensor Network (WSN).

A famous WSN project sponsored by the DARPA was the SensIT project and many research agencies were funded under this huge framework. For example, the PODS project of the University of Hawaii, the sensor network research group of the University of Wisconsin-Madison and the Self-Organizing Sensor Networks project from the University of Auburn, *etc.* The main objectives of the SensIT project are the networking technologies and networked information processing technologies for the wireless sensor networks. The former emphasizes more on the autonomous and dynamic network formation while the latter aims on a more intelligent network with information processing capability.

At the meanwhile, more academic oriented researches had been carried out and the industrial partners started to join. The Center for Embedded Networked Sensing (CENS) has been setup at the University of California, Los Angeles by the National Science Foundation (NSF) of the U.S. The CENS not only carries out theoretical works enabling better WSNs but also advocating civil applications of the WSN. Besides these efforts, some companies such as Intel, Crossbow, Moteiv *etc.* started producing their own WSN platform, which generally includes a set of sensor nodes, a gateway node and software package facilitating the development. Their efforts have greatly helped both the researchers from the academic point of view and the developers from an industrial point of view.

1.2 WSN Evolution

Currently, the wireless sensor network is considered as a promising method for many applications including both the traditional military applications and the newly emerged scientific and civilian ones. For the former, these include battlefield surveillance, country border monitor, mobile target tracking, *etc.* For the latter, environment monitor, volcano monitor and wild animal or plant study are typical scientific application areas, while more applications such as to monitor architecture safety, machine malfunction, health of people and the motor traffic, or to facilitate during hazard such as earthquake, flooding or fire. Obviously, one is unable to enumerate all these applications as they are expanding rapidly to many areas where traditional (wired) sensors are employed for monitoring purposes.

This diversification of WSN applications has provided an evidence about how the WSN is evolving and shed light on what future WSNs will look like. Actually, we notice that the primal applications of the WSN in all of the three mentioned areas are still limited to highly specialized domains. In such applications, the wireless sensor network is demanded, designed, deployed, used and maintained all by spe-

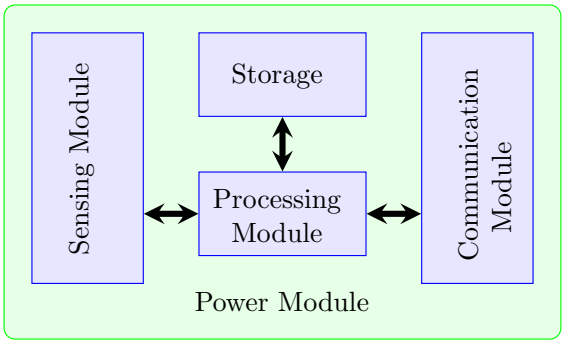


Figure 1.1: Sensor node components.

cialists. These WSNs are highly specialized by all aspects *e.g.* sensors are carefully selected, functional modules of both hardware and software are tailored, devices are manually deployed according to application requirements, wireless transmission parameters are adjusted in combination of the deployed position to form a network. However, an overview on all applications which are really deployed nowadays does give us some signs that the WSN has already opened its way to a low cost, easy to use network providing versatile services to the general public. In the vineyard application [BBB04], the design and deployment phases are already made general enough to support various situations and their experiences show that users without any knowledge about the wireless sensor network are able to operate and make use of such a network.

These observations recalled us the evolution of computers which has demonstrated a very similar trend: from centralized to paralleled or networked processing, and from specialized developer and users oriented to a standard office equipment and a kind of consumer electronics used around the world. If this trend continues for the WSN, it is possible that the WSN will be applied to almost all perspectives of our society in the near future, providing us many useful information including not only the basic parameters of the physical world we are living in, but also any event and state that we are interested in. We believe this trend will continue and finally lead to massively deployed interconnected networks of sensors and people will be able to retrieve reliable information everywhere.

Generally speaking, wireless sensor network is a data gathering and processing wireless network consisting of many sensor nodes and one or more sink nodes. The sensor nodes should be able to carry out some basic functionalities such as sensing the ambience environment, wireless transmission and reception, storing the sensed data and computation. Several modules are generally needed to fulfill these tasks: the sensing module, the processing module, the communication module, the storage and finally the power module as illustrated in the Figure 1.1. On the other hand, the sink node acts as a gateway connecting the networked sensors with the outside world where the back-end application server is located. Data is sent to the server for processing and then provided to the final users via the sink. While on the other direction, commands towards the sensor nodes are also forwarded through the sinks. These commands may include query, configuration and re-programming.

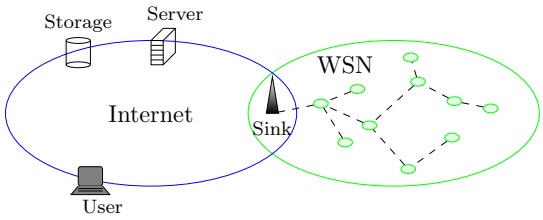


Figure 1.2: Fixed sink wireless sensor network.

Wireless sensor network design depends heavily on its application. Actually, the selection of on-board sensors, size of the sensor nodes, type of the sink nodes, the communication mechanism, the network structure and the related software should be carefully selected to meet the special requirements of the application. We will focus only on the networking aspect of the whole system and identify three typical WSN network architectures. While the first two have found their applications in the real world, the third one is just emerging.

Fixed Sink WSN In some military and scientific WSN applications, the sensors are installed manually at carefully designed positions to form a static network. A sink node is also installed at a fixed position and data is sent from the sensors to the center server through the sink. We call this network architecture the *fixed sink WSN*, as demonstrated in Figure 1.2. An obvious problem of the fixed sink WSN is the bottleneck formed around the sink, as a result of the aggregated data from all sensors. This bottleneck effect will drain the energy on the sensors around the sink much faster than other sensors and finally when these sensors die out of energy, the rest part of the network is separated from the sink. Installing more sinks in the network may alleviate the bottleneck effect, however, doing so increases both the investment of the network and the complexity of the network. In fixed sink WSN, the client or user access the services provided by the WSN through the back-end server.

Mobile Sink WSN Under certain circumstances, it is impossible or unnecessary to install each sensor node to an exactly controlled position and sometimes the sensors may not be able to form a fully connected network towards the sink. For example, sparsely deployed sensors are enough for the designed sensing task thus the distance between sensors may be beyond their maximum transmission range. In contrast, a connected network is unnecessary or highly expensive. Under this case, researchers have proposed to use some mobile devices to collect the data from the sensors when the devices move within the transmission range of the sensors. We name these mobile devices the mobile sinks and the corresponding network architecture the *mobile sink WSN*. The mobile sinks are also known as the *data mules* or *data collectors* in the literature. The data generated by the sensors is stored temporarily before a sink moves nearby, then transfered to the mobile sink, brought to the server for processing and finally provided to the users. In the mobile sink WSN, the mobile sink is a part of the network deployment, designed, deployed

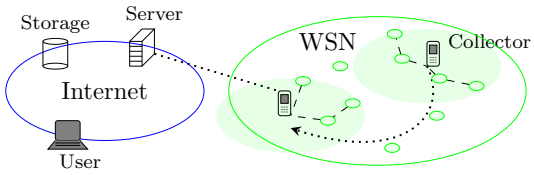


Figure 1.3: Mobile sink wireless sensor network.

and managed by the WSN operator. Mobile sink WSN alleviated the bottleneck effect found in the fixed sink WSN. Figure 1.3 serves as a brief illustration of the mobile sink WSN architecture.

Mobile User WSN In the future WSN applications serving the general public, the number of potential users of the network will be very large. Current network architectures with the fixed sinks or the mobile sinks may encounter scalability difficulties as the service is finally provided to the users via back-end servers. Furthermore, the users of such kind of general public services usually are interested in certain information about their ambient environment, *i.e.* the users are interested in what is happening nearby. This kind of in-site information may include temperature, noise, parking slot, traffic intensity, availability of certain product in the supermarket, *etc.* Under this case, a more efficient information retrieval scenario could be letting the users communicate directly with the surrounding sensor network and gather information with only interested types from only the sensors around them. We name this architecture as *mobile user WSN*, as illustrated in Figure 1.4. We emphasize also that the mobile user WSN allows the users to enjoy more freedom when querying the sensors and for the queries covering a large geographical area, multi-hop data forwarding is necessary.

Mobile user WSN has many advantages especially for certain types of applications compared with the previous two architectures. On one hand, many types of information only have real-time value thus there is no need to store them for further processing or for auditing purpose. Therefore, the data can be provided to the users directly without being forwarded to the back-end server first. On the other hand, the sensors under this scenario could be triggered only by a request from the user. If there is no query to process, the sensors would stay at low-power mode. Besides, communication happens only between users and the sensors around them, sensors not involved in the query are not affected. In contrast, fixed sink WSN sensors with no sensing tasks may also have to communicate in order to help forwarding data or command for others, unnecessarily consuming energy.

Mobile user WSN differs from the fixed sink or mobile sink WSNs in a very important way, *i.e.* the sinks in either fixed sink WSN or mobile sink WSN are not the final consumer of the information generated by the sensors. They act on behalf of the operator of the WSN collecting data from the sensors, forwarding the data to back-end servers, bridging commands from the servers to the sensors. The device in use is selected at the design time, their query scenario is pre-programmed, their position or trajectory is planned, all by the operator. Therefore, the resulting

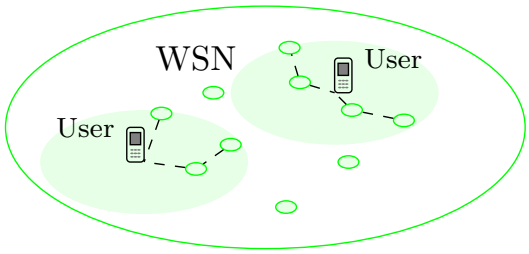


Figure 1.4: Mobile user wireless sensor network.

network is a closed system providing services only through some back-end servers. In contrast, the mobile user WSN architecture is more open to the users in a way that the users may gather data directly from the sensors with their handhold devices such as cell phones or PDAs. However, it should be noticed that the mobile user WSN and the other two architectures are not mutually exclusive. Instead, mobile user can co-exist with either a fixed or mobile sinks, resulting in a hybrid and more versatile architecture.

1.3 Motivations and Objectives

This thesis has been motivated by the large scale, general public service oriented, in-site information gathering applications and the resulting mobile user WSN architecture. These applications are gaining more and more interests from both industrial and academic organizations. An architecture with multiple mobile sinks has been proposed in [CM06]. In this architecture, cell phones are equipped with multiple wireless interfaces: one for basic mobile network communication and the other for communicating with other devices via short distance wireless links such as Bluetooth and ZigBee. Based on these multi-radio enabled cell phones, it is possible to provide some useful services to the general public. We highlight several of them here with only their basic scenarios and most distinct characteristics.

Pervasive Information System Wireless sensors could be deployed along the city streets and the parking places so that the drivers may have access to real-time information about the traffic ahead and the free parking slot nearby with their cell phones. Apparently, one usually cares more about the traffic information about a small region ahead towards his destination, or the parking information around the destination.

Counter-emergency Facilitating System Another potential application domain for the mobile user WSN could be the emergency operations. Wireless sensor nodes could be deployed within and around a fire site to facilitate the operation. With handhold devices, the firemen are able to gather various information from sensors around them in order to make wise actions so that they can keep themselves from dangerous explosions, avoid being trapped or locate the victims.

We emphasize several characteristics of these kinds of applications and the resulting mobile user WSN as follows:

- The wireless sensor network under these cases is more user-oriented than traditional military or scientific applications. For the pervasive information system mentioned above, the users gain access to the services likely through a contract, thus becoming clients of the network. As a result, it is necessary for the service providers to satisfy their clients. For the counter-emergency facilitating systems, to satisfy the basic needs of each single agent is important although it is more likely that they belong to the same organization.

- The users are not managed by the network operators. While it is true that the devices in use must meet some functional requirements and certain contract between the users and the operator is needed, the detailed user profile such as access pattern, position, *etc.* could not be controlled by the operator.

- The users are autonomous and they generally do not have direct contact with each other. On the contrary, they usually compete with each other for the network resources.

Under such an autonomous environment with strict resource constraints, multiple users have to share the common resources in a satisfying way in order to operate the network in an optimal state according to the user population and geographical distribution. Especially, one would like to prevent certain users from being starved only because they enter the network after other users have consumed all available resources. This requires to investigate a series of optimization problems which are the central concerns of this thesis.

This thesis aims at the optimization problems originated from the mobile user wireless sensor network defined previously. The problems will be formulated with mathematical programming and then solved analytically and algorithmically. Two optimization objectives will be investigated, one is to maximize the satisfaction of all users in the network and the other is to give each user a fair share of service provided by the network. Both optimization objectives emphasize more on the users' side. This user centric point of view is considered one of the novelty of this thesis. When the network is supposed to provide services to its clients, to optimize client side parameters such as their query range makes sense. As a result, sensor side parameters that are pervasively considered critical in the current literature, such as the power efficiency and life time, are of subordinate importance in this thesis. We consider the resource limitations on the sensor nodes only as constraints in formulating the optimization problems and the particular resource under consideration is the effective bandwidth of the sensor nodes.

Obviously, for an open network architecture such as the mobile user WSN we are studying, any practical solution must be capable to adapt to the network dynamics, *e.g.* user join or leave, position change, amount of information required, *etc.* This demands basically a distributed solution that enables the cooperation between the users and the sensors to achieve certain optimization objective. Therefore, this

thesis will focus mainly on the distributed solutions of the optimization problems formulated.

In summary, this thesis aims to propose distributed algorithms for the users and sensors in a mobile user wireless sensor network to collaboratively allocate an optimal query range for each user in an adaptive and scalable way.

1.4 Contributions

The major contribution of this thesis is that we identified, formulated and solved the multi-user resource sharing problem in the emerging mobile user wireless sensor networks. This consists of four aspects.

As the first step, the problem is formulated as a nonlinear optimization problem. Max-min fairness is employed as the optimization objective. The formulation is based on a disc-shaped query model where the sensors within a disc-shaped area around a querying user are assumed to send data at a constant bit rate to the user. As a direct result of this query model, the traffic load on the sensors could be analyzed with the method proposed in [LH05]. This model allow us to obtain an explicit expression of the max-min fair query range allocation for the simplest network setting where only two users are considered. Based on the solution to the two-user network, we developed a distributed algorithm to allocate max-min fair query to multiple users in the network. The core idea of the distributed algorithm is to let users which are competing for common resources calculate fair query allocation between each pair of them. The calculation is based on the two-user results and is able to provide a very good heuristic starting point for further exploring the optimal. The merit of the algorithm is it adapts to a near optimal state very fast after any query modification happens in the network.

Based on this first result, we further investigated another query scenario where the query range is measured by the hop number that the query messages may travel. This small modification in the query model actually turned out to be fundamental in that the optimization problem has to cope with discrete parameters. An immediate consequence is the max-min fairness for continuous parameters used in previous model may not exist. Instead, lexicographical max-min fairness has to be used under this case and the problem becomes combinatorial thus should be solved as integer programming problems. The combinatorial formulation of this resource sharing problem with discrete parameters is considered as a part of our second contribution in this thesis and in particular, we consider two optimization objectives: maximization of all queries and lexicographical max-min fair queries. The combinatorial problem behind is a member of the well known Knapsack Problem (KP) family, namely the Multidimensional Multiple choice Knapsack Problem (MMKP). Distributed algorithms are proposed for the MMKP for the two optimization objectives. The central idea is when the sensors are queried by many users thus about to experience congestion, they solve a local problem to obtain locally optimal solution for the related users, then the users adapt to their new query ranges. Combined with

an adaptive exploration phase, the resulting distributed algorithm is demonstrated to be very effective by computer simulations.

The third contribution of this thesis is that the problem formulation and proposed solution are further extended to the *de facto* standards for wireless sensor networks, namely, the ZigBee specification and the IEEE 802.15.4 standard. In particular, the synchronized tree mode specified by the ZigBee is investigated. This tree mode is specially designed for energy limited scenarios thus is ideal for wireless sensor networks. The most important modification on the previously proposed model, in order to adapt to the ZigBee tree based wireless sensor networks, is how the congested sensor nodes solve their local problems. More precisely, the ZigBee tree mode allows us to estimate the potential traffic load when a query grows larger, thus further enables a sensor to solve the local problem efficiently. In contrast, our previous model assumes a shortest path routing is in use and the solution relies on each sensor to measure the traffic load it is handling.

The last contribution of this thesis focuses on the MMKP itself which has been used in our previous problem formulations. MMKP is a very interesting problem mainly due to two aspects. Firstly, it is a complex problem thus is inherently hard. Actually, it has been proven to be NP-hard in general. Secondly, the MMKP has many real world implications. For example, it has been wildly used to model the adaptive QoS management problem [LLRS99]. Many heuristic and exact algorithms have been proposed in the current literature and they achieved better and better performance. While the performance achievement is quite promising, they are based on numerical experiments with very limited benchmark problem instances. Furthermore, our experiences gained from solving our multi-user resource sharing problem proposed previously suggest that the hardness of certain MMKP instance not only depends on its size, *i.e.* the number of items, classes and dimensions, as widely believed, but also depends heavily on the exact values of the problem parameters, *i.e.* its structure. How the structural properties affect the hardness of the MMKP instance remains unrevealed. Thus, the last part of this thesis intends to make a first step towards showing what kind of structure makes the hard MMKP instances. Two contributions are made in this part: We proposed a method to generate comprehensive benchmark MMKP instances which may facilitate future researches in the area; Then we use it to generate several groups of instances with designed special properties and we test current exact algorithms with these instances. Experiments show instances with similar set of items for classes and with high correlation between profits and weights are hard to solve. Constraint level also impact the hardness of the instances greatly and the impact is shown quite irregular in general. Therefore, a series of instances with constraint level covering the whole span of the solution space should be used when benchmarking the algorithms.

1.5 Thesis Organisation

This thesis is composed of seven chapters organized into three parts.

Chapter 1 and 2 belong to the first part which aims to provide an overview and

necessary background information on the thesis.

Chapter 1, "Introduction", *i.e.* the current chapter, provides a broad description on the context based on which the research is carried out. With a brief illustration of the wireless sensor network history, we identify several major evolutionary steps on the WSN architecture and motivate the thesis by advocating an emerging mobile user WSN architecture which seems to be a better fit to the future applications. Major contributions of this thesis are also highlighted very briefly.

Chapter 2, "Background Knowledge", as indicated by the title, provides some background knowledge which helps to understand how the main technical part is developed and why certain selections and assumptions are made. The background information mainly consists of four aspects. Firstly, a brief survey is given on the advances achieved in the wireless sensor network as a general research and engineering field. Many of the technologies such as media access control, routing, deployment, in-network data processing, *etc.* are related with the solutions that will be proposed in this thesis. Secondly, we present current theoretical results on the capacity of wireless ad-hoc networks, mainly because it is the bandwidth that we are sharing among multiple users. The necessary background knowledge also consists of an introduction to the knapsack problems, in particular its multidimensional multiple choice variant and the related algorithms. Finally, as fairness is a central concern of this thesis, we give a brief overview of some results related with this topic. Several fairness notions are introduced with their applications.

In the second part of this thesis, we formulate the problems, develop the models and propose the solutions.

Chapter 3, "Fair Query Allocation In WSN Under A Continuous Query Model", discusses the resource sharing problem formulation and the motivating application scenario in detail. Continuous query model is employed in the non-linear programming formulation. A simple network scenario is analysed for a max-min fair resource sharing solution and the results are used as a building block of the distributed algorithm.

Chapter 4, "Reformulating the Problem: A Discrete Query Model", investigate the discrete version of the same resource sharing problem. Instead of the max-min fairness, it is necessary to employ the lexicographical max-min fairness as the optimization objective. Besides this, a global maximization objective is also investigated. Multidimensional multiple choice knapsack problem is used in the formulation and distributed algorithm is proposed and evaluated.

Chapter 5, "Extension To A Practical Context: ZigBee Based WSNs", intends to make the problem formulation more specific such that the proposed methods adapt better to a realistic wireless sensor network. ZigBee synchronized tree networking layer and the related IEEE 802.15.4 MAC layer are investigated as they are designed especially for energy constrained networks such as the WSN. While the general problem formulation keeps almost intact, some ZigBee details should be considered in the proposed algorithm. Special techniques enabled by the ZigBee are further exploited. As a result, the algorithm has some nice properties with respect to message complexity and locality.

The third part of this thesis focuses on the MMKP problem itself which is the

core problem behind our problem formulation.

Chapter 6, "The Problem Behind: Empirical Study On The Hard MMKP Instances", offers an empirical study on the property of the MMKP instances, as a reasonable extension from experiences gained in the previous chapters where algorithms solving the MMKPs have been proposed and evaluated. Systematic method to generate benchmark MMKP instances is proposed which provides a better way for evaluating future algorithms. Numerical experiments reveal that there exist many hard instances which should have been considered when benchmarking any algorithm, however not. Experiments also imply that certain properties on the problem parameters contribute to the hardness of the instances.

Chapter 7, "Epilogue", summarizes the whole thesis. Pros and cons of the problem formulation, analysis, solutions and the verifications are discussed. Possible directions on which this work may be extended are highlighted.

Chapter 2

Background Knowledge

"The mechanic, who wishes to do his work well, must first sharpen his tools."

– *Confucius*

This chapter provides necessary background information which helps reading the main technical parts. A brief survey of the wireless sensor network domain is presented in Section 2.1, followed by a discussion of theoretical capacity results on wireless ad-hoc and sensor networks in Section 2.2. Detailed discussions on the algorithms for the multidimensional multiple choice knapsack problem can be found in Section 2.3. Finally, three fairness definitions widely used in the literature are discussed in Section 2.4.

2.1 State-of-the-Art Researches on WSN

From a researcher's point of view, although the WSN is a multidisciplinary research field, an interesting fact is that most of the current research works have focused on the energy efficiency of the sensor nodes. This is because most current sensor nodes are powered by batteries and replace or recharge them for each node after the network is deployed is considered to be impossible or very expensive. A natural objective is to make the network usable as long as possible. This has been the direction of almost all researches. On one hand, supporting hardwares, such as ultra-low power sensors, processors, wireless transceivers and high power density batteries, *etc.*, that enable a sensor node working for months or even years are being invented. On the other hand, software such as operating system, various layers of communication protocols, data processing algorithms, *etc.* which control the hardware in a more intelligent way to save power, are also a very fruitful research field.

In this thesis, we only focus on the software aspect and give a brief survey of some major achievements which have been made recently. We classify these achievements

into supporting techniques and communication protocols, mainly in order to give more emphasis on the latter.

2.1.1 Supporting Mechanisms

Several mechanisms are generally needed to support a wireless sensor network. They include topology control, coverage control, localization, time synchronization and security.

Topology and coverage control These are two highly related problems. On one hand, sensor nodes should be dense enough to cover the monitored area to carry out their designed tasks and sparse enough to eliminate useless redundancy. While redundancy of certain degree is needed for various reasons such as resiliency, thus preserved. How to make the working sensors cover the target area according to the application demand is the core challenge of the coverage problem and this may consist of either a static design problem before the network deployment or a dynamic and autonomous adjustment in an operating network. On the other hand, the sensor nodes have to be interconnected to form a network to retrieve information from the monitored area. Because it is often the case that the sensing range is smaller than the wireless transmission range, the deployment of sensor nodes should first meet the coverage requirement then form an efficient communication network with certain topology control algorithms. The aims of the topology control algorithm include adjusting the wireless transmission power of the nodes to minimize interference, or scheduling redundant sensor nodes to low power mode and back to work when nearby nodes fail. Both operations should always keep the connection requirement of the network. More detailed discussions on the topology and coverage control could be found in [LY06, CW04, CK07].

Localization The problem of localization consists of two different but related aspects. One is how to let each sensor node know its own position and the other is how to know the position of certain event being monitored. Generally, the former is the basis of the latter and emphasizes more on the WSN itself thus, is a fundamental problem frequently investigated. Here we give a brief discussion on this aspect only. Two reasons make the position information important for WSN. Firstly, many WSN applications require some knowledge of where the data is retrieved from. This requires either the geographical position in the real world or a relative position the node represents, *e.g.* node number 1 is mounted on certain position of a building. Secondly, the position information may be required by other protocols such as routing, coverage control and in-network data processing. They sometimes employ position information simplify the protocol design and implementation. However, sensor node localization is a challenging topic. Static binding between nodes and their positions is applicable under very limited situations where the number of nodes is small and the exact position the node is deployed could be controlled. Otherwise, we may need more adaptable and scalable mechanisms to decide for each sensor

node (or a group of sensors) its position. A comprehensive overview on the problem could be found in [MFA07].

Time Synchronization Time or clock synchronization is always an important problem for all distributed computer systems where a global time notion is generally needed to facilitate other protocols. In wireless sensor networks, the data is meaningful only when one knows when it is generated. Besides, some in-network data processing mechanisms require also some notions of time to merge multiple events. One could use an absolute time to label the data and the GPS devices are usually used for this purpose. However, as the GPS device is generally expensive, for a WSN with large number of sensor nodes, it is less cost-effective to equip each sensor node with a GPS. Therefore, many algorithms have been proposed to enable most GPS-less nodes to synchronize their clocks with several GPS-equipped nodes. Besides the absolute time, one has several other time notions such as the network time and even only a sequence number providing very limited information about a relative order of the data generation. Depending on the application requirement, time synchronization could be done at coarse-grained or fine-grained granularities, allowing a trade-off between protocol complexity, energy efficiency and synchronization error. Survey papers of recent advancement in solving the problem could be found in [SY04, YMG08].

Security Securing the wireless sensor networks has recently gained great emphasis as a result of its expanding applications. In more and more of these applications, security is not a plus any longer but becomes a basic requirement. A typical example of such applications would be the health care WSN where a malicious attacker to the system may cause death to the patient. Several standard security mechanisms have been developed and applied to traditional wired or wireless networks. These include encryption, authentication, intrusion detection *etc.* However, because the computation and communication resources are very scarce on the sensor nodes, current mechanisms can not be directly applied to the WSN. As a result, lightweight protocols that consumes very few system resources are under pursuit. Several books on WSN contain chapters on this topic, for example in [RSZ04, ZG04].

Relations to our study The above mentioned topics are related with the query allocation problem which is the central concern of this thesis. We make the following assumptions about these issues throughout this thesis:

- The sensors have necessary and sufficient coverage of the monitored field. This implies also that the data from any single sensor node is necessary thus the network must try its best to deliver the data when requested. We notice a more realistic case is that the generated data has certain redundancy thus some of the data could or should be dropped, or merged in the network. These cases will not be investigated.

- The sensor nodes are assumed to be fully and densely connected to a single network in Chapter 3. Although the dense network assumption allows a suc-

cinct analytical solution of the problem, it is not practical. Therefore, this assumption is relaxed in the following chapters. As a result, it allows the presence of any topology control algorithm which generally makes the network as sparse as possible in order to mitigate the interference, constrained only by the network connectivity.

- We make no assumption about whether the sensor nodes know its position or not. Actually, the shortest-path routing or tree based routing which will be employed by our solutions could be position based or not. Our problem formulation and solution do not require any time synchronization mechanism.

- We do not consider security issues in this thesis although the application scenarios that have motivated this research should be secured in a realistic deployment.

2.1.2 Communication Protocols

Wireless communication is generally considered expensive with respect to its energy consumption compared with other tasks such as sensing and computing on a sensor node. Therefore, energy efficient communication protocols are critical to the life time of the sensor node. In this section, general discussions are given to the wireless communication protocols for wireless sensor networks. Physical, media access control, network and transport layers will be discussed.

2.1.2.1 Physical Layer

Physical layer is responsible for selecting the transmission medium and proper signal modulation, transmission, detection and reception. Radio and infrared are two most employed transmission media by the wireless sensor networks, while others such as optical and acoustical media are occasionally used for special applications. We consider only the radio based technologies in this thesis.

As the radio spectrum is a precious resource and should be shared by all communication activities, its usage is strictly regulated by authorities such as the government. As a result, wireless sensor network, as a newly emerging wireless communication participant, has to co-exist with other existing wireless communication technologies. Currently, most wireless sensor devices employ a wireless transceiver operating at the licence-free Industrial, Scientific and Medical (ISM) bands, within which the two bands with center frequency at 915MHz and 2450MHz are frequently used.

Although many advanced modulation techniques have been developed and practically used for various wireless communication systems, only the simplest ones have been incorporated in transceivers for wireless sensor networks. This is mainly due to the low-power, low-cost requirement of the sensor nodes. As a result, simple modulations such as ASK, BPSK or QPSK, when accompanied by the low transmission power and a low speed micro controller, offer only very moderate data transmission rate.

In this thesis, we do not consider the exact modulation schemes in use.

2.1.2.2 The Medium Access Control Layer

Many medium access control schemes have been proposed for the wireless sensor networks in the literature. Most of them are adapted from the wireless ad-hoc networks and enhanced with energy awareness. Although almost all state-of-the-art MAC protocols such as CDMA, FDMA, TDMA, CSMA have been reported in use for wireless sensor networks, we consider CDMA and FDMA are generally too heavy to fit in the sensor nodes as they require complex signal processing facilities thus results in expensive hardware. In contrast, TDMA and CSMA are simpler to implement thus have been used for many practical WSN applications.

In Chapter 3, we assume an ideal MAC layer that schedules the neighboring nodes in a perfect way that achieves the link capacity. While in Chapter 4, our problem formulation is based on a general MAC layer and the effective bandwidth provided by this general MAC layer is used as constraints in the problem. Finally, one more step to the reality, we build our model based on the IEEE 802.15.4 MAC layer in Chapter 5. Therefore, we now give the IEEE 802.15.4 more detailed discussions.

The IEEE 802.15.4 Standard The 802.15.4 is proposed by the IEEE as a standard for the Low Power Wireless Personal Area Networks (LP-WPAN) [80206].

It requires a physical layer supporting multiple data rate, more precisely defined as follows: 250kbps at 915MHz with O-QPSK modulation and at 2450MHz with O-QPSK modulation, and optionally at 868MHz with ASK modulation and at 915MHz with ASK modulation. An optional 100kbps data rate could be used at 868MHz band with O-QPSK modulation. A data rate of 40kbps should be supported at 915MHz with BPSK modulation and another 20kbps data rate is supported at 868MHz band with BPSK modulation. It defines 16 channels at 2450 MHz band, 30 channels at 915 MHz band and 3 channel at 868 MHz channel.

Star and Peer to peer network topologies are supported by the standard, both requires a *PAN Coordinator*, which shall be a Full Functional Device (FFD). Other nodes can be either FFD or a Reduced Functional Device (RFD). However, a RFD can only talk with a FFD. The FFD could be a *Coordinator* when it carry out certain coordination functions for the RFDs attached to itself. The standard makes use of the standard 64bit MAC address, however, a network can use a short 16bit address when provided by the coordinator, allowing more energy-efficient communications.

The basic MAC mechanism employed by the 802.15.4 is the CSMA-CA channel access, while it supports also the optional Guaranteed Time Slots (GTS). Other physical layer functionalities are also supported such as the low power mode, energy detection and link quality indication, which maybe used by upper layers to facilitate topology control, routing, *etc.*

The 802.15.4 devices should be able to operate in either the *Beaconed* or the *Beacon-less Mode*. In the beaconed mode, the coordinator organizes the transmission/receive as super-frames. A super-frame begins with a beacon frame, followed

by an active period and then an optional inactive period. The inactive period is for the coordinator to go sleep. Within the active period, the time is divided into 16 slots (the beacon itself uses the first slot). The slots can be allocated to a contention access period (CAP) or an optionally Contention Free Period (CFP). The CFP, if presents, goes after the CAP. Within the CFP, the slots are allocated to several Guaranteed Time Slots (GTS), a GTS could use multiple slots. The GTS is mainly designed for low latency applications. In the CAP, the nodes contend for the channel using slotted CSMA-CA. The super-frame structure is defined by the coordinator, and the structure information is sent to the nodes in the beacon frame. Under the beaconed mode, any data communication begins with a beacon frame and the ACKs are optional (on requirement of the initiator). In the beacon-less mode, there will be no super frames, but the beacons still have to be used for network discovery. When the node wants to send data to the coordinator, it sends directly using un-slotted CSMA-CA.

The 802.15.4 devices communicate with each other in either one of the following modes: the *Direct Mode*, *Indirect Mode* or *Peer-to-peer Mode*. The first two are basic communication modes while the third one makes use of the two basic modes. When the node wants to send data to the coordinator, it sends directly. When a node wants to get data from the coordinator, it has to send a request first, then the coordinator send the acknowledgement followed by the actual data transmission. For this scenario, the ACKs are mandatory. In short, direct mode is used for data from RFD to FFD or from FFD to FFD, while indirect mode is used for data from FFD to RFD. As a result, receivers of RFDs will be able to go sleep while those of the FFDs has to keep alive in order to receive the pushed data. For peer-to-peer mode, the nodes can send to any other node within its transmission rage and in this case, un-slotted CSMA-CA should be used.

The 802.15.4 standard defines a set of *Primitives* which could be generally separated into two categories: data service and management service primitives. The execution of these primitives may trigger transmission of certain frames on the wireless channel, or trigger upper layer actions such as message reception, or simply report certain status. The standard also defines four types of *Frames*, namely, Beacon, Data, Acknowledgement and MAC Command. The type of a frame in indicated with the frame type field in the frame.

When the network is beacon enabled, the coordinators shall bound its channel time using a *Superframe* structure separated by beacon frames. The network uses the superframe by setting the BO value chosen from $[0, 14]$ and setting the SO value chosen from $[0, BO]$. The combinations of these two values define different superframe length and beacon interval. Networks do not use the superframe shall set $BO = SO = 15$. The BO and SO may be the most critical parameters affecting the MAC performance.

All the coordinators in the same network should have the same BO values, so as to the SO values. The coordinators have to synchronize their beacon transmission. Actually, a coordinator can send his own beacon only during the inactive period of its coordinator, as a result, synchronization can only be achieved when $SO < BO$.

CSMA-CA is the basic channel access method employed by the 802.15.4. De-

pending on the network is beacon enabled or non-beacon enable, the CSMA-CA could be slotted or unslotted. The difference is in slotted CSMA-CA, all actions should be aligned to the slot boundary while in unslotted case, the actions are taken immediately. Furthermore, slotted CSMA-CA uses CW parameter to control the number of CCA that should succeed before the channel access while unslotted CSMA-CA transmit immediately once the CCA is asserted as clear. Finally, the slotted version can support the battery life extension option.

The 802.15.4 defines several basic functions for the network maintenance. This includes functions like channel scan, PAN ID assignment and PAN ID conflict resolution, MAC address assignment, superframe configuration and so on. A 802.15.4 network is initiated by the PAN coordinator. Devices wants to join an existing PAN should send an association request command to a coordinator, then the coordinator should send a response. If the association is successful, an MAC address will be assigned. Disassociation is done through a similar command and could be initiated by either coordinator or a device.

In conclusion, 802.15.4 defines a CSMA-CA based MAC layer to control the operation of the PHY layer. The MAC layer is further controlled through a set of primitives by the upper layer. 802.15.4 MAC supports both slotted CSMA-CA and unslotted CSMA-CA.

2.1.2.3 The Network Layer

The network layer answers two basic questions: the address assignment and data routing. There is a huge volume of research works on the issues concerning the network layer of wireless sensor networks. Among all proposed mechanisms, we emphasize a category of the shortest-path routing protocols due to their simplicity. It is argued that simple solutions should be used if they have sufficed in practice, and they are really found so [RC08]. The Chapter 3 of this thesis assumes a shortest path routing protocol which has nice properties making the analytical model tractable.

Shortest-path routing could be combined with various addressing schemes. One can make use of geographical position information of the sensor nodes then make position-based shortest-path routing. Under this case, the geographical position is used as addresses of nodes. If geographical position is required also by the application, routing protocol can simply benefit from it without extra cost. For WSNs where geographical position information is not available, shortest-path routing can be realized by routing table managed by each node. This requires a proper addressing scheme to identify each single sensor node and routing messages are necessary.

In Chapter 4, we do not consider the specific routing protocol in use. As a result, the formulation is more general and the solution relies only on the bandwidth constraints and the measured traffic load. We propose a solution to the multi-user resource sharing problem for the ZigBee based WSN in Chapter 5 since the ZigBee is widely considered as a promising standard to be used for future short range wireless communications. A tree based addressing scheme and the related routing mechanism has been incorporated in the ZigBee specification [zig08] proposed by the ZigBee alliance.

The ZigBee Specification ZigBee specifies most of the necessary functionalities on top of the 802.15.4 MAC layer for the LR-WPAN applications, but we consider only its networking related aspects, namely, the addressing and routing functionalities.

The ZigBee defines three network entities: the *ZigBee Coordinators* (ZC), the *ZigBee Routers* (ZR) and the *ZigBee End Devices* (ZED), corresponding to the PAN coordinator, coordinator (FFDs) and devices (RFDs or FFDs that are not coordinators), respectively. The ZC is responsible for the initial setup of the network and may participate in the routing.

Two address assignment mechanisms are specified in the ZigBee specification. The default address assignment mechanism is the *distributed address assignment* based on the tree structure. The basic idea is to let parent assign address or a segment of addresses to its children according to the depth of associated with the device (d), the maximum number of children a device can have (*nwkMaxChildren (Cm)*), the maximum number of routers among these children (*nwkMaxRouters (Rm)*). Note that the maximum number of children, maximum number of routers and the maximum depth of the network (*nwkMaxDepth (Lm)*)are defined by the ZigBee coordinator at the network initiation phase. Then the address block that a router can use to assign to its routing capable children is computed as follows, suppose the router is in depth d.

$$Cskip(d) = \begin{cases} 1 + Cm(Lm - d - 1), & \text{if } Rm = 1 \\ \frac{1+Cm-Rm-Cm\times Rm^{Lm-d-1}}{1-Rm}, & \text{otherwise} \end{cases}$$

A device with $Cskip(d) = 0$ can not accept any child device and is the end device itself. The $Cskip(d)$ is used as an offset when assigning addresses to the children devices. If the device is supposed to be an end device, its address is assigned as follows:

$$A_n = A_{parent} + Cskip(d) \times Rm + n, \tag{2.1}$$

where d is the depth of the parent, n starts from 1. If the children devices are routing capable, the first child is assigned to an address 1 greater than the parent, then the second is assigned to an address $Cskip(d)$ greater than the first, and so on.

The assignment does not enforce any balancing, so the network may use up its address before reaching the furthest devices, leaving them unable to join. One paper deals with this problem and proposed an improvement [PT07].

The other address assignment scheme suggested in the ZigBee specification is the *stochastic address assignment*. Under this case, the parent randomly chooses an address that does not appear in its Network layer Information Base (NIB) and assigns it to its child. The devices will keep an assigned address unless it is notified to change it as a result of a possible address confliction.

The ZigBee network layer (NWK) supports star, tree and mesh topologies, this conforms with the 802.15.4 standard. We emphasize on the tree and mesh network structures which are considered scalable since we consider the WSN where there are usually many devices participating in the network. In the *tree network* structure, devices are organized with a parent-child relationship, addresses are assigned

accordingly and a hierarchical routing is used. Under this case, it is possible to employ the beacon enabled mode of 802.15.4 with a lower duty cycle in order for energy saving. In the *mesh network* structure, as there is no inbuilt way to help identifying the destination as in the tree structure, a table driven routing protocol is specified based on the AODV protocol. Furthermore, under this topology, ZigBee forbids the beacons, so the MAC employs the un-slotted CSMA-CA.

The tree structure is essential because during the network formation procedure, the nodes are associated to a parent node and the addresses are assigned by this parent node, this basically forms a tree. Furthermore, the routing algorithm uses tree routing as its last resort if table based routing fails. The ZigBee routers shall maintain a route table and a route discovery table. A device that does not have a route table can still route its data by using hierarchical routing along the tree, if permitted.

In hierarchical routing, the next hop could be decided by simple calculation based on the destination address and the current node address. If the destination is not a descendent of the current router, the router sends the data to the parent and let it make the decision. Otherwise, the current router has to select one of its children that contains the destination. With the hierarchical address assignment scheme, this could be quite simple. To decide if a destination D is the descendent of the current router which takes address A and at depth d of the tree, the router tests if the following condition is satisfied:

$$A < D < A + Cskip(d-1)$$

If this is true, then the destination is one of his descendent, so the next hop could be given by: $N = D$ for ZigBee end devices where $D > A + R_m \cdot Cskip(d)$, otherwise:

$$N = A + 1 + \left\lfloor \frac{D - (A+1)}{Cskip(d)} \right\rfloor \times Cskip(d).$$

2.1.2.4 The Transport Layer

The central functionalities of a transport layer are congestion resolution and packet loss recovery. Traditional transport protocols used in the wired or wireless networks may not be applied to wireless sensor networks directly. This is either because they do not provide special functions required or because they are less efficient under wireless sensor network environment. For example in the UDP, no reliability or congestion control is promised. And in the case of TCP, its connection overhead is considered unacceptable for variable link qualities which is often the case considering the harsh operating environment of the WSN. Other problems related with TCP consist of the unfairness for faraway sensors, slow end-to-end response time and most importantly, the energy inefficiency.

Congestion Resolution Congestion resolution consists of three parts: congestion detection, congestion notification and rate adjustment.

Several congestion detection methods have been proposed in the current literature, *e.g.* timeout and duplicate ACK as in TCP; queue length has been used

in [HJB04, WEC03]; packet service time has been used in [EB04]; while a complex measurement that combines both the packet service time over packet inter-arrival time has been shown more effective in [WSL+06a]; finally, paper [WEC03] employs the wireless channel load as an indicator of the congestion.

When a congestion state is detected, this information should be sent back to the source, this process is known as congestion notification. Congestion notification could be achieved through several methods proposed in the current literature. One bit congestion state notification in packet header is used in [SOBAA03, HJB04, WEC03]. Allowable data rate notification is used in the solution proposed by [EB04]. A new notion namely the congestion degree is proposed in [WSL+06a].

Rate adjustment can be either AIMD like if only congestion state is provided or it can be accurate rate allocation if more information is available.

Packet Loss Recovery Packet loss recovery consists of three issues similar to congestion resolution. One has to detect the possible packet loss, then notify the sender and finally the sender retransmit the lost packet.

Most frequently used mechanism is to let the sender wait for an acknowledgement message from the receiver. If the acknowledgement is not received before a timer expiration, a packet loss is asserted. Hop by hop packet loss recovery could be achieved in wireless networks by the sender overhearing the next hop node's transmission activity. Employing the broadcast nature of the wireless channel, if the sender does not hear its neighbor node forwarding his packet within a period of time, the packet is considered lost. Finally, packet loss could be detected on the receiver by tracing the packet sequence number.

For the sender based packet loss detection mechanisms, there is no need for a packet loss notification mechanism. For the receiver based detection mechanisms, a special NACK message could be used to notify the sender. On noticing a packet loss event, the sender decides if it is necessary to really carry out the retransmission. Because under certain cases, retransmission may be energy inefficient. Under the regime of wireless sensor networks where the wireless links are usually considered less reliable, end-to-end packet loss recovery may be infeasible as a packet has a large chance to be dropped during the multi-hop forwarding path. Therefore, how to decide if a retransmission is necessary and where the retransmission should be really carried out, *i.e.* where to cache the current packet so it can be retransmitted later, become two challenging issues. By deciding to cache at the source or only at some nodes on the path, we can make a trade-off between efficiency and storage cost.

The main concern of this thesis is the bandwidth sharing between multiple users in the wireless sensor networks. We formulate the problem with bandwidth limitation of the sensors as constraints and these constraints actually require there is no congestion on the sensor nodes. Therefore, our solutions belong to the congestion resolution mechanisms. However, we tackle the problem from the user's point of view. The related works and novelties of our approach will be discussed in more detail in the following chapters where each contribution is presented.

2.2 Capacity of Wireless Sensor Networks

Recent years have seen the burgeon of multi-hop wireless networks where devices with wireless interfaces can be connected to each other in pure ad hoc fashion or through infrastructures and communicate in a multi-hop fashion. As one of the most basic concerns, the capacity of such networks have received much attention from the researchers, especially after the pioneer works done by Gupta and Kumar [GK00].

The main concern of this thesis is to share the bandwidth capacity among the users. Theoretical results show that under the optimal case, the per-node transport capacity of ad hoc networks scales as $O(1/\sqrt{N})$ if independently and randomly distributed traffic is assumed. However, in typical WSN applications, the data traffic are usually directed to the same destination, making the traffic pattern in WSN quite different from traditional ad hoc networks. In [MDMLN03], the per-node throughput capacity was shown to scale as $\Theta(1/N)$. We adopt this result as the basis of our derivation since it meets our assumptions of the network. We make use of the per-node throughput capacity as our capacity notion and we assume there is a perfect scheduling. As a result, the physical layer details such as path loss and interference are totally transparent to our problem formulation, *i.e.* we do not consider interference models. Instead, we build our model on top of the network with certain known per-node throughput capacity. Apparently, the multi-user wireless sensor network belongs to the hybrid network where two types of nodes exist. However, a basic distinction of the multi-user WSN is that there is no direct links between the users. Instead, the users here are logically individual entities. Therefore, the traffic pattern in this network is an aggregated effect of multiple many-to-one traffics. Throughout the thesis, we assume a static network although mobile users are more realistic for the application scenarios motivated this study. However, our problem formulation and solution have inbuilt mechanisms to handle limited query dynamics, *e.g.* new query, modified query or a query is canceled. This property provides a naive way to incorporate user mobility: query when the user is in static state, stop the query and move, then start a new query. Full support of user's query when it is moving requires more efforts and is left for future work. A dense network is assumed in Chapter 3 and this assumption is removed in Chapter 4 and 5 where only connectivity is required.

2.3 The Knapsack Problems

Knapsack Problem (KP) has been studied for a long time. Together with several variations of its original form, they are successful in modeling problems from different scientific and engineering fields. However, all of them are NP complete problems. Comprehensive discussions on the general KP family could be found in [MT90, KPP04].

MMKP is one of the most difficult variations of KP which could be seen as adding the multiple choice constraint to the multidimensional knapsack problem (MDKP). This problem has received more attention only recently. MMKP could be

described as follows. A number of items, each has a certain value, are to be selected from a set of items grouped into several classes then put into several knapsacks with each associated a certain capacity. Each item has a unique weight when put into a certain knapsack. The goal is to select exactly one item from each class such that the sum value is maximized and the sum weight in each knapsack is no more than its capacity. Particularly, an MMKP with a single constraint degrades to a Multiple Choice Knapsack Problem (MCKP). The distributed algorithm we will propose in Chapter 4 and 5 basically relies on solving the MCKP locally.

In general, MMKP could be seen as a mixed integer programming (MIP) problem which could be solved by applying the branch and bound strategy. Many algorithms have been proposed to solve the MMKP and they could be divided into two classes: exact algorithms and approximate algorithms.

Exact algorithms are designed to find the optimal solution but with a higher cost of computation effort. The algorithms proposed in [Kha98] and [Sbi07] belong to this category. Due to its computation complexity, exact algorithms are usually unable to solve problems with large number of variables. Under this case, its linear programming relaxation could be used to obtain an upper bound of the optimal solution. This method is useful in evaluating the approximate algorithms as well as in developing faster algorithms. On the other hand, approximate algorithms only find a sub-optimal solution but they are usually much faster than exact algorithms thus suitable for online applications which demand real time decisions. Thus, most literature deal with approximate algorithms, representative works are [MJS97, Kha98, KLMA02, AMSK01, HMS04, HMS06, ARK+06] and [CH08].

Besides the algorithms specifically designed for MMKP, there are several off-the-shelf MIP solvers available, *e.g.* the GLPK package[glp]. These solvers integrate many complex optimization techniques used in general integer programming.

2.3.1 Exact Algorithms for MMKP

Only two exact algorithms have been proposed for the MMKP in the literature and both of them are based on the branch-and-bound principle.

A straightforward extension of the branch-and-bound method for KP to MMKP is the BBLP algorithm [Kha98, KLMA02]. BBLP starts from a state that all variables are undecided. At each round, all partial feasible solutions (some variables have been decided while others are undecided) are examined and the best one with the maximum upper bound is selected. The upper bound is obtained by solving the LP relaxation of a sub-problem containing only the undecided variables. BBLP then tries to assign all possible values to one of the undecided variables. This may generate several feasible partial solutions while the unfeasible ones are dropped immediately. If at a certain round, the best partial solution selected has all variables fixed, an optimal solution is obtained.

Another exact algorithm is the EMKP algorithm [Sbi07]. EMKP initially sorts the items in each class in descending order of their profits. A tie between items is resolved by comparing the relative aggregated resource consumption of the items which is obtained by summing up the weight-constraint ratios across all dimensions.

Then it starts the branch-and-bound procedure from the first item in the first class. Based on the current node, two nodes are further developed if they exist: a son node corresponding to selecting the first item of the next class and a brother node corresponding to selecting the next item in the same class. The bounding procedure employs an upper bound obtained with an auxiliary MCKP problem and a supplementary KP. The auxiliary MCKP is formed with resources and constraints in the original problem aggregated across multiple dimensions, and the supplementary KP is formed with all items not selected and residual aggregated capacity. In order to further trim the branches of the search tree, a feasible solution obtained by the MRLS heuristic [HMS06] is used as a lower bound. A node with a solution upper bound below the lower bound is dropped.

2.3.2 Heuristic Algorithms for MMKP

2.3.2.1 Moser's Heuristic

The first heuristic algorithm for the MMKP has been proposed by [Mos96] and [MJS97] based on the Lagrange Multiplier Method.

The algorithm starts from a preprocessing step that normalizes the weights by the constraints. Then an initial solution is constructed with the most valuable item in each class. The algorithm then calculates the sum weight on all dimensions for the most valuable item in each class. If any constraint is violated in the initial solution, the algorithm identifies the most violated constraint dimension, *i.e.* a dimension that realizes the maximum sum weight. Based on the most violated dimension, the increase of the Lagrange Multiplier when the most valuable item is substituted by another item in the same class is computed. After the computation, the item that realizes the minimum increment is selected to substitute the most valuable item in the corresponding class. The Lagrange Multiplier and the sum weight is updated according to the substitution. The substitution procedure continues until no constraint is violated or no item can be used for the substitution. For the former case, the current solution is feasible and an improvement phase is followed. For the latter case, no feasible solution is found and the algorithm fails.

The improvement phase works as follows: for every class, the possible increment by exchanging the current selected item with a not-yet-selected item is computed. The item which realizes the largest increment is selected to substitute the current selected item. This improvement procedure repeats until no item is available for the substitution and the final solution is obtained.

Moser's algorithm has the worst-case run time of $O(ln^2)$, where $n = \sum_{i=1}^{m} n_i$ is the *total* number of items, m is the number of classes and l is the number of dimensions.

2.3.2.2 The HEU, M-HEU, I-HEU and MVRC Algorithms

The heuristic HEU has been proposed by [Kha98] and then enhanced by [AMSK01] and [KLMA02]. They are used to solve the QoS problem found in the adaptive multimedia systems. We describe the basic ideas of the enhanced version, namely,

the M-HEU [AMSK01]. M-HEU algorithm is based on the idea of feasibility factor, value-resource ratio, aggregate resource consumption, *etc.* proposed by [Toy75].

The basic idea of the M-HEU algorithm is to start with a minimum feasible solution, then further explore better solutions by exchanging repeatedly current selected items with other items. The objective of the exchange is to reduce the resource consumption or, in case that this objective is unachievable, to maximize the profit gain. The key problem is how an item is selected for the exchange. This is achieved by applying the aggregate resource notions used by [Toy75].

As the first step of the M-HEU, the minimum feasible solution is constructed by selecting the items with the least profit in each class. The resource usage vector is computed for this initial solution. M-HEU then checks the feasibility of this solution and if all dimensions of resource constraints are satisfied, the algorithm proceeds with the iterative improvement phase. Otherwise, it finds an exchange that realizes the maximum *saving in aggregate resource*. This step is repeated until no exchange is possible. Whenever a feasible solution is found, the M-HEU then tries to improve it by feasible exchanges. The objective of this phase is to first find an exchange that is feasible and realizes the maximum saving in aggregate resource. In case this objective is unachievable, it finds an exchange that is feasible and realizes the maximum utility gain of per-unit-of-extra-resource.

A hybrid improvement procedure with both upgrade and downgrade operations is applied to the results obtained by the previous phase. It begins with an upgrade that finds an exchange that maximizes the utility gain per unit of total-per-unit-resource. The obtained solution is further exchanged by a downgrade operation that minimizes the utility gain per unit of total-per-unit-resource and increases the total profit. If the exchange is possible and the resulted solution is feasible, then M-HEU accepts it and enters the next iteration of improvement. If the exchange is possible but the resulted solution is infeasible, M-HEU keeps to downgrade it. Finally, if the exchange is impossible, the M-HEU exit with the current solution.

M-HEU has time complexity of $O(m^2(n-1)^2l)$ with m the number of classes, n the number of items in each class and l the number of dimensions. Experiments show that M-HEU finds solution within 96% of the optimal in average.

Based on the M-HEU, an incremental version of the algorithm, namely the I-HEU algorithm, is also proposed by [AMSK01], which aims to be used for real-time applications with large number of variables. The merit of I-HEU is that the results from previous computations are used as basis of the current computation and only new classes are added into the computation. The system needs only to adapt these new users from previous state instead of recompute for all users.

Besides, the MVRC algorithm is proposed by [CA06], which is based on the HEU and with only minor modifications.

2.3.2.3 Parallel HEU and Multiprocessor M-HEU

A parallel MMKP algorithm proposed by [SIH05] is also based on the idea used in the M-HEU. The authors adapted the M-HEU to the Concurrent Read Concurrent Write Parallel Random Access Machine (CRCW PRAM). The basic modification

is to assign one class of items for each processor. The calculations on necessary parameters such as the saving in aggregate resource, *etc.* could be done in parallel. The item is fixed according to the same criteria as in M-HEU but only with help of parallel sort algorithms.

A multiprocessor algorithm based on M-HEU is proposed by [SARN08]. The algorithm has time complexity of $O\left(\frac{T}{p} + s(p)\right)$, where T is the time required by the algorithm using single processor, p is the number of processors and $s(p)$ is the synchronization overhead.

2.3.2.4 The CP and CCP Algorithms

How to find a good initial feasible solution seems to be important for the heuristic algorithms. [HMS04] proposed a Constructive Procedure (CP) and a Complementary Constructive Procedure (CCP) for this purpose.

In the CP algorithm, the *pseudo-utility* of each item is first computed. Then it selects the item with the highest pseudo-utility in each class to be the initial solution. If the solution is feasible, CP terminates with this solution. Otherwise, CP tries to drop an item and add another repeatedly to get a feasible solution. The most violated constraint is then decided and the item with the highest weight at the corresponding dimension is dropped. At the add stage, another item in the same class is selected trying to make the solution feasible, if there exists no such an item, the item with the lowest weight at the most violated dimension is selected. Under the latter case, another round of drop and add is needed as the current solution is infeasible.

The CCP aims to improve the solution obtained form CP with simple operations. It tries every class to substitute the current selected item with a not-yet-selected one that generates a feasible solution with larger profit. If this swap operation turns out to give more total profit, the CCP accepts it and enters the next iteration. The number of iterations is controlled by a predefined condition.

The CP has run time complexity of $O(\max\{n_{\max} l, m\})$, while the CCP has run time complexity of $O(n_{\max} \max\{l, m\})$, where $n_{\max}$ denotes the maximum number of item among all classes. Therefore, they could be used for obtaining an initial solution.

2.3.2.5 The Der_Algo, RLS, MRLS and Other Variations

Several heuristic algorithms have been proposed based on the CP and CCP, for example, the Der_Algo proposed by [HMS04] and the RLS and MRLS proposed by [HMS06].

In Der_Algo, the CP is first used to obtain an initial feasible solution. Then at each iteration, a better solution is obtained by CCP. Based on this solution, a penalize technique is applied to bring diversity into the searching process. The main idea is to randomly decrease the profit of some items and then solve the new problem with CCP. If a feasible solution is obtained, the items are restored with their original profit values and this transformation keeps the feasibility of the solution since it does

not change the weights of the items. This penalize-normalize procedure is repeated for several times and the best result is kept. The worst case time complexity of Der_Algo is $O(n_{\max}^2 \max\{l, m\})$.

The RLS and MRLS are two similar algorithms proposed by [HMS06]. Both consist of an initial constructive procedure to get a feasible solution to the problem by using a constructive procedure CP and CCP [HMS04] and a degradation procedure (which exchange the items within a certain class) trying to escape from local optima. The RLS then applies a deblock procedure to provide diversity to the searching process. While for the MRLS, the deblock procedure is replaced by a memory list which remembers the visited solution. With this list, the searching process is able to avoid revisit the solutions which give the same objective value. The RLS and MRLS has the same worst cast complexity of $O(mn_{\max}l^2)$.

Several heuristic algorithms are proposed by [Hir08]. However, there is no fundamental difference between the first two algorithms proposed and the CP and CCP proposed by [HMS04]. The last heuristic algorithm proposed by [Hir08] is a metaheuristic which applies the tabu search. The resulted FLTS algorithm is very similar to the MRLS proposed by [HMS06].

2.3.2.6 The HMMKP Algorithm

The HMMKP algorithm is proposed by [PHD05] where the MMKP is first relaxed into a MDKP, then Lagrange Multipliers are calculated for the MDKP. The Lagrange Multipliers are further used to determine the *pseudo-utility* of the items. The main search process is then guided by pseudo-utility and the *resource value coefficient*.

The HMMKP starts with calculating the *pseudo-utility* and the *resource value coefficient*. Based on these two sorted lists, the items are selected in a greedy way. HMMKP checks all solutions obtained till now, if there are feasible solutions, the one that realizes the best objective function is remembered. Otherwise, a greedy solution construction process similar to the one above is used, but this time with the resource value coefficient information.

Whenever a feasible solution is obtained in the steps described above, the solution is further improved by iteratively applying a local feasible exchange twice: once with the pseudo-utility and once with the profit directly.

The complexity of the HMMKP is $O((mn)^{3.5}t)$, assuming all classes contain the same number n of items and t is a precision-time trade-off parameter introduce by the interior-point algorithm employed when solving the LP relaxation.

2.3.2.7 The C-HEU Algorithm

C-HEU is proposed by [ARK+06]. The algorithm is based on constructing convex hull on the items which are put in a resource-value coordination. This convex hull approach is first proposed by [LLRS99] in solving resource allocation in QoS management, which could be transformed into a MMKP. However, in Lee's problem settings, the profit is monotone to the resource consumption which makes their algorithm not fit the general MMKP case. So [ARK+06] modified the Lee's algorithm making it fit

for general MMKP problems where the value could be non-monotone to the resource consumption. The C-HEU has a time complexity of $O(nlm + nl \log l + nl \log n)$.

The basic idea of both the C-HEU and Lee's algorithm is to aggregate the multidimensional resource into one dimension then construct a convex hull for items in each class by putting the items into a aggregate resource-profit coordinate. The first step employs a different way than the aggregate resource consumption defined for the HEU by [Kha98] and M-HEU by [AMSK01]. In C-HEU, the multidimensional resource consumption of an item is first penalized with the resource consumption of the current solution as the penalty vector on initialization of the algorithm. The penalty vector is updated at each iteration according to the residual resource.

The second step is to construct convex hull based on the aggregate resource-profit coordinate. Many convex hull algorithms could be used for this step. C-HEU then sorts the line segments which define the convex hull by their gradients (used by [LLRS99]) or by the angles they form with the positive direction of the resource axis (by [ARK+06]). The advantage of using the angles is that it could be used for non-monotone resource consumption cases. Finally, the items on the hull are selected according to the order of the segments related with them.

Although C-HEU achieves very low time complexity, it is still insufficient to be applied to run-time critical situations such as the multi-processor scheduling problem studied by [YCNCC06]. Therefore, the authors proposed another MMKP algorithm based on similar ideas of aggregate resource. A Pareto filtering pre-process is applied by [YCNCC06] which keeps only the items that satisfies the Pareto condition. Based on experiences, many items could be removed by this step for the popular problem instances reported in [OR-]. Then the resource consumption is aggregated into one dimension with a slightly different penalty vector as used in C-HEU. Then all items in all classes are sorted by their profit-resource ratios in non-decreasing order. This gives priority to the items with higher profit but consumes less resources. Then a greedy selection procedure is applied for each item. The time complexity of the resulted algorithm is $O(l + 2mn + mn \log(mn))$.

2.3.2.8 The CGBA Algorithm

Recently, [CH08] investigated the feasibility to apply Column Generation in solving the MMKP. The resulted CGBA algorithm combines the column generation with a heuristic search.

Column Generation was first proposed by [GG61, GG63] for solving the cutting stock problem which is a linear programming problem with several rows and a large number of columns. The idea of column generation comes from the simplex method for LP problems. In the simplex method, we look for a non-basic variable which improves the solution and put it in the basis which consists of calculation for all variables. This calculation can be prohibitive when the number of variables are large. Instead of considering all the variables during the pricing procedure, column generation works with only a subset of columns of variables. The problem composed of this subset of columns is referred to as the restricted master problem. Solving the LP of the restricted master problems produces a first solution. Then similar

to the simplex method, one looks for new variable that has negative reduced cost (for minimization problem) and this step is usually referred to as solving the subproblem that minimizes the reduced cost. If no such variable can be found, the LP of the master problem is optimized already, otherwise, the new variable enters to the restricted master problem and column generation continues. For integer programming problems, the column generation should be combined with a branch-and-bound procedure.

The major steps of the CGBA could be described as follows:

- Construct an initial solution and restrict the master problem to variables appear in the initial solution. The initial solution is obtained with the CP or CCP procedure proposed by [HMS04].

- Column generation is applied to the restricted master problem related with each node in the search tree.

- A greedy rounding procedure is performed on one part of the variables resulted from the column generation while a specialized solution procedure is performed on the other part of the variables, in order to obtain an integer solution.

- Generate new branches and remove inspected and unpromising nodes. The branching strategy could be one of the following as suggested in the paper: (i) to branch on the most fractional variable or, (ii) simply split variables in one class into two parts and let all variables in one part be zero and let one in the other part be one. The split item could be the first fractional variable or, (iii) branch only on the newly added variables by the column generation procedure.

- Exit when no more node has to be inspected or stop condition is satisfied.

Experiments on the OR Library instances show that CGBA is very effective in solving large instances.

2.3.3 Summary

In this thesis, we consider the problem of allocating query ranges for the users, which is formulated into the MMKP in Section 4.1.2. As the users are geographically distributed in the network of sensors, we would like to find a distributed algorithm which enables the users and the sensor nodes to solve the problem in a collaborated and dynamical way. However, most of the existing algorithms are supposed to run in a centralized way with only the Parallel HEU and Multiprocessor M-HEU for as exceptions. The Parallel HEU is designed for the CRCW PRAM which has no real-world implementation, while the multiprocessor M-HEU is not efficient to handle user join and leave. The I-HEU is designed to recalculate the MMKP solution after user joins or leaves the multimedia system, however, it is a centralized algorithm.

2.4 Fairness

Several fairness notions, namely the max-min fairness, lexicographical max-min fairness, proportional fairness and the $(p, \alpha)-$proportional fairness, have been used widely in the network flow related studies. We give each of them a very brief introduction in this section.

2.4.1 Max-min and Lexicographical Max-min Fairness

Max-min fairness is defined as a special feasible state when sharing certain amount of resources among multiple parties, such that any party can not increase its share without decreasing that of the others which is already of less amount. Max-min fairness gives absolute priority to the parties consuming less resources and it could be achieved by simple water filling algorithm [RD]. However, implementing this algorithm in packet switching networks, which is distributed in nature such that each node has to schedule its packet transmission rate, requires global state information and timing.

The above discussions are based on the assumption that the resources shared are of continuous values. This is generally true for most problems such as the transmission rate allocation problem. However, under situations such that the resources to be shared are of fixed sizes, we have to use the number of potions as a notation of the resources obtained by each party, which is often an integer value. For example, in the layered transmission scheme considered by [LMC04], the bandwidth is allocated in discrete fashion. Under this case, the max-min fairness may not exist (as the optimal may require fractional resource allocations to some party). In contrast, lexicographical max-min fairness exists under this case. The lexicographical max-min fairness is defined as the following (feasible) state: the sorted allocation of resources for each party (resources obtained by each party are sorted in increasing order), is the largest among all possible allocations, where the "largest" is defined by the lexicographical order. Because of the lexicographical order plays a fundamental role in defining the optimal state, the lexicographical max-min fairness is also referred to as, classically, the leximin maximality, *i.e.* a feasible state of resource allocation is lexicographical max-min fair if and only if it is leximin maximal.

Although very similar to the (continuous) max-min fairness, the lexicographical case is basically a combinatorial optimization problem which is NP-hard.

2.4.2 Proportional Fairness

Proportional fairness is proposed by [Kel97] as a compensate fairness definition that give less aggressive priority to the parties that receive less resource as max-min fairness does.

If an allocation of resources to each party i is denoted as x_i, then a special allocation x_i^* is said to be proportionally fair if x_i^* is feasible and the aggregated

proportional change is zero or negative, *i.e.*:

$$\sum_i \frac{x_i - x_i^*}{x_i^*} \leq 0. \tag{2.2}$$

Proportional fair allocation could be obtained by solving a constrained optimization problem whose objective is to maximize the sum of the logarithm of each allocation. Under the bandwidth allocation problem regimes, [KMT98] have shown that simple rate control algorithms with additive increase, multiplicative decrease converge to proportional fairness.

2.4.3 $(p, \alpha)-$proportional Fairness

$(p, \alpha)-$proportional fairness proposed by [MW00] is a generalization of the proportional fairness such that the proportional change is augmented by positive parameters p and α as follows:

$$\sum_i p_i \frac{x_i - x_i^*}{x_i^{*\alpha}} \leq 0. \tag{2.3}$$

Obviously, $(p, \alpha)-$proportional fairness reduces to proportional fairness when $p_1 = p_2 = \cdots = 1$ and $\alpha = 1$.

Similarly to proportional fairness, $(p, \alpha)-$proportional fairness could be obtained by solving an optimization problem whose objective is to maximize the sum of a special function of each allocation defined as follows:

$$f_\alpha(x) = \begin{cases} \log x & \text{if } \alpha = 1, \\ \frac{1}{1-\alpha} x^{1-\alpha} & \text{otherwise.} \end{cases} \tag{2.4}$$

The above objective function corresponds to proportional fairness when $\alpha = 1$ and corresponds to max-min fairness otherwise. A formal proof of the latter case is provided by [MW00].

Chapter 3

Fair Query Allocation In WSN Under A Continuous Query Model

"Anticipate the difficult by managing the easy."
– Lao Zi

This chapter is organized as follows. In Section 3.1, we first describe in more detail an example application scenario for the counter-emergency facilitating system discussed in Section 1.3. The problem is formally defined and the notations are introduced in Section 3.2. Theoretical results when only two users share the network are derived in Section 3.3. Then Section 3.4 describes a distributed algorithm which will be analyzed extensively by simulations in Section 3.5. We review the related works in Section 3.6 before drawing a conclusion in Section 3.7.

3.1 Introduction

In certain critical areas such as those regulated by the Seveso directive, the density of sensors is expected to be high enough to build a self-organized multi-hop wireless sensor network (WSN) [CES04]. During crisis, such networks could substantially help the intervention teams by notifying selected events [LMFJ+04]. A typical application of this network could be a monitoring system with some device-equipped firemen gathering data from a burning area in order to determine a safe perimeter, while others operating on the hearth are under real-time alert about risks of nearby explosions. The firemen send requests to and collect data from the sensors within a specific area in a multi-hop fashion. Thus they could be seen as *mobile users*.

This monitoring system should be trustworthy so that all events within a monitored area must be reported to the querying user, and should be adaptable since the number of users changes over time. Besides, it is desired for each user to monitor the largest possible area to ensure individual security. However, since the sensor

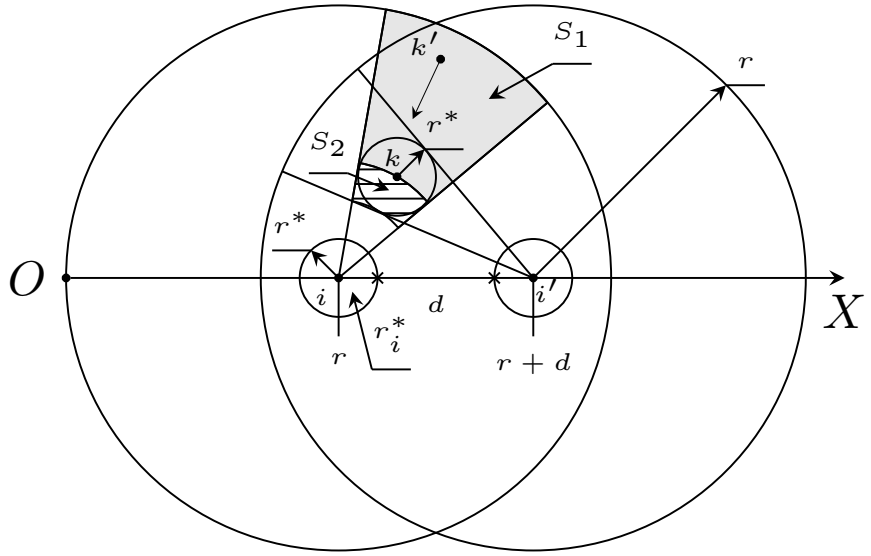


Figure 3.1: Traffic load model.

nodes have limited data transmission capability, multiple data requests could congest sensors in certain area, depending on the position of the users, the size of the requested area, *etc.* Furthermore, if the sensors are already saturated by current queries, newly joined users will be kept in a starved state and unable to retrieve information from those sensors until some of the existing users leave. On the contrary, if each fireman accepts to decrease the size of its query area, it may enable newcomers to use the network as well. Thus, fairness among users when sharing the network is preferred. As a result, the users tend to increase their queries as long as some fairness rules are kept. We assume that users do not coordinate through direct communication links because this will form a complex network structure which will be studied in future work.

The contributions of this chapter are threefold. First, we introduce a simple model for the capacity sharing problem. We give a formal definition of *supported users* and *feasible network configurations*, then we describe immediate results obtained from previous works. Second, we present a distributed algorithm allowing users to determine their query radii with respect to nearby sensors and users. The algorithm is designed to be adaptive to the number of users. Finally, some related issues are illustrated and the validity of the algorithms proposed is justified with simulations.

3.2 Continuous Query Model

In this section, we introduce our system model with the definitions and notations. Some of them with geometrical meanings are presented in Figure 3.1.

Wireless sensor network: A WSN consists of a set V of l sensor nodes deployed in a fixed area according to a Poisson distribution with density λ_0 and a set S of m user nodes. Each node is assumed to have an omnidirectional wireless

transceiver that is able to transmit or receive W bits of data per second to or from a fixed maximum distance $r^* > 0$. Nodes are unable to receive simultaneous messages. It is natural to assume the wireless transmission range is unable to cover the whole network, thus for the communications between sensors and users, a multi-hop routing protocol has to be employed. Furthermore, we assume the routing protocol ensures shortest path and load balancing for packets. Finally, we do not consider malicious behaviors in this study.

Query model: A *query* emitted by a user $i \in S$ pertains to a sub-area assumed to be a disk of radius r_i centered at the user and will be referred to as $Q_{r_i b}$, where b denotes that each sensor within the queried area will generate b bits of data per second in response to this query. We assume there is no aggregation or compression when the data is collected and restrict each user always has one running query, thus there are exactly m running queries in the network.

We say that a sensor is covered by a query if it generates data for the query. The set of sensors covered by query $Q_{r_i b}$ is noted as $V_{r_i b} \subseteq V$. A sensor may be covered by more than one queries and the set of queries sensor k has to process is noted as $Q(k)$. The circle with radius r^* around the user i is noted as r_i^*, as shown in Figure 3.1, and will also be referred to as the *critical region* of i, since it is usually the busiest region within the query. User i is a (r_i, b)-*supported user* by the network if for a query $Q_{r_i b}$, all data emitted by sensors in $V_{r_i b}$ arrive to i.

Obviously, the positions of sensors and users are necessary for this kind of area based query model. However, GPS capability is not mandatory since several localization protocols are available, *e.g.* [AS01], as a result of intensive studies on such mechanisms.

Traffic load model: The traffic load model we will employ has been proposed in [GK04] and extended in [LH05]. The authors of [GK04] observed that for two sensors k, k' and a user i such that k' is outside the one-hop region r_i^* of the user, the data generated by k' destined to i is forwarded by k if the distance from k to the line $\overline{k'i}$ is less than r_k^* and the projection of k on $\overline{k'i}$ lies between k' and i. Figure 3.1 is largely depicted from [LH05], where i and i' are users and k and k' are two representative sensor nodes. Two cross marks denote the most loaded points. When sensor k is outside r_i^*, it is responsible for forwarding the data generated by all sensors within the sector S_1 behind it (gray shadowed sector of ring in Figure 3.1), whose sides are the tangents from the user to the circle centered at sensor k with radius r^*. Since we assumed that the underlying routing protocol would provide us with shortest path and load balancing properties, a single sensor k is unlikely to handle all the traffic from S_1. Rather, the traffic load from both S_1 and S_2 regions will be shared by all sensors within S_2 region. Let d_{ik} denote the distance between user i and node k, the region S_2 is a sector of ring whose inner and outer radii are $d_{ik} - r^*$ and d_{ik}, respectively, and whose sides are the same tangents as that of S_1, as shown by the area shadowed with horizontal line pattern in Figure 3.1. Thus, the average traffic load of k is proportional to $(S_1 + S_2)/S_2$. On the other hand, sensors within the r_i^* will share the traffic from all over the query region. Let $\delta(d_{ki})$ the traffic load of a sensor k at distance d_{ki} to a user i, we have:

$$\delta(d_{ki}) = \begin{cases} \frac{br^2}{(r^*)^2} & k \in r_i^* \\ b + \frac{2br^2}{\pi(r^*)^2} \arcsin\left(\frac{r^*}{d_{ki}}\right) - \frac{2bd_{ki}^2}{\pi(r^*)^2} \arcsin\left(\frac{r^*}{d_{ki}}\right) & k \in Q_{r_i b} \setminus r_i^* \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Max-min fair configuration: A *configuration* is defined by a set of radius-user pairs: $C = \{r_i | i \in S\}$ and we say that C is a *feasible configuration* if $\forall r_i \in C$, user i is (r_i, b)-supported. Although there exists many feasible configurations for a given network topology, we are interested in the one with *max-min fair* properties such that smallest radii are maximized, then the second smaller radii are maximized and so on. A basic rule is that maximizing a query should not shrink other already smaller queries. Obviously, the capacity constraint should be kept at the same time. Note that in the context we are considering, the traffic b is fixed and each user is interested in maximizing its query radius to maximize individual security.

3.3 Analysis of Two-user Case

Now we consider only two users i and i' are in the network. This simple case provides a good introduction to our capacity sharing problem. Intuitively, the max-min fair radius for each user should take the same value. We distinguish three cases based on the relative position of the two users: distant users, not-so-distant users and nearby users. For each case, we shall analyze the traffic load and then derive the max-min fair query radius for each user.

Distant users: In a WSN with Poisson node distribution and all-to-one communication paradigm, the bottleneck is the user itself [MDMLN03]. We say that a query $Q_{r_i b}$ is *globally maximized* when the bandwidth of all nodes within r_i^* are saturated by traffic exclusively dedicated to i. If we ignore the bandwidth consumed by protocol overheads, query $Q_{r_i b}$ is globally maximized when $\pi r_i^2 \lambda_0 b = W$. When the queried data rate b and the density of sensors λ_0 are fixed, we naturally obtain:

$$r_{max} = \sqrt{\frac{W}{\pi b \lambda_0}} \tag{3.2}$$

The users i and i' are considered as *distant* when the critical region of one query does not overlap with the other query. This happens when $d_{ii'} > r_{max} + r^*$. In this case, the maximum amount of data a user can receive is bounded by its bandwidth and the two queries could be both globally maximized with the same radius r_{max}. Obviously, the configuration $C = \{r_i, r_{i'}\}$ with $r_i = r_{i'} = r_{max}$ is max-min fair.

Not-so-distant users: When $2r^* < d_{ii'} \leq r_{max} + r^*$, the critical region of one query may be covered by the other query but r_i^* and $r_{i'}^*$ do not overlap. Under this case, if either of the two queries is globally maximized, the other query has to shrink accordingly. On the other hand, the two users may coordinate to achieve an equilibrium state such that they have the same query radius. This state is exactly the max-min fairness we want to achieve. Under this configuration, both users i and

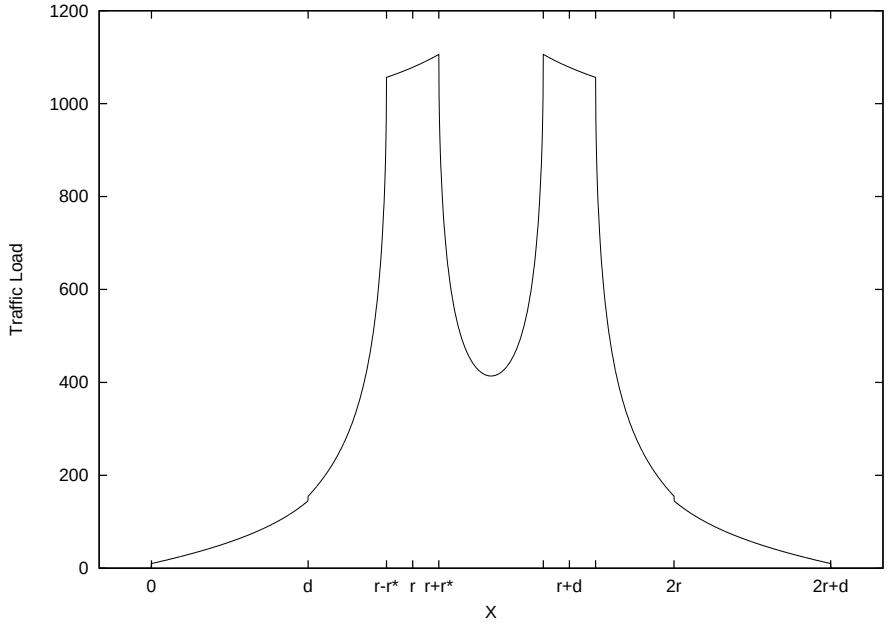


Figure 3.2: Traffic load for sensors along X axis.

i' experience similar queries and sensor throughput. The total traffic handled by a sensor k is $\delta(d_{ki})+\delta(d_{ki'})$. Due to the bottleneck effect of the user, the total traffic of the sensors within the critical region of either user should not exceed the bandwidth of the user. Note that the traffic generated or forwarded by sensors in r_i^*, whether it is for i or i', consumes the bandwidth of i because all sensors in r_i^* share the wireless medium with i. Thus, in order to avoid congestion within the critical region, the sum traffic in this region should always be kept under W. The calculation of the sum traffic within the critical region requires integration of the traffic load function (3.1) within the region, which gives no explicit solution of the max-min fair query radius. However, the characteristics of the traffic function produces two most loaded points lying at the intersection of the line joining i and i' and the two circles delimiting r_i^* and $r_{i'}^*$, as shown in Figure 3.1 with two cross marks. If we setup a coordinate system with X axis and origin O as shown in Figure 3.1, the traffic load of sensors on X could be plotted in Figure 3.2, where two users lie at $x = r$ and $x = r + d$ and we see two maximum at $x = r + r^*$ and $x = r + d - r^*$. Note that the traffic load for other sensors not on X is always lighter than those on X. As a result, we could limit the traffic of these two busiest sensors under the *shared* bandwidth of sensors within the critical region to make sure that no congestion occurs.

The shared bandwidth of sensors within r_i^* is $\frac{W}{\pi(r^*)^2\lambda_0}$. Together with (3.1), the total traffic $T(k)$ handled by a sensor k, when maximized with max-min fairness, should be:

$$T(k) = \delta(r^*) + \delta(d_{ii'} - r^*) = \frac{W}{\pi(r^*)^2\lambda_0} \tag{3.3}$$

Solving this equation for r, we obtain a common radius for both users, noted as

r_c:

$$r_c = \sqrt{\frac{\left(\frac{W}{\pi(r^*)^2\lambda_0} - b + t_1\right)}{t_2}}, \tag{3.4}$$

where:

$$t_1 = \frac{2b(d-r^*)^2}{\pi(r^*)^2}\arcsin\left(\frac{r^*}{d-r^*}\right) \tag{3.5}$$

$$t_2 = \frac{b}{(r^*)^2} + \frac{2b}{\pi(r^*)^2}\arcsin\left(\frac{r^*}{d-r^*}\right) \tag{3.6}$$

As a result, two users can be $(r_c, b)-$supported and each user is able to determine r_c by communicating with a sensor within its query. Thus the max-min configuration for the not-so-distant case is $C = \{r_i, r_{i'}\}$ where $r_i = r_{i'} = r_c$.

Nearby users: In this final case, two users are even nearer, *i.e.* $d_{ii'} \leq 2r^*$, meaning that the critical regions of i and i' overlap. The reasoning is the same as in previous cases, except that all sensors in the area $r_i^* \cap r_{i'}^*$ are bottlenecks. Thus, each sensor within this area should handle a total traffic load as:

$$T(k) = \frac{2br^2}{(r^*)^2} = \frac{W}{\pi(r^*)^2\lambda_0}. \tag{3.7}$$

Solve the above equation for r which is actually the common radius r_c for the two users:

$$r_c = \sqrt{\frac{W}{2\pi\lambda_0 b}}. \tag{3.8}$$

The max-min fair configuration is $C = \{r_i, r_{i'}\}$.

We should note here that in our firemen application scenario, due to the various situations of the fire site, all of the three cases discussed above could exist. For example, firemen could operate separately in the forest on fire, or form small groups searching for survivors inside a building.

As a conclusion of this section, we present in Figure 3.3 the max-min radius for both users with respect to the distance between them. The transmission range of nodes is set to $r^* = 10$m. We see that the query radius does not increase linearly with the distance between two users, instead, when the distance is short, increasing it by even a small value will substantially enlarge the common radius.

3.4 Distributed Algorithms

In this section, we develop algorithms and a protocol to solve the max-min fairness problem under generic cases when more than two users are in the network.

3.4.1 Brute force algorithm

It is known that there exists a simply algorithm based on brute force to achieve max-min fairness, which is as known as the water filling algorithm [Bou06]. This

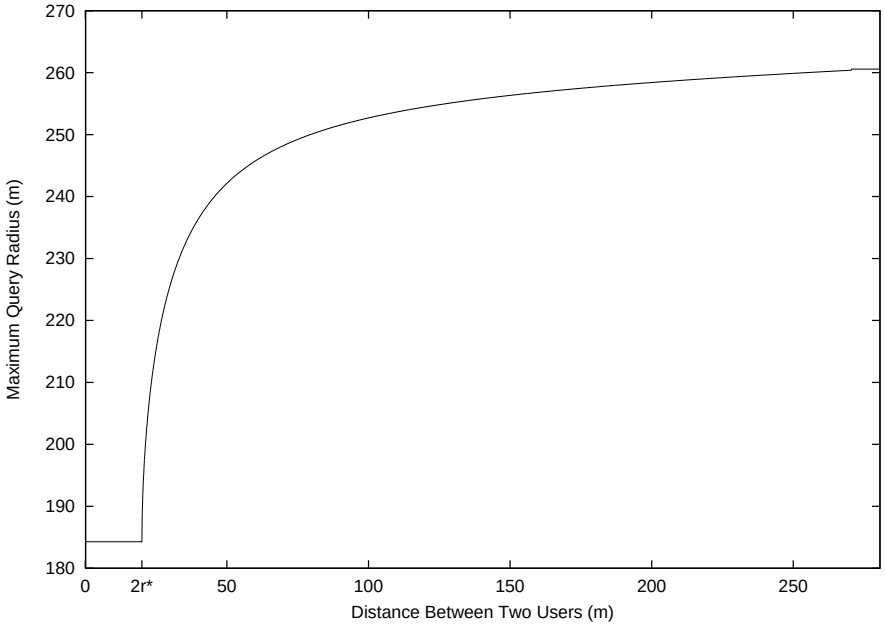


Figure 3.3: Common maximum query radius.

algorithm, we note it as OPT, works as follows: all users are simultaneously switched on, then increase their query radii with the same step length in a perfect synchronous manner until a sensor is saturated (this sensor will be referred to as a bottleneck sensor). When a sensor is saturated, the users querying this sensor stop increasing their queries, while others keep increasing. The algorithm ends when no query can increase. Although this algorithm is not realistic, it is helpful to understand the behavior and evaluate the performance of other algorithms.

3.4.2 Inspiration from two-user case

From the discussions on the two-user case, we know that if both users want to maximize their queries while keeping max-min fairness in mind, the resulting radii must converge to a common value. Based on this observation, we can imagine that when more than two users are in the network, each pair of users can agree on a max-min fair radius between them. As a result, if there are m users in the network, each user will have a maximum of $m - 1$ common radii with other users. If each user set its radius to the minimum one of its common radii with others, no sensor will be overloaded. Moreover, this mechanism also ensures that the minimum query is maximized. However, we found this is not true for non-minimum radii.

This problem could be better explained with help of Figure 3.4, which illustrates a typical WSN with eight users with numbers as their labels. The small circles around the numbers denote critical regions of each user while the larger ones denote the query boundaries and the crosses denote bottleneck sensors. Other sensors are omitted in this figure. For user 2 and 3, the radii of their queries are bounded by a bottleneck sensor between them, so the users adopt the same query radius with

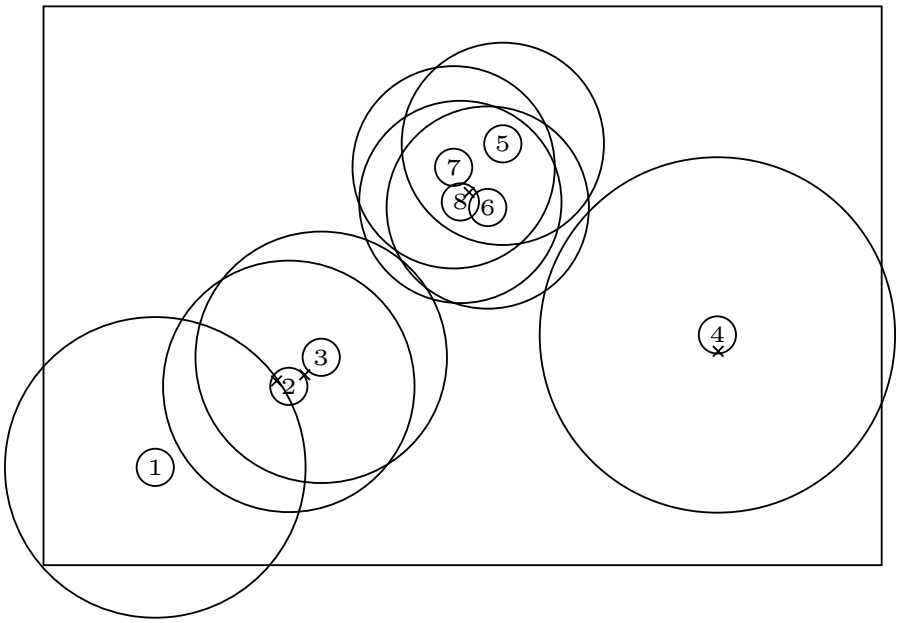


Figure 3.4: A WSN shared by eight users.

respect to the max-min fairness policy. Now let us consider user 1. The proximity of user 1 with 2 let it assume a bottleneck between them which lies at one of the intersections of the boarder of $Q(1)$ and r_2^* in Figure 3.4. Thus, user 1 determines a radius based on this bottleneck and *thinks* that 2 acts accordingly. But, user 2 *does not* because the bottleneck with user 3 is more constraining. Therefore, user 1 is unable to achieve its maximum query radius.

Let us associate a bottleneck node $\beta(i) \in V \cup S$ with each query $Q_{r_i b}$. When a user is $(r_{\max}, b)-$supported, the bottleneck $\beta(i)$ is i itself, otherwise, the bottleneck is a certain sensor. As shown in Section 3.3, when there are only two users i and i', the two bottlenecks experience the same traffic with the same bandwidth sharing. By notation abuse, we claim $\beta(i) = \beta(i')$. For example, in Figure 3.4, the users 5, 6, 7 and 8 have the same bottleneck sensor. The problem occurs when a user i adjusts its radius without being informed that the radius of its buddy i' does not depend on $\beta(i)$.

However, it is intuitively expected that the problem discussed above is infrequent so an algorithm based on their own bottlenecks may be sufficient for each user. This idea inspires an algorithm, which will be referred to as LOCAL and can be stated as follows. A saturated sensor k considers itself as a potential bottleneck for all users in $Q(k)$. Therefore, it notifies these users with the following information: all the radii of the queries in $Q(k)$, the positions of each user in $Q(k)$ as well as the position of itself. When a user receives such a notification, it computes a common radius with respect to all the users contained in the notification and adjusts its query radius to the minimum one.

An experiment was carried out over a WSN with different number of users for both LOCAL and OPT algorithms. The minimal, average and maximal query radii over all users are plotted in Figure 3.5. As expected, the minimum radius are

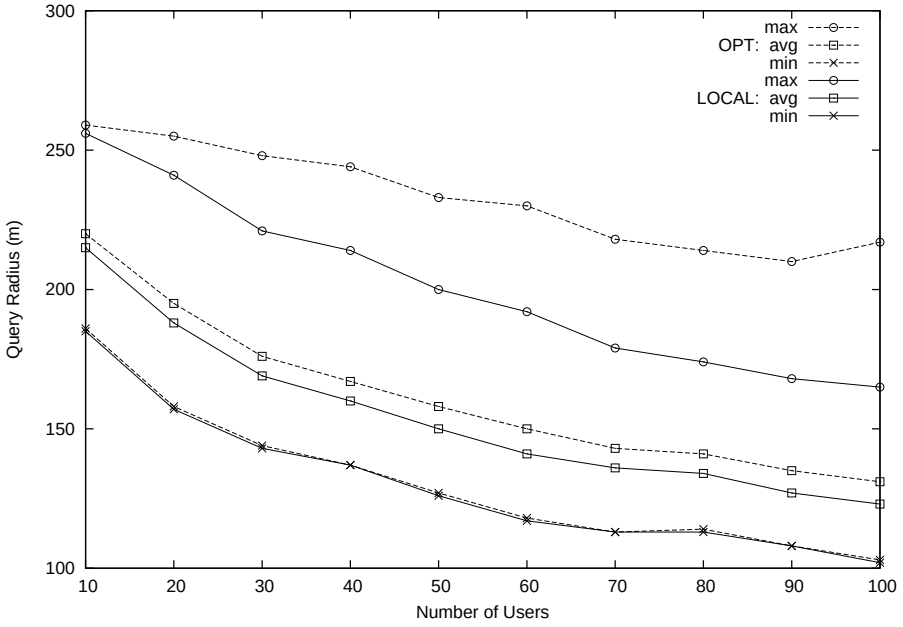


Figure 3.5: Query radii of OPT and LOCAL algorithms.

always the same for both algorithms, but average and maximum values of the LOCAL algorithm are below those of the OPT algorithm.

Another observation is illustrated in Figure 3.6. For three distinct contexts: 30, 60 and 90 users, we sort the users by their OPT radii in ascendant order, so that the users with smaller radius have a lower rank. Then for each user, we compute the ratio between the LOCAL radii and OPT radii. On one hand, LOCAL algorithm achieves max-min fair query for users with minimum radius but the performance gets worse for users with larger radius. Under the worst cases, the ratio could be less than 0.8, indicating that the area covered by this user would have been 1.56 times larger were OPT algorithm used instead. On the other hand, still shown in Figure 3.6, only 10% of queries are notably smaller ($\leq 90\%$) than their OPT counterparts, implying that searching for OPT values based on LOCAL algorithm could be efficient.

3.4.3 Distributed algorithm and protocol

Now we propose a distributed algorithm (DIS) and a protocol realizing it to achieve max-min fairness for all queries in the network by combining the two algorithms discussed in the previous section.

The idea of the algorithm is quite simple and widely known in congestion control algorithms [WDM01]. The pseudo code is shown in Algorithm 3, where several messages carrying control information are defined: `<query>`, i, r_i`>` message, sent to the sensors within circle (i, r_i) by user i to start a query with radius r_i; `<modify-query>`, i, r_{mod}`>` message, sent by i to the sensors within a circle whose radius equals to $\max\{r_i, r_{mod}\}$ to modify the query radius to r_{mod}; `<saturated>` message, sent by a

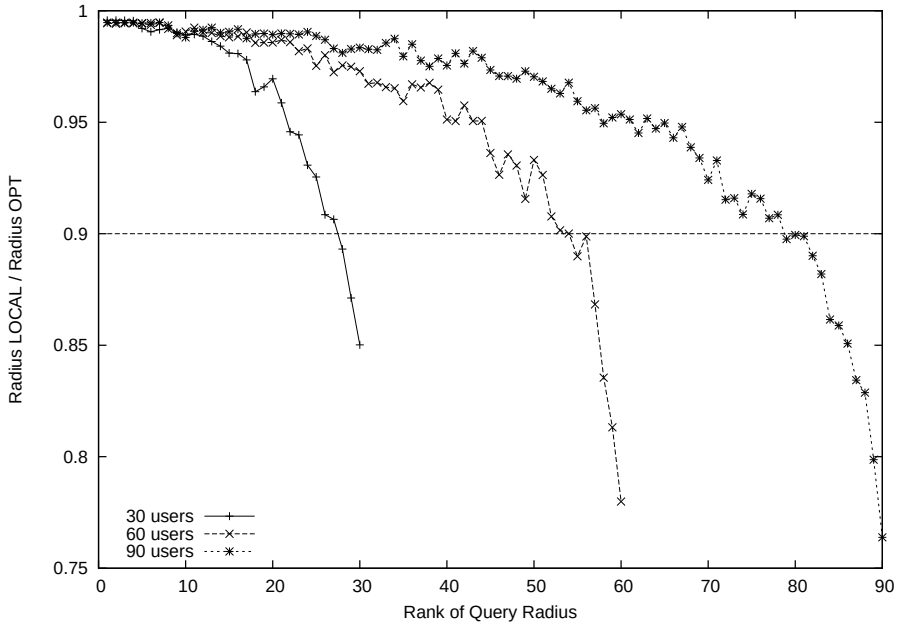


Figure 3.6: Ratio of LOCAL radii to OPT radii.

saturated sensor k to the users in $Q(k)$ when:

$$T(k) = \sum_{i \in \mathcal{Q}(k)} \delta(d_{ki}) > \frac{W}{\pi (r^*)^2 \lambda_0} \tag{3.9}$$

and it carries all the radii of the queries in $Q(k)$, the positions of each user in $Q(k)$ as well as the position of itself.

The algorithm consists of two phases. The first one, usually called *slow start*, is used until an approximation of the achievable radius is obtained. The radius starts at a unit value and increases according to a certain strategy until the user is alerted by a `<saturated>` message. The function `initRadius` manages the initial radius increasing. On receiving the first `<saturated>` message, the radius is set to a lower value according to some criteria, then the system enters the second phase. In the second phase, each user tries to increase its query radius periodically, in order to explore the OPT radius. This is handled by the `increaseRadius` function. Eventually, the user is alerted by a `<saturated>` message, then it decreases its radius again and resumes increasing. The new radius is obtained by `resetRadius`.

Various implementations of `initRadius`, `increaseRadius` and `resetRadius` functions could be used. We propose one as follows. The `initRadius` is based on exponential growth, *i.e.* the query radius is initialized to 1 and increased by doubling its previous value each time it is called. The idea is to detect as quickly as possible the bottleneck sensors. Then, the `resetRadius` employs the LOCAL algorithm which returns a radius close to the optimal one in most of the cases as previously discussed. Finally, the `increaseRadius` function makes a linear increasing of the radius after every predefined time interval by a predefined step length.

Algorithm 3: Distributed Algorithm (User Part)

```
Data: radius
send <query, i, 1 >
while no <saturated> message do
    radius =initRadius ()
    send <query, i, radius >
end
radius =resetRadius ()
do
    while no <saturated> message do
        radius =increaseRadius ()
        send <modify-query, i, radius >
    end
    radius =resetRadius ()
    send <modify-query, i, radius >
loop
```

By choosing these implementations, the users see their query radii oscillate between the values of the LOCAL algorithm and the OPT algorithm. In fact, from the observation of Figure 3.6, it is expected that a large part of users experience small variations because these two values are very close. Meanwhile a small fraction of the users, those with the largest radius, benefit from a larger area for most of the time.

3.5 Performance Evaluation

In this section, we shall present several simulation results to further illustrate the problems encountered when multiple users try to share the network capacity, and analyze the ability of our algorithm in solving these problems.

First of all, we describe briefly our simulation implementation. We implemented DIS algorithm in a simplified simulator, where a round based message passing mechanism with hop delay is provided. At each simulation round, events are scheduled to be processed at each user and sensor and this procedure may generate new events to be scheduled later. We did not implement the data traffic nor the lower MAC layer. Instead, the bandwidth utilization is computed by the traffic load function (3.1) and we assume a perfect scheduling mechanism is employed by the underlying MAC layer which achieves the link capacity. All simulations follow a common set of network settings depicted in Table 3.1.

Our first simulation studies the overall capacity utilization μ of the network under a OPT configuration, where μ is defined as:

$$\mu = \frac{\sum_{k \in V} \sum_{i \in S} \delta(d_{ki}, r_i)}{\frac{lW}{\pi (r^*)^2 \lambda_0}} \tag{3.10}$$

Table 3.1: Simulation parameters.

Network area	1000m × 1000m
Number of sensors (l)	3000
Number of users (m)	Variable
Sensor distribution	Poisson
User distribution	Poisson
Wireless Tx/Rx bandwidth (W)	12.8kbps
Wireless Tx/Rx radius (r^*)	50m
Query data rate (b)	20bps
Query radius increase interval	10 simulation rounds
Query radius increase step	10m
Total simulation rounds	400

Note that the traffic load function δ defined in (3.1) becomes a function of both d_{ki} and r_i since the users may have different query radii.

In order to eliminate random effects, we run the simulation with a hundred different topologies. Figure 3.7 shows the results. We see that the bandwidth utility grows with the user number but at a decreasing rate. On the other hand, the utility is quite low especially when less users are in the network. This is due to the query model we have employed. Future works will have to consider different query models in order to achieve a higher network utility.

The second simulation focuses on the distributed algorithm. We observe the query radius during a simulation run and show how DIS algorithm copes with users joining and leaving the network. So, 30 users are initially deployed and they start a query at the beginning of the simulation. Then at simulation round 200, three users join the network and start their queries. We record the radii of each user at each simulation round and compare it with the OPT query radii under the same network topology.

Figure 3.8(a) shows the OPT radius before and after three users are added in the network. Gray plates denote queries not affected by user addition or removal. Dotted and solid circles denote the affected optimal query radii before and after topology change, respectively. Most of the affected queries shrink in order to share the network capacity with the newcomers, for example query 1 and 8 are forced to shrink by query 31. However, some affected queries enlarge. This may happen when their neighboring queries are forced to shrink. Such behavior is expressed by user 26 since query 2 is further limited by a new query 32. Similar behaviour was observed when users leave the network in Figure 3.8(b), where most of the impacted users experience an enlarged query radius, while a few others have to shrink, *e.g.* user 21.

Now we analyze how the query radii evolve with time. In Figure 3.9(a), four pairs of users are selected for demonstration: user 1 with 31, 9 with 30, 2 with 32 and 26 with 32. In each pair, the query of the first (original) user is affected by the second (newly added) user. The OPT values are also plotted as references.

The two plots in Figure 3.9(a) show that the slow start procedure eventually reaches the OPT value. After that, the distributed algorithm will determine an

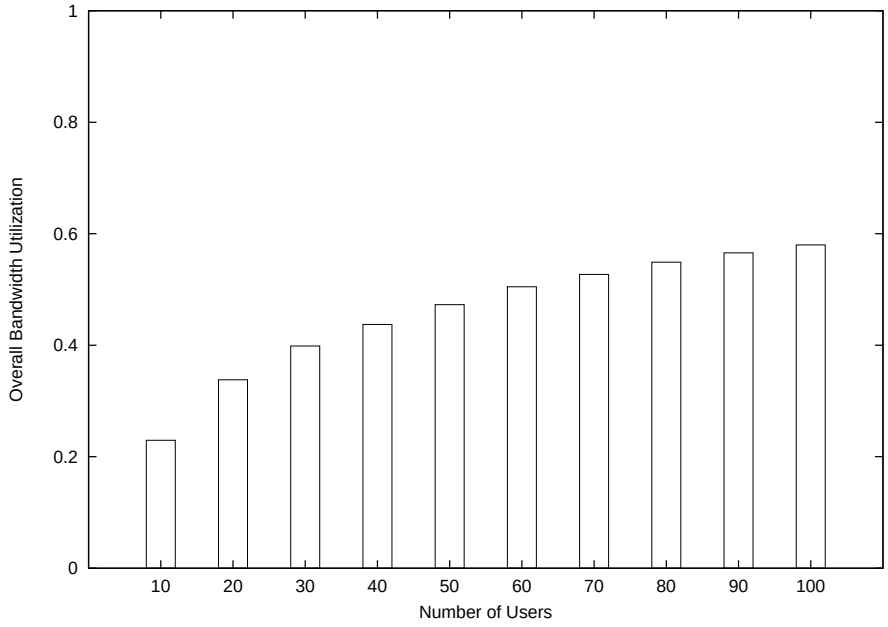
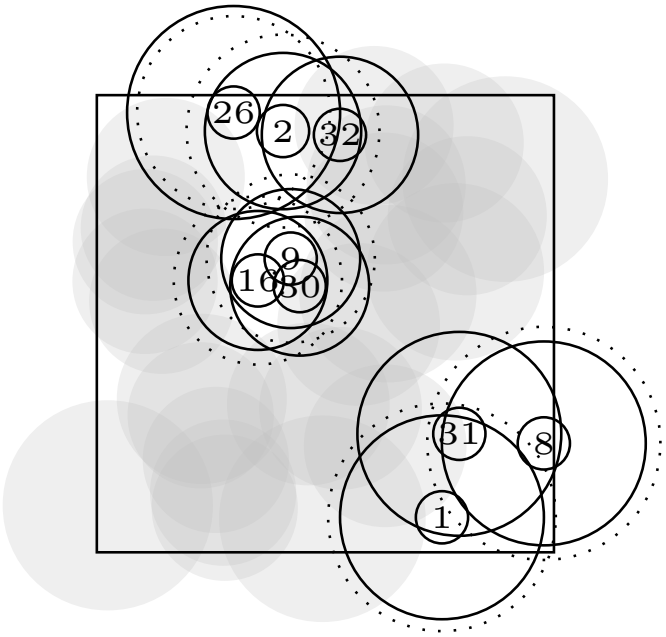


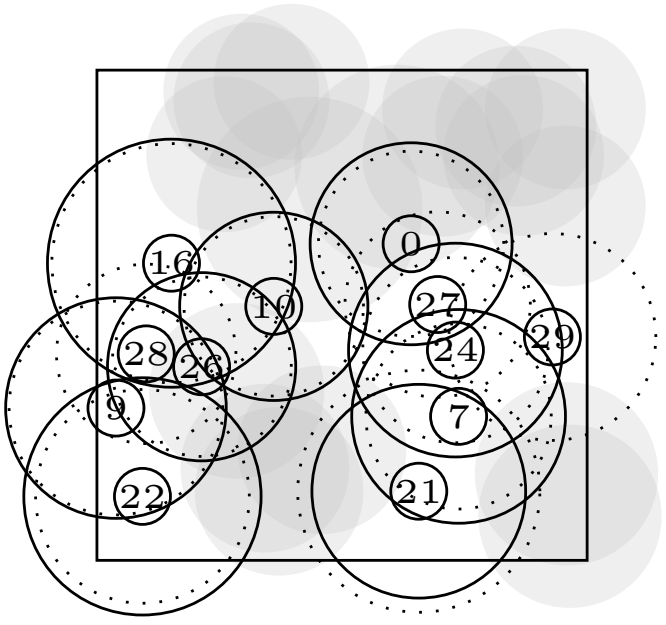
Figure 3.7: Overall bandwidth utilization.

approximation to the OPT value with LOCAL algorithm. The approximation is good since it coincides with the OPT value for most of the cases, *e.g.* for user 9 and 30. Although the approximation is below the OPT value for user 1 and 31, the difference between them is small. Thus, based on this approximation, the `increaseRadius` procedure is able to achieve the OPT value very quickly. Besides, we could see that query radius of the original users (user 1 and 9) begin to drop when the new query grows large enough to produce an interaction between them. For user 1, the interaction takes place when the new query 31 is above the OPT value. As a result, they immediately find the bottleneck between them (also for user 8, as shown in Figure 3.8(a)). However, the interaction happens quite early for users 9 and 30. This could be explained by the fact that different sensors emit `<saturated>` messages at different time when the query is expanding until the actual bottleneck is found. Once the first `<saturated>` message is received, user 30 quits the slow start procedure and increases its query linearly, so the time required to achieve the OPT value is significantly increased (from round 250 to 350).

The two plots in Figure 3.9(b) records the query radii of user 2, 26 and 32 in Figure 3.8(a). The higher one shows the interaction between query 2 and 32. It is shown in order to be compared with the lower one, which is more interesting. In this plot user 26 and 32 have different OPT values because they are not sharing a common bottleneck sensor, thus, not directly interacting with each other. However we see that once the new user 32 achieves its OPT values, query 26 increases its radius and adjusts it to its own OPT values, which is higher than its original OPT value. This can happen because user 2 is forced to shrink its query, which eliminates the bottleneck between user 2 and 26, as shown in Figure 3.8(a).

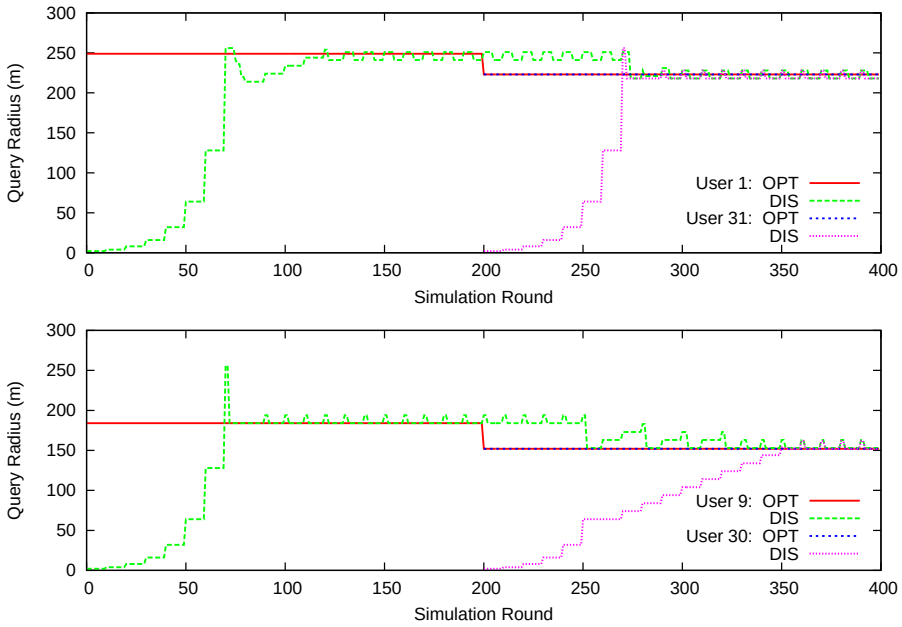


(a) User 30, 31, 32 added.

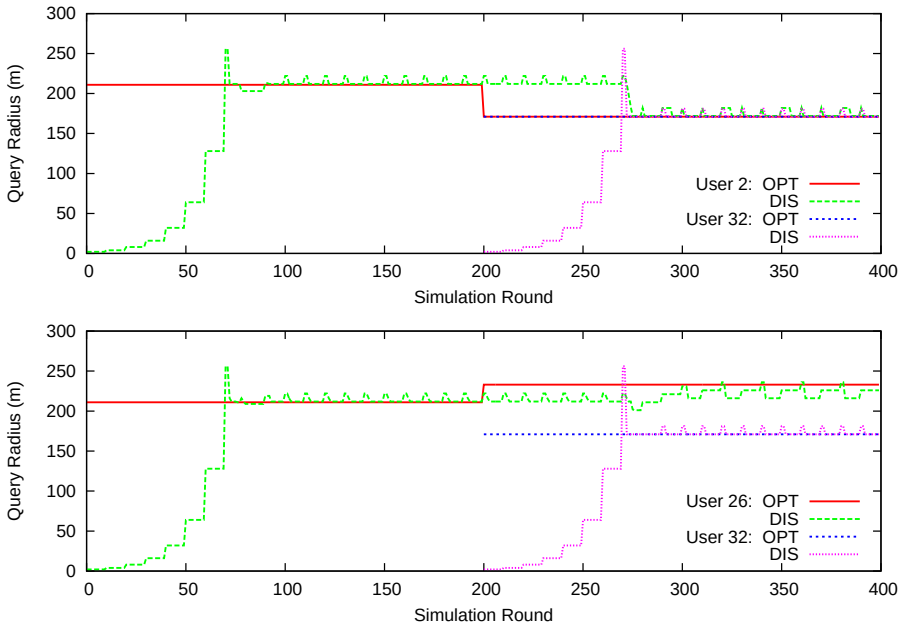


(b) User 27, 28, 29 removed.

Figure 3.8: Query radii dynamics.



(a) User 1 and 9 with user 31 and 30 affecting them, respectively.



(b) User 2 and 26 with user 32 affecting them.

Figure 3.9: Evolution of query radii decided by DIS algorithm.

3.6 Related Works

A basic requirement of the query allocation problem is congestion control which has been studied in WSN in recent years. Most proposed solutions have focused on the transport layer [WSL+06b], and dealt with providing fairness for sources (sensors) *i.e.* allocating for each sensor a fair amount of bandwidth [RGGP06]. In contrast, we investigate both maximality and fairness objectives *in favor of users*, *i.e.* allocating for each user a proper query range to achieve global optimality. We emphasize that it is not necessary to cover every sensor with at least one query, instead, data from sensors within a reasonable query range should not be dropped due to congestion. As a consequence, some sensors far from any fireman do not have to generate data. The idea behind this is obvious: only when a fireman is near to a certain position, the data from this position is meaningful, under the considered application scenario.

Although we have employed a mechanism similar to the slow start procedure in TCP, there exists an important difference. On detecting a potential congestion state, our protocol shrinks the related queries to a theoretically safe value based on the LOCAL algorithm to avoid the congestion, while TCP-like mechanisms set the new contention window to an empirical value, for example, to the minimum value, half of the current value or the slow start threshold.

Paper [MA06] also deals with the problem of sharing a WSN among multiple users but it emphasizes more on the efficiency. Algorithms have been proposed to merge the queries from different users to meet the bandwidth constraint of the WSN. This work and ours are complementary since we have dealt with different aspects of the same problem. In fact, it could be interesting to apply fairness constraints to a WSN with merged queries.

Max-min fairness has been studied within different contexts such as in wireless ad hoc networks [HB01] or wireless mesh networks [AB06]. Especially, it has been studied for WSNs in [SK04], but at the MAC layer. Besides, network utilization has been maximized with max-min fairness in [SK07]. This last paper is very close to ours in that both achieve a maximum network utilization under bandwidth constrains and provide max-min fairness. However, fairness is based on traffic flows on each sensor and only one user is considered in [SK07]. Fair capacity sharing between multiple users has never been formalized and studied before, to the best of our knowledge.

3.7 Summary

In this chapter, we have dealt with the problem that multiple users have to adjust their query ranges according to max-min fair rules in order to meet the capacity constraint of a WSN.

We have proposed a query model based on a disk centered at a user with variable radius and derived analytically the max-min fair radius for each user for two-user case. For general multi-user case, a distributed algorithm was proposed to enable each user determining a near-optimal radius most of the time with information

provided by sensors under its query coverage.

Some future work is necessary to solve the problem thoroughly. First, simulations show that the overall bandwidth is not fully utilized. The query model could be modified to match better the capacity of a region. Besides, the relationship between fairness and bandwidth utilization in WSN with multiple users deserves further investigation. Research efforts are needed if user mobility is considered. Simulation results show that the algorithm sometimes reacts slowly to topology changes, which would hardly be acceptable in mobile user scenario.

Chapter 4

Reformulating The Problem: A Discrete Query Model

"They must often change, who would be constant in happiness or wisdom."

– *Confucius*

In previous chapters, the mobile user WSN and one of its most typical applications, namely, the counter-emergency facilitating application, have been discussed. An example fireman scenario has been given in Chapter 3. Max-min fair query allocation problem has been formulated with a continuous query model and analytical results and distributed algorithm have been given. In this chapter, we discuss a discrete version of the same problem which is more complex than its continuous counterpart.

This chapter is organized as follows. The problem is reformulated with a hop-based query model in Section 4.1. Then we present the hardness result and its proof for one of the problems in Section 4.2. Distributed algorithms are presented and evaluated in Section 4.3 and Section 4.4, respectively. We summarize this chapter with Section 4.5.

4.1 Model and Problem Formulation

This section presents a model dedicated to mobile-user wireless sensor networks where some users have a local query on the sensor nodes. We confine our study with the following assumptions: (i) Each user tends to set its query range as large as possible. The query range is measured by hop numbers such that a j hop query range will cover all j hop neighbors of the user. As now the query range is discrete, we use j instead of r in the previous chapter. (ii) Shortest path routing is assumed and users do not forward data for sensors. (iii) The requested data reporting rate should at least be equal to a predefined threshold in order to detect certain real time events and no in-network data aggregation or compression is employed.

4.1.1 System Model

We consider a set V of l nodes and a set S of m users. A communication graph $G = (V \cup S, E)$ could be formed where there is a link between two elements of $V \cup S$ if the distance between them is less than a fixed value. Each node k admits a limited and quantifiable amount of resources c_k it is able to provide to some users.

The set of nodes providing resources to a user i is noted as $V_{j_i b}$. The amount of resources on node k that user i consumes when its query range is set to j is denoted as w_{ijk}. If the node i does not belong to $V_{j_i b}$, we have $w_{ijk} = 0$. We assumed that all nodes at less than j_i hops to user i have to provide a certain amount of resources. Note that if a node $k \in V_{j_i b}$, then all nodes along the data forwarding path $(k \rightarrow_G i)$ are also in $V_{j_i b}$ since they consume resources in forwarding the data from k. As we also assume a shortest path routing mechanism is in use, the amount of resources a node k has to spend, as a result of the impact of i, is equal or larger than its upstream nodes along the route if the data flows towards the user. This holds as long as we assume there is no compression or aggregation on the data along the paths. This model could be seen as a special case of the traffic model proposed in [LH05], which, together with another well-known result on the capacity of sensor networks [MDMLN03], shows that the number of supported nodes is bounded by the capacity of the user and its immediate neighbors.

Similarly to Chapter 3, a *configuration* is a set of query ranges chosen by all users, noted as $C = \{j_i | i \in S\}$. We say that a configuration is *feasible* if $\forall j_i \in C$, user i is (j_i, b)-supported. This implies that the amount of bandwidth captured by users on every node is less than the node capacity and could be expressed as $\forall k \in V, \sum_{i \in S} w_{ijk} \leq c_k$. The set of feasible configurations will be referred to as $\mathcal{C}$.

4.1.2 MMKP Formulation of Problems

The discrete query allocation problem with maximizing the sum utility of each user, as will be presented soon, is exactly the classical MMKP and could be solved by existing MMKP algorithms. However, all of the these algorithms consider solving the problem in a centralized fashion. Since the problems that motivated this work are distributed in nature, a distributed algorithm is preferred. We shall propose such an algorithm in this chapter. Besides, we also formulate the MMKP problem with max-min fairness as its objective which is novel and interesting since it shows the possibility of formulating different optimization objectives under a uniformed MMKP framework.

4.1.2.1 General MMKP formulation

Let each possible query range of user i corresponds to an item to be selected and the value of the items could be the hop-distances in G from 0 to the diameter of G noted d_G. Since each user is allowed to select only one query range at a time, items could be seen as grouped into m classes and each class corresponding to a certain user. Thus we actually have the multiple choice constraint that exactly one item is to be selected within each class. A binary variable x_{ij} is associated with

query range j for user i where $x_{ij} = 1$ indicates that user i sets its query range to j and $x_{ij} = 0$ otherwise. The amount of resources provided by a node k to a user i when i takes each of its possible query range $j \in \{0, 1, \ldots, d_G\}$ will be referred to as a set $\{w_{i0k}, w_{i1k}, \ldots, w_{id_Gk}\}$ and each w_{ijk} could be mapped to the kth dimension of weight of an item j in class i. Each node k forms a constraint dimension with its available resource c_k. As there are many nodes in the system, we finally have multi-dimensional constraints of MMKP.

In general, we are interested in maximizing the query range of all users. However, other optimization objectives are possible, such as achieving various kinds of fair configurations among the users. Thus a generalized MMKP could be formulated with a general objective:

$$\begin{aligned} \text{Achieve:} \quad & \text{General Objective} \\ \text{Subject to:} \quad & \sum_{i \in S} \sum_{j=0}^{d_G} w_{ijk} x_{ij} \leq c_k, \ k \in V && (4.1) \\ & \sum_{j=0}^{d_G} x_{ij} = 1, \ i \in S && (4.2) \\ & x_{ij} \in \{0, 1\} && (4.3) \end{aligned}$$

We suggest in the following two special objectives dealing with different optimization goals.

4.1.2.2 MNU problem formulated as MMKP

Maximizing the sum of users' query range is our first optimization objective. The query range reflects the utility a user could get from the network, so we define the **Maximal Network Utility** (MNU) problem which find the feasible configuration where $\sum_{i \in S} j_i$ is maximized. This problem is known to be NP-complete and could be expressed as:

$$\begin{aligned} \text{Achieve:} \quad & \text{MNU}(\mathcal{C}) && (4.4) \\ \text{Subject to:} \quad & (4.1), (4.2), (4.3) \end{aligned}$$

4.1.2.3 MMF problem formulated as MMKP

Another global objective deals with fairness since allowing a fair sharing of resources is a constant concern of network designers against selfish behaviors. We are interested in max-min fairness. However, as already mentioned, the max-min fair solution defined in [RB06] may not exist in a discrete feasible solution set [HB01]. The **Lexicographic Max-Min Fairness** concept generalizes the above definition. The merit of the lexicographic max-min fairness is that it is equivalent to the previous definition on convex solution sets and exists on general solution sets as well, although the uniqueness of the solution is not ensured [ST00]. Since under our problem settings, the parameter under consideration is the hop numbers which are discrete thus the

solution space is non-convex, we shall adopt the lexicographic max-min fairness as our max-min fair notion. In the sequel, we shall use MMF for short of lexicographic Max-Min Fairness.

The MMF has been employed in formulating various resource allocation problems in the networking area and general MMF concepts and formal problem formulations, algorithms as well as example design problems could be found in [NPar, OPT05]. Lexicographic max-min fairness is also used in [CFX07] as the fairness notion for allocating bandwidth to the sensors and an iterative linear programming solution is proposed. Unfortunately, none of their algorithms apply to non-convex problems, which is the case in this chapter.

How to allocate discrete bandwidth shares to the users in a multicast network was studied in [ST00]. However, as searching the MMF allocation is proved to be NP-hard, a weaker form of max-min fairness, named as maximal fairness, was introduced. The optimization problem so formed could be solved by a polynomial complexity algorithm. The MMF problem was solved by a genetic algorithm in [LMC04] and later through a dual-objective tabu search in [LC07]. Both algorithms obtain nice approximate to the MMF allocation of discrete bandwidth layers in a multicast network. Auxiliary linear variables and inequalities have been introduced in [OS06] to reformulate the MMF problem so that it could be solved efficiently. Although remarkable results have been obtained by the proposed algorithms, it could be very difficult to implement them in a distributed way in a networking environment. Thus, in the following, we shall formulate the MMF optimization problem within the MMKP framework then propose a distributed algorithm to solve it.

Consider two feasible configurations $C_1, C_2 \in \mathcal{C}$. Define now an ordering operator $\langle\cdot\rangle$ on a configuration C that sorts the elements of C in ascending order. The configuration C_1 is lexicographically greater than C_2, noted as $C_1 \succ C_2$, if there exists an index i such that the i^{th} element of C_1 is greater than the i^{th} element of C_2 and for all indexes $j < i$, the j^{th} elements of both C_1 and C_2 are equal. We say C_1 is greater than or equal to C_2 if $C_1 \succ C_2$ or $C_1 = C_2$, noted as $C_1 \succeq C_2$. The MMF optimization problem is to search for the configuration C_{MMF} such that, for all other configurations $C \in \mathcal{C}$, we have $C_{MMF} \succeq C$. The MMKP formulation of this problem is shown as follows:

$$\begin{aligned} \text{Achieve:} \quad & \text{MMF}(\mathcal{C}) \\ \text{Subject to:} \quad & (4.1),\ (4.2),(4.3) \end{aligned} \tag{4.5}$$

Searching for MMF solutions in discrete solution spaces has been proved to be NP-hard [ST00].

4.2 NP-hardness Proof

While searching for MMF solutions in discrete solution spaces has been proven to be NP-hard [ST00], we will prove that MMKP-MNU is also NP-hard.

Note that the NP-hardness of MMKP-MNU seems straightforward as it is related with MMKP. However this is misleading because, (i) the amount of bandwidth a

sensor k has to spend, as a result of the impact of i, is equal to or greater than its upstream sensors along the data gathering path. This holds as long as we assume there is no compression or aggregation is in use. (ii) for a given sensor, its bandwidth consumption as a result of impact of i, is non-decreasing when the query range of i increases, *i.e.* $w_{ijk} \geq w_{ij'k}$ if $j \geq j'$. These unique characteristics make the resulted MMKP a special case thus possibly easier to solve. Thus, a rigorous proof is needed.

Theorem 1. *MMKP-MNU problem is NP-hard.*

We prove Theorem 1 by proving the corresponding decision problem is NP-complete.

For a graph $G = (V, E)$ and a subset $A \subseteq V$, we denote by $G[A]$ the subgraph induced by A, *i.e.* $G[A] = (A, E \cap (A \times A))$.

MMKP-MNU Decision Problem

INSTANCE: A graph $G = (V \cup S, E)$ such that $E \cap (S \times S) = \emptyset$, a capacity function $c : V \longrightarrow \mathbb{R}^+$, for each sensor $k \in V$ and for each sink $i \in S$ a path $k \rightarrow_{G_i} i$ with $G_i = G[V \cup \{i\}]$ of length d_{ki} and a positive integer $K \in \mathbb{N}$.

QUESTION: Is there an impact function $j : S \longrightarrow \mathbb{N}$ such that:

(*i*) $\sum_{i \in S} j(i) \geq K$, *i.e.* the sum of impact is at least K,

(*ii*) for every $k \in V$ we have that the number of data forwarding path which go through the sensor k is lower or equal to the capacity of this sensor, *i.e.* $|\{(k', i) \in V \times S : d_{k'i} \leq j(i) \text{ and } k \in k' \rightarrow_{G_i} i\}| \leq c(k)$.

Independent Set Decision Problem

INSTANCE: A graph $G = (V, E)$ and a positive integer $K \leq l$.

QUESTION: Does G contain an independent set of size K or more, *i.e.* a subset $V_1 \subseteq V$ such that $|V_1| \geq K$ and such that no two vertices in V_1 are joined by an edge in E, or more formally $E \cap (V_1 \times V_1) = \emptyset$?

Proof. Given an instance of impact range allocation problem and an impact function j, verifying the conditions (i) and (ii) is clearly polynomial in the size of the problem. Hence, MMKP-MNU problem belongs to NP.

We now transform independent set problem to our problem. Without lost of generality, we can assume that the graph from the independent set instance does not have isolated vertex. Given a graph $G' = (V', E')$ and a positive integer $K \leq l$, we construct the incidence graph G from G' by adding a vertex α connecting it to every edge of G' as follows.

- Let $S = V'$, $V = E' \cup \{\alpha\}$ and let $G = (V \cup S, E)$ where $E = \{\{x, e\} : x \in V', e \in E' : x \in e\} \cup \{\{e, \alpha\} : e \in E'\}$, as shown in Figure 4.1. Notice that we have $E \cap (S \times S) = \emptyset$.

- Define the capacity c as $c_k = 1$ for $k \in V'$.

- For any edge $e \in E'$ and for any vertex $u \in V'$, we define the path $e \rightarrow_G u = [e, u]$ if $u \in e$ and otherwise we take any shortest path from e to u in $G[V' \cup \{u\}]$. Notice that we have $\alpha \in e \rightarrow_G u$ if and only if $u \notin e$.

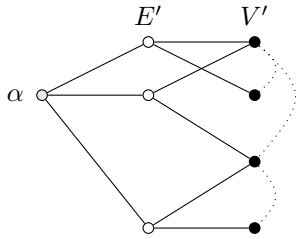


Figure 4.1: Construction of G from G'. G' is composed of solid nodes (V') and dotted edges (E'). E' is also used to denote auxiliary nodes (empty circles) since each auxiliary node corresponds to an edge in G'.

- For any $x \in V'$, we take an arbitrary $e \in E'$ such that $x \in e$ and we define $\alpha \rightarrow_G x$ as $[\alpha, e, x]$.

Notice that by construction and as G has no isolated vertex, for any vertex $v \in V$ we have $d_{\alpha x} = 2$ and that, for any edge $e \in E$ and for any vertex $v \in V$ it holds $d_{ev} = 1$ if and only if $x \in e$. Notice also that it is clear that the instance of query range allocation problem can be constructed in polynomial time. We claim that G has an independent set of size at least K if and only if there exists an impact function $j : S \longrightarrow \mathbb{N}$ which fulfills the conditions (i) and (ii).

For the forward implication, let $V_1 \subseteq V$ an independent set of size at least K, then we define the impact function $j : V' \longrightarrow \mathbb{N}$ such that $j_v = 1$ if $v \in V_1$ and $j_v = 0$ otherwise. As V_1 is an independent set of size at least K, it follows that the condition (i) is clearly fulfilled. On the other hand, for any $e \in E'$ it holds $|e \cap V_1| \leq 1$. Thus, for any $e \in E'$, we have that $|\{(x, i) \in V \times S : d_{xi} \leq j_i \text{ and } e \in x \rightarrow_G i\}| \leq 1$. As for any $i \in S$ we have $d_{\alpha i} = 2$, we can conclude that condition (ii) is fulfilled.

For the backward implication, let j an impact function $j : S \longrightarrow \mathbb{N}$ which fulfills the conditions (i) and (ii). First of all, we show that for any $i \in S$, we have $j(i) \in \{0, 1\}$. Indeed, assume by contradiction that there exists $i \in S$ such that $j(i) \geq 2$, thus we have $d_{\alpha i} = 2$ as we have no isolated vertex. As a result, there exists $e \in E'$ such that $\alpha \rightarrow_G i = [\alpha, e, i]$. But now as $d_{ei} = 1$, we have that $\{(\alpha, i), (e, i)\} \subseteq \{(x, m) \in V \times S : d(x, m) \leq j(m) \text{ and } e \in x \rightarrow_G m\}$ which leads to a contradiction with the fact that $c_e = 1$. Now, we can conclude that the set $V_1 = \{x \in V' : j(x) = 1\}$ is an independent set of size at least K. Indeed, assume that V_1 is not an independent set, then there exists $x, y \in V_1$ such that $e = \{x, y\} \in E'$, which implies that $\{(e, x), (e, y)\} \subseteq \{(x, m) \in V \times S : d_{xm} \leq j(m) \text{ and } e \in x \rightarrow_G m\}$ which leads to a contradiction with the fact that $c_e = 1$. Now, as $\sum_{i \in S} j(i) \geq K$ and as for any $i \in S$, we have $j(i) \in \{0, 1\}$, it is clear that $|V_1| \geq K$. □

4.3 Algorithms

In this section, we propose several algorithms to solve the problems discussed in Section 4.1. We are interested in developing distributed algorithms that run on each user and node enabling the users to decide proper query ranges dynamically.

Algorithm 4: Exact Algorithm for MMKP-MMF

```
Input  : V, S, d_G, w_ijk (i ∈ S, j ∈ U), c_k (k ∈ V)
Output: MMF configuration set C_MMF
C_0 ← {S_i = (j_i ← 0, s_i ← active)},  i ∈ S
C_MMF ← {C_0}
for j ← 1 to d_G do
    C'_MMF ← ∅
    for ∀C ∈ C_MMF do
        A ← ∅
        for ∀S_i ∈ C and s_i = active do  j_i ← j
        for ∀k ∈ V do
            if c_k < Σ_i w_ij_ik then
                A ← A ∪ {S_i : w_ij_ik > 0, s_i = active}
            end
        end
        for  t ← 1 to |A| do
            for ∀A' ∈ P(A) and |A'| = t do
                C' ← C
                for ∀a ∈ A' do
                    j_a ← j_a − 1, s_a ← stop
                    if feasible (C') then
                        C'_MMF ← C'_MMF ∪ {C'}
                        feasiblefound ← true
                    end
                end
            end
            if feasiblefound then  break
        end
    end
    C_MMF ← lexopt(C_MMF ∪ C'_MMF)
    if ∀C ∈ C_MMF, ∀S_i ∈ C, s_i = stop then  break
end
```

However, since the problems are NP-hard, approximate but fast algorithms are practically needed. Nevertheless, we shall also discuss the centralized exact algorithms for demonstrating the quality of solutions obtained by the distributed approximate algorithms.

4.3.1 An exact algorithm for MMF

In the MNU problem, the objective is to maximize the sum of the utility which makes the problem exactly same as the traditional MMKP studied in previous works. Different types of exact algorithms have been developed to solve this problem and off-the-shelf general solvers could be used. While on the contrary, no effective algorithm has been proposed for the MMF problem. We propose an algorithm which is able to find all the exact solutions to this problem.

The proposed algorithm is based on the branch and bound idea and it operates by obtaining feasible partial solutions at each round and selecting only the lexicographically largest ones for the next round of branching. Each round is divided

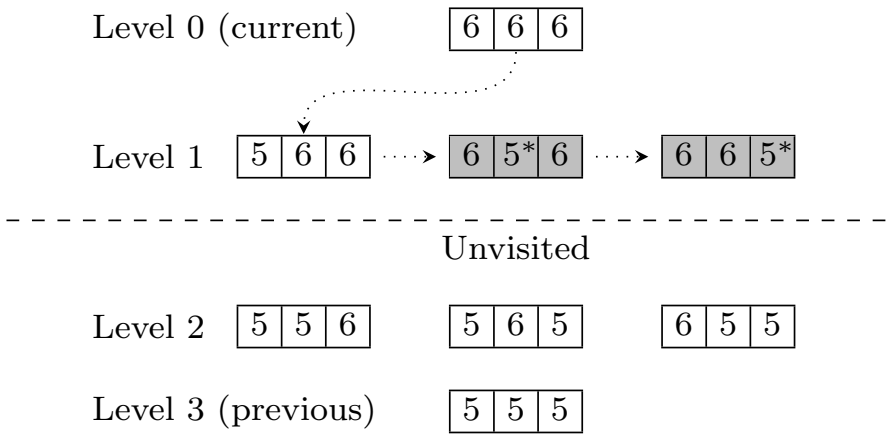


Figure 4.2: The fixing procedure checks the solutions between the previous (feasible) solution and the current (infeasible) one.

into two stages: a greedy lower bound discovering stages and a fixing stage. The algorithm starts at a trivial solution where the query ranges are set to the lowest levels and all users are marked as 'active'. Each round begins with a discovering stage where the algorithm simply increases the levels of all active users simultaneously. Then the current configuration is checked for its feasibility against all the constraints. If any violation is found, the algorithm enters the second stage. At this user, we know that (i) the previous solution is feasible but the current one is not and it tries to fix the current configuration with the fixing stage. Since we assume that the resource consumption is a non-decreasing function of the query range level, the only way to fix the current solution is to decrease the query range level of one or more users. (ii) We should decrease only one level for a particular user due to the max-min fairness constraint. (iii) If a feasible solution could be derived from the current one by decreasing only one user, we do not need to check further to see if there are feasible solutions could be possibly derived from the current solution by decreasing two or more users, since it is impossible to develop a better solution from those already lexicographically smaller partial solutions. The fixing procedure is shown in Figure 4.2. In the figure, the number in each cell denotes the query range of the user and it has an asterisk if the user is 'stopped'. The gray configurations are feasible ones while others are infeasible. At level 1, the algorithm tries each user to see if decreasing its query range level by one will generate a feasible solution. If no feasible solution is found, the algorithm continues trying level 2, where all combinations of 2 users are examined to see if decreasing the levels of both will generate a feasible solution. This procedure continues until all feasible solutions are found at a certain level (since they are lexicographically identical) and higher levels are no longer visited. The users whose query range levels are decreased at the current round will be marked as 'stopped', meaning they will not be increased in the next round because they have achieved their max-min fair query range levels.

After the fixing procedure, some partial solutions are obtained. Then the algorithm enters the next round and tries to increase the levels of the active users. Then

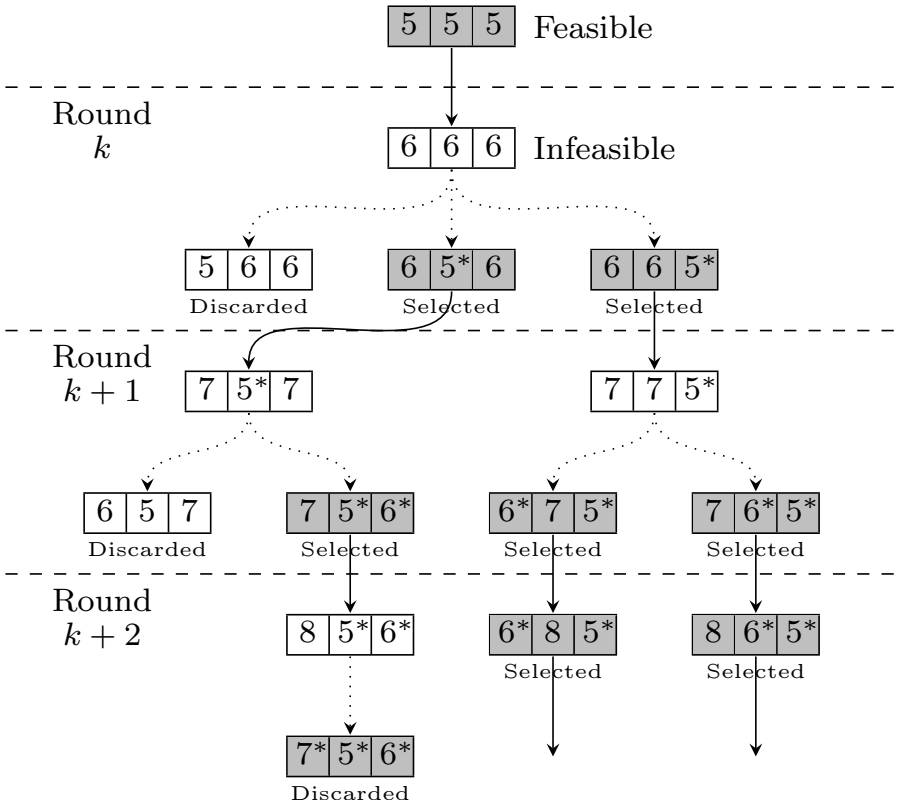


Figure 4.3: Part of the execution paths of Algorithm 4 in a three-user case.

the constraints are checked and if violated, the same selecting and decreasing steps are followed. This procedure should be applied for each partial solution obtained in the previous round. As a result, all the partially optimal solutions are found at each round and the algorithm ends with all the optimal solutions. Figure 4.3 shows part of the execution paths of the algorithm when applied to a problem with three users. Solid arrows in the figure indicate a greedy exploration and dotted arrows indicate a fix operation. The resulted algorithm is formally presented in Algorithm 4. Some notations deserves more explanations. A structure is associated with a user i and we denote it as $S_i = (j_i, s_i)$, which keeps the query range level of i in j_i and its current state *i.e.* active or stopped, in s_i. A configuration of query ranges of all users is noted as C. All active users that contributes to a congested node found at a certain round are put into set A and $\mathcal{P}(A)$ denotes the power set of A. Finally, $\mathcal{U}$ is the set of query range levels which is assumed identical for all users.

At the end of a certain round, all the solutions from the same branch are lexicographically identical, but this is not necessarily true for those solutions derived from different branches. So only the lexicographically largest ones are selected to be further developed. As at the round $k + 2$ in Figure 4.3, three feasible solutions are derived from three branches, but only two of them are selected for the next round. The solution $(7, 5, 6)$ is lexicographically sub-optimal thus is discarded. Function `lexopt` sorts all the configurations lexicographically and returns the largest ones.

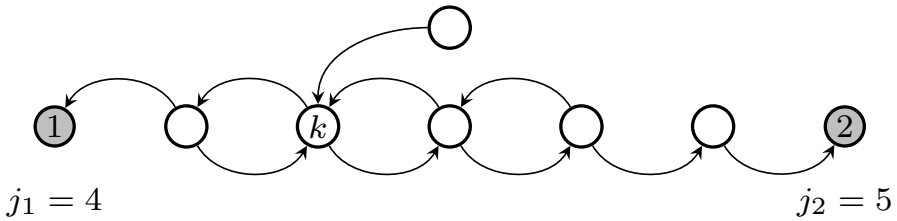
4.3.2 Distributed algorithms for MNU and MMF

As stated in previous sections, fast and distributed algorithms are preferred if one wants to solve the two optimization problems online. We propose two such algorithms in this section. Both algorithms require only local information from nearby nodes and can be formulated into one uniformed framework. The basic idea of the algorithms is to solve a much smaller localized problem at each node. To this end, each node keeps track of all the traffic passing through itself. With this information, when a node is impacted by several users thus has to provide bandwidth more than its capacity, it is able to formulate a local MMKP with only the related users as parameters and a single bandwidth constraint. This problem is usually much smaller and could be solved quickly. Then the node notifies the related users with the solution. Finally, a user adopts one of the possibly multiple solutions as its new query range. In the rest of this section, we first give detailed discussions on the key techniques we shall employ, then we propose a unified algorithm for both MNU and MMF problems.

4.3.2.1 Local MCKP solution

When the node is required to provide resources more than its capacity, it formulates a smaller MMKP problem with its local traffic measurements and a single constraint, or equivalently, an MCKP problem. This MCKP is to be solved in a centralized fashion thus either exact algorithms or heuristics can be exploited. Various algorithms have been proposed to solve an MCKP with sum utility maximization objective (MKP-MNU). We again adopt the GLPK mixed integer solver to solve this MCKP-MNU problem as part of distributed algorithm to be proposed for the MNU problem. On the other hand, no algorithm has been proposed for an MCKP with MMF objective (MKP-MMF). Since the MCKP is a special case of MMKP, it is possible to apply the exact algorithm we previously proposed directly to solve this MCKP. However the exponentially growing computation cost of the exact algorithm may be hardly supported by the nodes, even if the number of users impacting a congested node is reasonable. In the following, we propose a heuristic for such an MCKP problem. This heuristic will be integrated into the distributed algorithm to be proposed.

The basic idea of the heuristic is to start at a trivial configuration $C = (0, \ldots, 0)$ with all users marked as 'active'. Then it discovers a partial feasible solution greedily round by round. When a node examines the partial solutions, it relies on an estimation of the traffic it should carry. Figure 4.4 illustrates an issue related with this estimation. Suppose there are two users $i = 1$ and $i = 2$ having impact on the node k with a current configuration $C = \{j_1 = 4, j_2 = 5\}$. The matrix M_k is the traffic load measured by node k where the number of rows is the number of users impacting k and the number of columns is the number of different query ranges these users may choose. Each element t_{ih} of M_k denotes the bandwidth the node k has to spend in forwarding traffic from all nodes at exactly h hops away of the user i. In other words, for a user i with query range j, the total bandwidth consumption of k may be expressed as $\sum_{h \in [0 \ldots j]} t_{ih}$. As illustrated in the Figure 4.4, the node k



$M(k)$

h	1	2	3	4	5	6
$i = 1$	0	10	20	10	10*	10*
$i = 2$	0	0	0	10	10	10*

Figure 4.4: Traffic measurement on node k.

does not have to spend bandwidth for user 2 for all values of $h = 1, 2, 3$ because it is actually located at 4 hops away of user 2. The real issue occurs for $t_{1,h=5}$, $t_{1,h=6}$ and $t_{2,h=6}$. In fact, node k can only estimate the traffic for these ranges because it has not experienced such configurations yet. In the current study, we suggest to use a very basic estimation which is the traffic load for the last known hop number. But we could also expect the number of nodes at h hops from a user and whose traffic has to be carried by a node is growing with h. This traffic estimation seems to be still an open problem.

The way to detect this approximate solution is based on the measurement of the *saving* presented in [KLMA02]. The t_{ih} is equivalent to the saving but better expressed as the extra cost when the query range $j - 1$ becomes j. Recall the algorithm is round-based, and at each round, a query range level j is examined to see if it is possible to generate a feasible solution with all or some users set to j. The active users are sorted by their extra costs at the corresponding query range in ascending order. Then the algorithm increases their query range levels by one, one user after another, from the less costly user to the most costly one. If all users are increased without violating the bandwidth constraint, the algorithm enters the next round. Otherwise, the first user that violates the constraint and all users after it are marked as 'stopped' before entering the next round. The whole algorithm stops once all users are marked as 'stopped'. Algorithm 5 formally describes this heuristic.

The input S' of the algorithm is a subset of the users which have impact on the node. Since the algorithm runs on each node k, it considers only one constraint c_k. The `sort` function in the algorithm sorts the active users in A in ascending order with $t_{\cdot j}$ as keys. Finally, the function `feasible` tests if the constraint c_k is violated.

The algorithm is heuristic because at each round, only one possibility of query range assignment is inspected, thus the procedure is prone to be trapped by a local optima. The merit of the heuristic is its polynomial time complexity as compared with an exact algorithm, which is critical for a distributed online algorithm. It is userless for each node to spend too much efforts to reach an optimal local solution because this solution is probably not globally optimal. Thus, we suggest to first give a local solution quickly, which is not necessarily optimal but is expected to be

Algorithm 5: Local MCKP-MMF

input : $S' \subseteq S$, U, $M(k)$, c_k
output: MCKP-MMF configuration C

for $i \in S'$ **do**
  $C \leftarrow \{S_i = (j_i \leftarrow 0, s_i \leftarrow \mathsf{active})\}$
end
for $j \leftarrow 1$ *to* $|U|$ **do**
  $A \leftarrow \{S_i : s_i = \mathsf{active}\}$
  if $A = \emptyset$ **then** break
  `sort` $(A, t_{.j})$
  for $a \leftarrow 1$ *to* $|A|$ **do**
    $C' \leftarrow C$, $j'_a \leftarrow j$
    if `feasible` (C') **then** $j_a \leftarrow j$
    else
      for $a' \leftarrow a$ *to* $|A|$ **do** $s_{a'} \leftarrow \mathsf{stop}$
      break
    end
  end
end
Return C

a fairly good approximation of the optimal solutions. Then, we further explore the global optimal by applying a dynamic query range adaption procedure at each user, as will be discussed soon.

4.3.2.2 Dynamic query range adaption

After the local MCKP problem is solved, the node sends the result to all related users indicating them their new query range levels which are expected to conform with the constraint on this node. However, a user may receive multiple such notifications from multiple nodes. Thus, in order to make the most stringent constraint satisfied, a user needs to adjust its query range level according to the smallest one of all notifications. The side effect of this policy is the query range levels tend to decrease in the long run and a user may not be able to know its optimal query range due to the incomplete information it has. To mitigate this side effects and help the users to jump out of a local optima assigned by Algorithm 5, each user should increase its query range level periodically. Similar effects have been observed and the same countermeasure has been employed in [HS07].

4.3.2.3 Unified algorithmic framework

Now that all the prerequisites are discussed, we are able to give the main algorithms for both the MNU and MMF problems. As stated at the beginning of the this section, the two algorithms could be described in one generic framework, as shown in Algorithm 6.

Algorithm 6: Distributed Heuristic

```
Sink Part : Run at user i
send < level, i, 1 >
while no < adjust-level > message do
    level ← initLevel()
    send < level, i, level >
end
level ← adjustLevel()
while true do
    while no < adjust-level > message do
        level ← increaseLevel()
        send < modify-level, i, level >
    end
    level ← adjustLevel()
    send < modify-level, i, level >
end
Sensor Part: Run at sensor k
while true do
    if congested() then
        C ← solveMCKP()
        for ∀i : j_i ∈ C do
            send < adjust-level, j_i > to i;
        end
    end
end
```

In this algorithm, when a user wants to set its query range in the network, a query range message < `query range`, i, j > will be sent. It notifies any node who receives it that a user i has set its query range to *level*. The < `modify-level`, i, *level* > message is used by i to modify its query range to *level* when needed. The < `adjust-level` > message is sent by the congested nodes to the users associated with them to notify the users their new query range levels.

At the user side, the algorithm consists of two phases. The first one is a well-known *slow start* phase and it is used to initially discover a congested node. The query range starts at the unit level and increases according to a certain strategy until the user is alerted that one or more nodes covered by its impact is congested. The function `initLevel` manages the initial query range increasing and it uses an exponential growth strategy, *i.e.* the query range is initialized to level 1 and increased by doubling its previous value each time it is called. On receiving the first message indicating a congestion on the node, the user sets its query range to the suggested level if it is smaller than the current one, then the system enters the second phase. Function `adjustRadius` fulfills this operation by returning the smaller of the current query range level and the one suggested by a congested node. In the second phase, a user tries to increase its query range periodically, in order to explore the

Table 4.1: Simulation parameters.

Node distribution	Uniform
User distribution	Uniform
MAC layer	802.11
Wireless Tx/Rx radius	30m
Bandwidth constraint	2pkt/s
Requirement data rate	1/60pkt/s
Simulation time	800s

optimal levels. This is handled by the `increaseLevel` function and it increases the query range linearly. Eventually, the user is alerted again by an overloaded node, then it decreases its query range according to the level indicated in the message and resumes increasing later.

At each node, the `measureTraffic` function makes the measurements for traffic coming from nodes at a certain hops away to each user, as discussed in 4.3.2.1. The function `solveMKP` is called when the node finds itself congested by all the users having impact on it. Different algorithms should be used according to the objective of the main algorithm, *i.e.* MNU or MMF. Besides, various algorithms could be used for either objective. For example in this chapter, we use GLPK solver for the MNU objective and Algorithm 5 for the MMF case.

4.4 Performance Evaluation

In this section, we evaluate the proposed algorithms by comprehensive simulations. To this end, we need to implement the algorithms as well as some basic networking functionalities on both the users and the nodes. Here we briefly recall how the network works: the users send their query ranges by constrained flood to the network. Routes to the corresponding users are built at the same time on each node by recording the upstream nodes from which the query range message is received. Then the nodes start to generate data at a certain rate and send it back to the users. Meanwhile, the nodes measure the traffic and calculate the query range levels for users according to the optimization objective. This procedure is implemented in the ns-2 simulator [ns2].

4.4.1 Simulation setup

We consider a densely connected network with uniformly deployed nodes and users. Thus, the network area, number of nodes and the wireless transmission range are selected to create a densely connected network in a rather large region. Table 4.1 summarizes the common parameters. We shall examine the algorithms under several network scenarios with different sizes, as listed in Table 4.2. The density of nodes is fixed and we vary the number of users, number of nodes and the network area.

Please note that the bandwidth constraints are manually set as how many packets

Table 4.2: Simulation scenarios.

Scenario	Number of Nodes	Network Area
1	100	320m×200m
2	200	400m×320m
3	400	640m×400m
4	800	800m×640m
5	1000	800m×800m

a node can transmit per second. With this virtual bandwidth constraint, we no longer need to consider the real bandwidth of the links, the collisions and other link layer issues since we want to concentrate to the algorithmic aspect. Actually, we set the real bandwidth to 1 Mbps which can effectively eliminate most collisions and the overall packet loss is controlled under 5%. In order to demonstrate the behavior of the algorithms, the constraint value should allow most users have a reasonable dynamic range of query range levels, depending on the other parameters. Obviously, if the constraint is too low, most of the users will be limited by the traffic of themselves so that they choose almost the same levels. On the contrary, users are free to choose the largest levels if the constraint is too high.

Finally, in order to eliminate random effects, average results are taken from 100 randomly generated topologies for each simulation and the error bars in the figures show the ±1 Standard Error of the Mean (SEM) of the data set.

4.4.2 Time complexity of the exact algorithm

Our first simulation focuses on the time consumption of Algorithm 4. To do this, we randomly generate 100 topologies for each of the scenarios $1, 2, 3, 4$ in Table 4.2 and put $5, 10, 15, 20, 25$ users in each scenario. Besides, each simulation is run with maximum query range level set to $5, 10$ or 15 in order to show how different parameters impact the time consumption of the algorithm. A personal computer with a CPU at 3.2GHz and 1GB memory is dedicated for the simulations and Figure 4.5 shows the results with each sub-plot corresponding to a maximum query range level setting. As expected, the time consumption grows exponentially with the number of users. Besides, there is no significant difference in the time consumption of different network scenarios when the number of users keeps the same. Furthermore, examine vertically through the sub-plots, we see no significant difference between the time consumption when varying the maximum query range level, provided the number of users is fixed. These results conform with the previous analysis. Finally, the error bars are higher for larger number of users, indicating that the time consumption could be very different and increasing the number of users increases this diversity. Note that we are unable to solve the 25-user case in reasonable time thus no corresponding results could be presented.

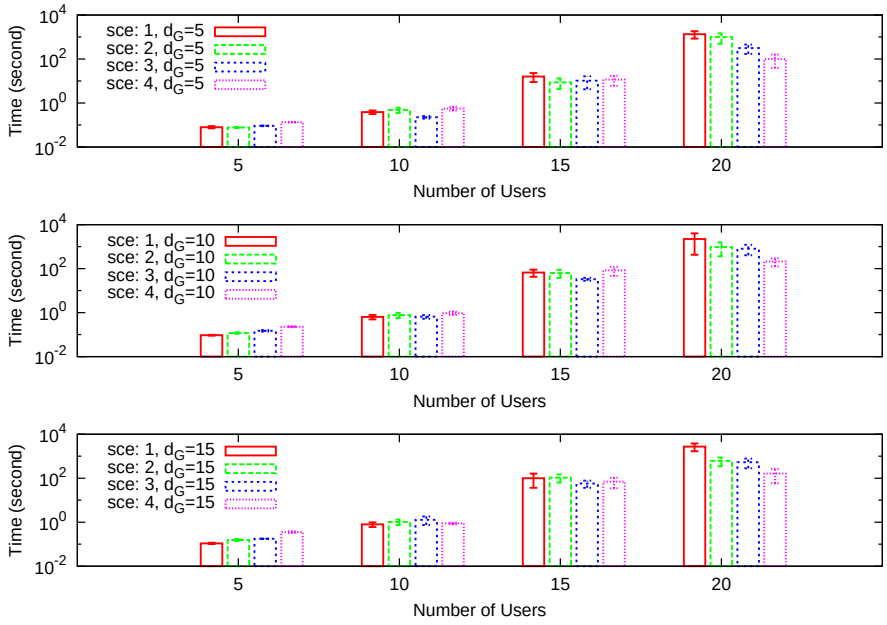


Figure 4.5: Time consumption of Algorithm 4.

Table 4.3: Average query range level comparison.

m	MNU			MMF		
	j_{DIS}	j_{OPT}	j_{UP}	j_{DIS}	j_{OPT}	j_{UP}
5	12.27	12.22	12.89	11.29	11.97	12.84
10	11.47	11.93	12.46	10.44	11.4	12.37
20	10.55	11.81	12.10	9.44	10.52	11.73
30	9.76	–	11.78	8.75	–	11.24
40	9.47	–	11.52	8.38	–	10.90
50	8.67	–	11.21	7.96	–	10.59
60	8.48	–	11.07	7.75	–	10.45
70	8.31	–	10.94	7.55	–	10.28
80	8.03	–	10.71	7.39	–	10.22
90	7.88	–	10.59	7.23	–	10.14

4.4.3 Distributed heuristics in a large network

All the following simulations concern the distributed Algorithm 6. Besides showing the effectiveness of the algorithm, we shall emphasize more on how the two optimization objectives, MNU and MMF, impact on the behavior of the network. We use scenario 5 for all the simulations below since the network is scaling free. The number of users is chosen from the set $\{5, 10, 20, \ldots, 90\}$.

4.4.3.1 Quality of solutions

First, we would like to demonstrate the quality of configurations obtained by the distributed Algorithm 6 through comparing with those obtained by the exact algorithms. To this end, we run two simulations with or without the distributed algorithms turned on. For the latter case, the query range of users are set to the maximal. The approximated solutions could be obtained directly by running the simulation with the distributed algorithms. While on the other hand, the traffic information obtained from the simulation without distributed algorithms is fed to the exact algorithms and the results are considered as optimal solutions.

Average query range levels are shown in Table 4.3. The column m shows the number of users and the column j_{DIS} and j_{OPT} denote the average query range obtained by the distributed algorithm 6 and by the centralized exact algorithms, respectively. Please recall that we use GLPK for the MNU problem and Algorithm 5 for the MMF one. We indicate that the algorithm is unable to finish within a reasonable time for problems with a larger m, *e.g.* $m > 20$, under current simulation settings. For these cases, a bar is put in the corresponding cells in the table and the average query range levels obtained by solving the linear programming relaxation of the original problems are shown in the column j_{UP}. Although such relaxation provides good upper bounds for the MNU problems, it gives very loose upper bound to the MMF ones. Nevertheless, they give us some hints on the solution quality of the distributed algorithm. As shown in Table 4.3, for a small size of user population when the optimal solution could be obtained, *i.e.* $m = 5, 10, 20$, Algorithm 6 achieves fairly good solutions.

4.4.3.2 Congestion resolution capability

Now we show the effectiveness of Algorithm 6 on eliminating congested nodes with the two introduced optimization objectives. We run the simulations with or without query range level control and the query range is set to its maximum on each user for the latter case. The average number of congested nodes are calculated and the results are shown in Figure 4.6. We clearly see that if the network runs uncontrolled, there will be many congested nodes and the ratio of congested nodes increases as more users come into the network. When the query range level is controlled by the Algorithm 6, with either MMF or MNU objective, we see most of the congested nodes are eliminated and the ratio of such nodes increases with the number of users in the network, but now at a lower speed. It is interesting to observe that the MNU algorithm gives a lower congested node ratio than MMF algorithm does and we will discuss about this later.

4.4.3.3 Comparative study on MNU and MMF

In the following, we investigate in detail the differences between configurations obtained by the MNU and MMF algorithm, by comparing the average bandwidth utilization, average query range level, the distribution of users at each query range level and finally, the distribution of nodes on the number of users impacting them.

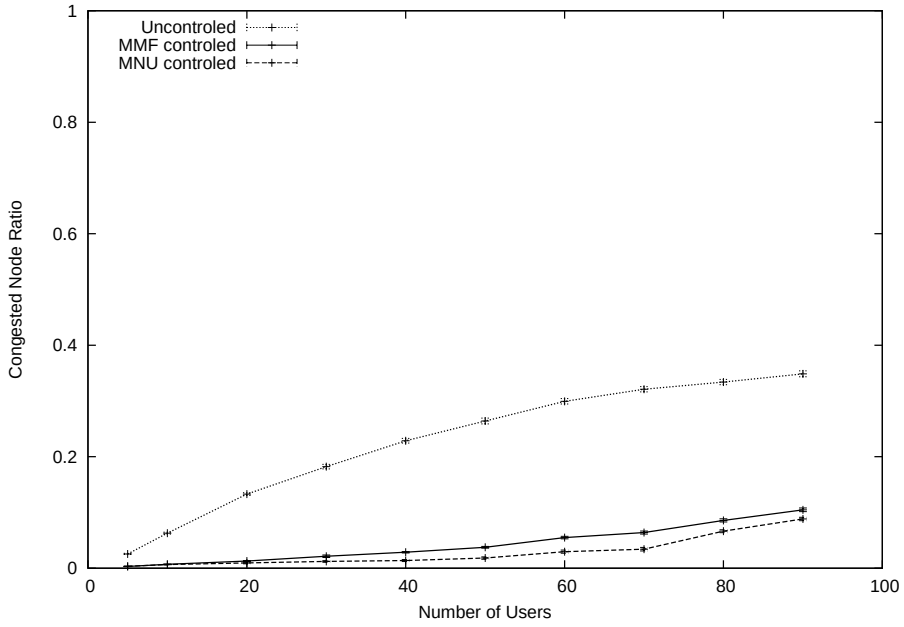


Figure 4.6: Congestion rate of nodes.

Bandwidth utilization We compare the average bandwidth utilization of the nodes when the network is optimized by the MNU and MMF objectives in this paragraph. The average bandwidth utilization is defined as the average of the bandwidth spent on each node for transmitting data to the users.

Figure 4.7 shows this result. We see similar trends as in Figure 4.6 for the three curves. Here we emphasize on the difference between the MMF and MNU algorithm. We see again that the curve of the MMF lies above that of the MNU, indicating that the MMF algorithm results in a higher average bandwidth consumption than the MNU does. This result appears to be paradoxical since we have named the algorithm as MNU. However, it is actually correct because the MNU algorithm aims at maximizing the sum of the query range levels. A maximum sum of query ranges does not necessarily indicate a maximum traffic load, or bandwidth consumption, since the traffic load on each node is a non-linear function of the query range level [LH05] under the local traffic model. This could also explain the similar phenomenon on the congested node ratio in Figure 4.6. Note that the MNU optimization objective indeed gives larger average and maximum query range levels than the MMF one, this could be verified by Figure 4.8.

Actually, we see again the trade-off between fairness and resource consumption: MMF is fair but gives a generally lower network utility and consumes more bandwidth while MNU achieves maximum network utility at a lower bandwidth consumption but at the cost of serving the users unfairly.

It is noteworthy to mention that neither MMF algorithm nor MNU algorithm promises a maximum overall bandwidth utilization and this could be used as an optimization objective to formulate another problem similar to the MNU or MMF problem.

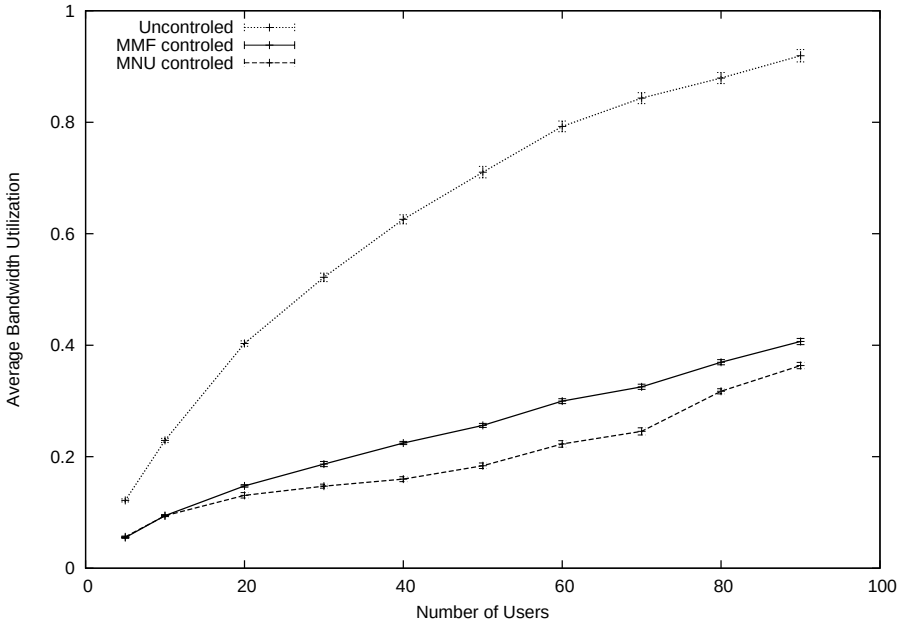


Figure 4.7: Bandwidth utilization of nodes.

Query levels The query range level indicates the utility of a user. We show the differences between the networks optimized by the two objectives in this paragraph, by studying the average and the maximum query range of the users. Figure 4.8 shows that the MNU optimization objective gives larger average and maximum query range levels than the MMF does, as already mentioned. More interestingly, we see different trends of the maximum and average query range level as the number of users increases: the average is monotonically decreasing; the maximum is around the maximum allowed level.

The monotonically decrement of the average query range level is a reasonable effect of bandwidth sharing among multiple users: as more users are in the network, each user has to shrink its impacted area. In general, the MMF policy requires nearby users have similar query range levels and a user is able to have a remarkably higher query range level only when it is geographically far from the others. As a result, MMF objective generally gives users smaller chance to achieve maximum allowed level than the MNU objective does. This explains why the former always gives a maximal query range to the users lower than the latter.

Fairness among users More information about how the users are satisfied could be provided by demonstrating how the users are distributed in the query range level domain, as shown in Figure 4.9. Three cases with 10, 50 and 90 users are investigated and the height of the bar in the figure represents the ratio of the users that have the corresponding query range level. Results of the MMF algorithm and the MNU algorithm are grouped for comparison. We can observe that the MMF algorithm results in a more concentrate distribution than the MNU algorithm does under all the three cases. Actually, the distribution could be used as an indicator of

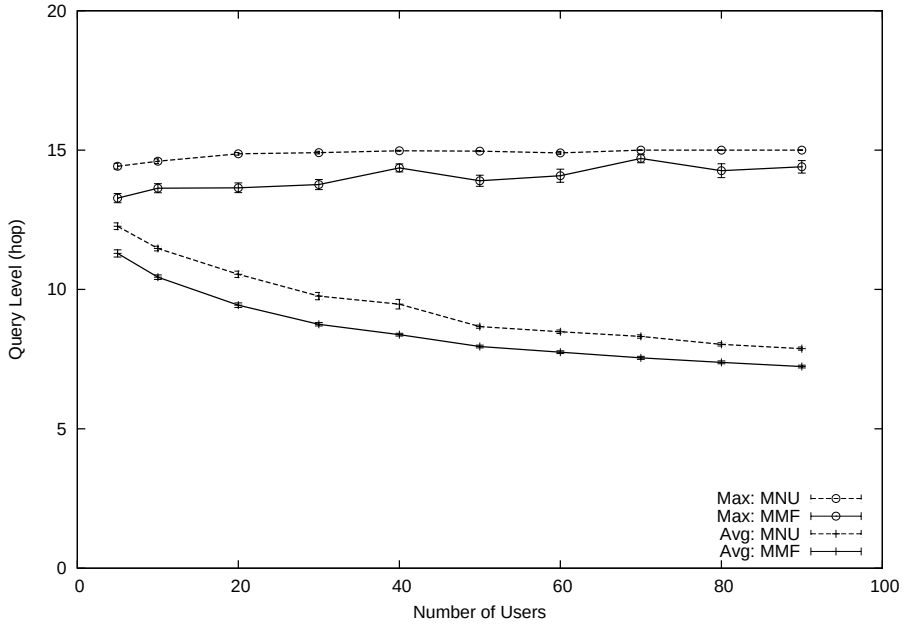


Figure 4.8: Perimeter level of users.

fairness: the more concentrate the distribution is, the fairer the configuration is.

To measure the fairness quantitatively, we employ the *fairness index* introduced in [JCH98] as the following ratio f. If $C = (x_1, x_2, \ldots, x_m)$ is a configuration of m users, then:

$$f = \frac{(\sum_{i=1}^{m} x_i)^2}{m \sum_{i=1}^{m} x_i^2}, \tag{4.6}$$

and the configuration is 100% fair when $f = 1$.

Figure 4.10 shows this index for both MMF and MNU algorithms when they are applied to networks with different users. We see that the MMF algorithm is fairer than the MNU algorithm, verifying that the design goal of the MMF algorithm is achieved.

Fairness among nodes It is also interesting to know how the nodes are utilized. Figure 4.11 shows the distribution of nodes according to the number of users impacting them. Three cases with 10, 50 and 90 users are demonstrated and a node may have $0, 1, \ldots, m$ impacted users if there are m users. Our first observation is that the difference between MMF algorithm and MNU algorithm becomes larger as the number of users increases although it is insignificant under the 10 user case. More interestingly, the distribution is now more concentrate for the MNU algorithm than that for the MMF algorithm, which is indicated by higher and narrower curves for the MNU than those for the MMF. Comparing with the distribution of users shown in Figure 4.9, the following conclusion could be made. When the MMF optimization objective is applied in favor of users, it gives fairness to each user but results in unfair resource consumption on the nodes. On the contrary, if the MNU objective is applied, it gives maximum overall network utility at the cost that the users may

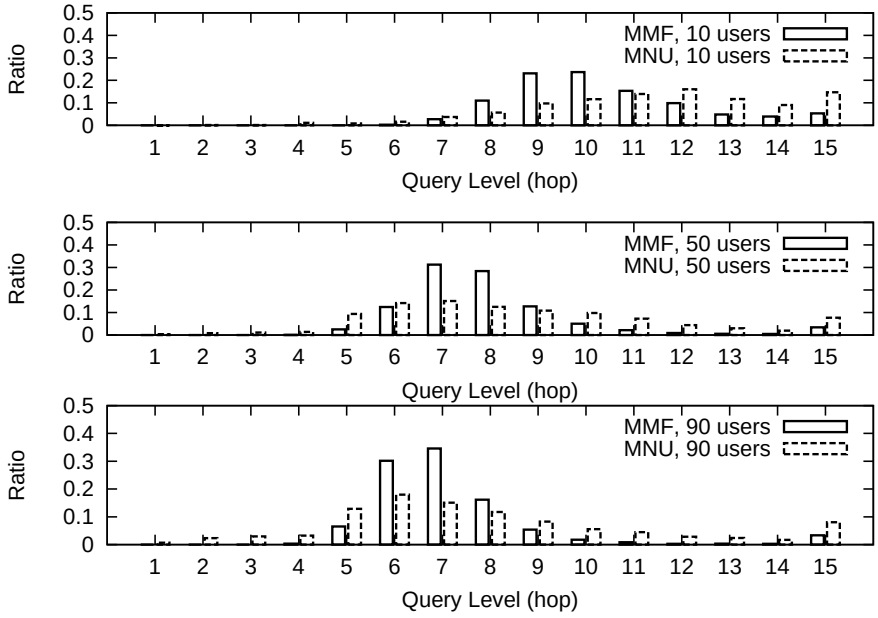


Figure 4.9: Distribution of users at each query range level.

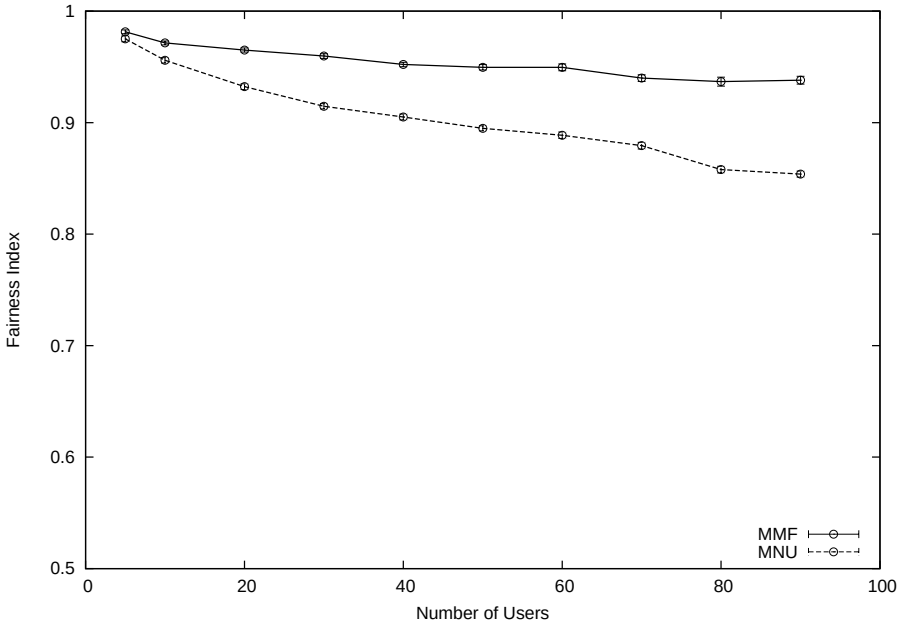


Figure 4.10: Fairness index.

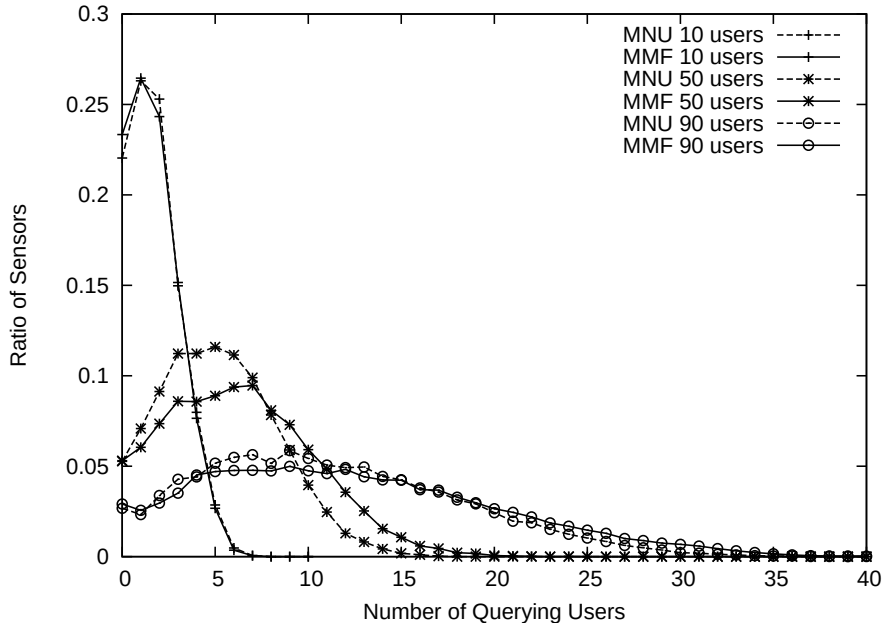


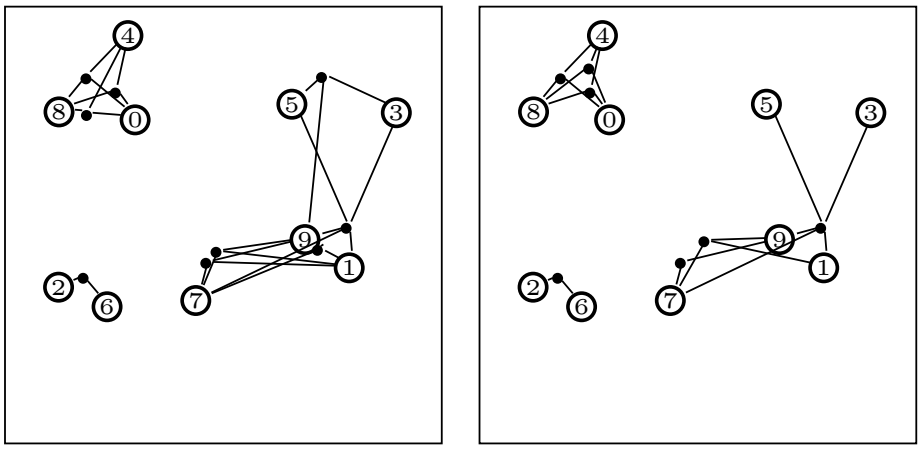
Figure 4.11: Node distribution on the number of impacting users.

be served unfairly, but the resulted configuration makes use of the resources on each node more evenly. Thus, the MNU may be favorable in particular network designs. For example, in some wireless sensor networks with multiple data collectors, the energy consumption on each node should be well balanced in order to maximize the network lifetime.

Another interesting observation on the Figure 4.11 is about the ratio of the nodes with no user impacting them. Over 20% of nodes are not impacted by any user when 10 users are in the network while this ratio decreases quickly to about 5% for 50 users and 2% for 90 users. This observation may be valuable when the network coverage is concerned. Suppose the users are the service providers as the access points in a mesh network while the nodes are the users of such mesh networks. One design concern is how to cover maximum number of the users within a certain region with minimum number of randomly deployed access points. However, Figure 4.11 shows us that if a certain coverage is required, the number of access points must achieve a certain threshold. It is impossible to cover the network only by enlarging the coverage of a few access points.

4.4.4 A dynamic network example

In this final simulation, we try to show how the users cooperate according to Algorithm 6 in order to achieve a certain global optimization objective, by a simple example. We use the scenario 5 in Table 4.2 and a network topology with 10 users is generated for it. The simulation runs for 5000 seconds, however, at the beginning of the simulation, only 5 users start working while the rest stay idle until the simulation time 1500s. The topologies after 1500s is shown in Figure 4.12(a) and 4.12(b),



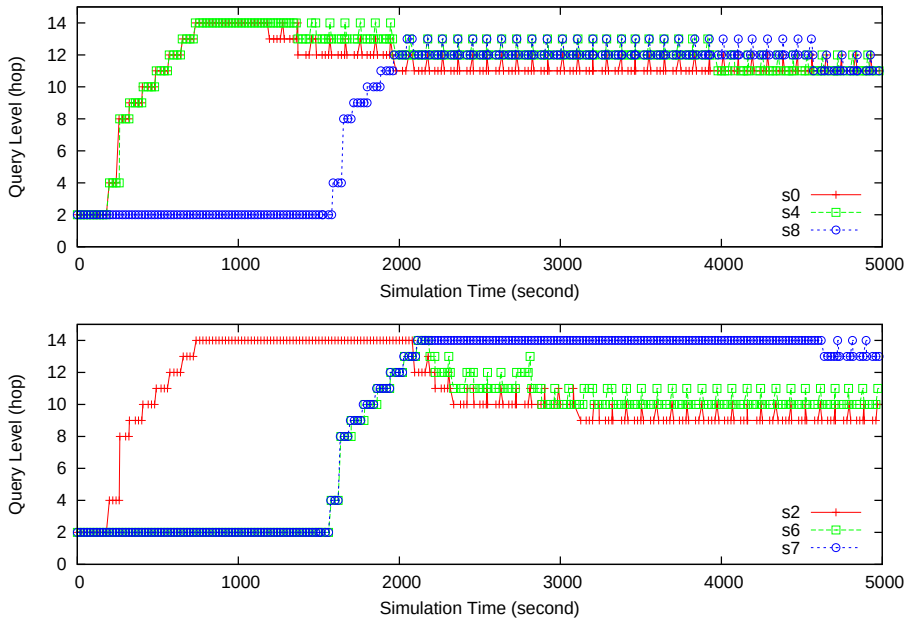
(a) MMF (b) MNU

Figure 4.12: Network topology with 10 users and the congested nodes.

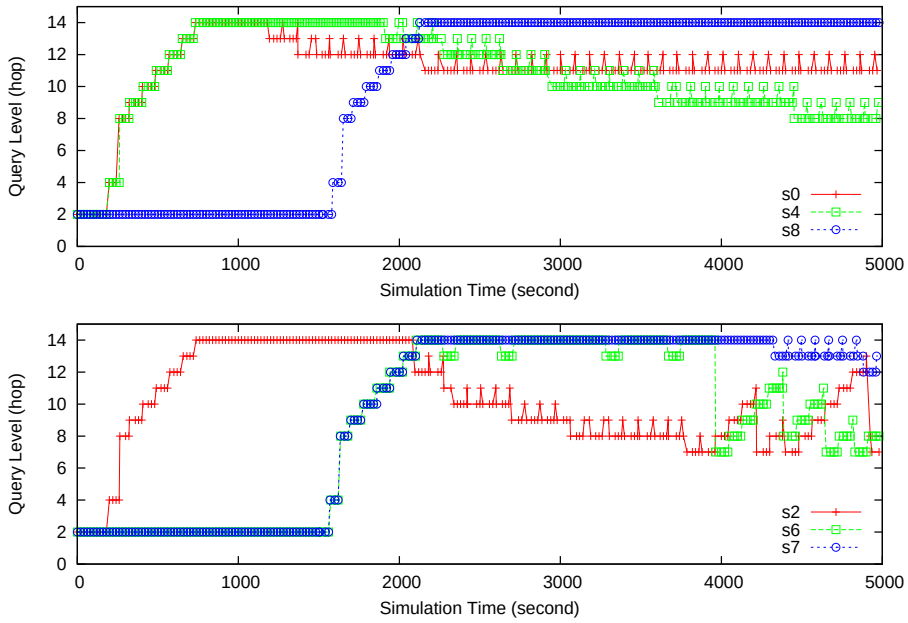
for the MMF and MNU cases, respectively. The topologies before 1500s are trivial thus not shown. The congested nodes and the users related with them are depicted while other nodes are omitted for clarity.

The users are connected by a link with the congested nodes if their query range levels are actually modified by the nodes. These users could be seen as the most critical impacting users to the corresponding nodes. Other users around a congested node may have impacts on it as well, but as their impacts could be so weak that decreasing them will not generate a better solution, their query ranges are not modified by the node. As user 7 to the congested node between user 2 and 6 in Figure 4.12(a). It has impact on the node but its impact level is not modified by the node. This phenomenon may be used in developing smarter distributed heuristics. Besides, the number of the congested nodes shown in the figures is trivial as compared to the total population of nodes which is set to 1000 in this example. Together with the limited number of users related with each congested node, one can expect a limited computation effort has to be spent on the nodes, as well as a modest number of messages have to be generated by those nodes.

Finally, we present Figure 4.13(a) and 4.13(b) to show how the query range levels of users evolve, under the MMF and MNU optimization objective, respectively. A common observation is whenever a congestion is detected, the algorithm needs some time to adjust the query range levels between users and then converges at an approximate optimal configuration. Long term evolution of the MMF and the MNU optimized users could be easily distinguished within each clique of users, *e.g.* $\{0, 4, 8\}$ and $\{2, 6\}$. In the MMF case, query range levels tend to converge to similar values while they diverge in the MNU case.



(a) MMF



(b) MNU

Figure 4.13: Query level evolution of users.

4.5 Summary

MMF and MNU optimization objectives have been studied in the chapter. Both of them conform with the MMKP formulation which takes the resources constraints on each node into consideration. In consequence, the resulted algorithms are able to prevent the nodes from being congested. Although we consider only bandwidth in this study, it is possible to generalize to other types of resources *e.g.* energy, storage, computation power, *etc.*

The average and maximal query range levels have been inspected and they provide precious indication which may help for dimensioning the wireless sensor networks. The comparative study on the bandwidth utilization of the two optimization objectives has shown that although the MNU maximizes the network utility, the average bandwidth consumption on the nodes are not necessarily maximized at the same time. On the contrary, under all of our simulation scenarios, the bandwidth consumption of the nodes in a network optimized for MNU is lower than in those optimized for MMF. Furthermore, the distribution of query range levels are more concentrate among users under the MMF objective than under the MNU objective, indicating the former gives more fairness to the users. More interestingly, as the number of users in the network grows, the distributions tend to concentrate first instead of shifting leftwards and this holds for both MNU and MMF objectives although it is more obvious for the latter case. This phenomenon implies that the network under local traffic model tries to give more users a medium query range when the number of users grows.

It will be interesting to cope with dynamic networks: transient users and moving nodes are the first challenges on the way to a realistic network. Finally, the distributed heuristic could be improved by investigating new ways to estimate the traffic in wireless sensor networks.

Chapter 5

Extension To A Practical Context: ZigBee Based WSNs

"He, who every morning plans the transactions of the day, and follows that plan, carries a thread that will guide him through a labyrinth of the most busy life."

– *Victor Hugo*

Query allocation problems have been formulated and studied with both continuous and discrete query models for multi-user WSN in previous chapters. In this chapter, we adapt the algorithms for the discrete version of the problem, namely the MMKP-MNU and MMKP-MMF, to a ZigBee cluster tree based wireless sensor network. The rest of the chapter is organized as follows: How to build mobile user WSNs based on the IEEE 802.15.4 and ZigBee standards is discussed in Section 5.1. The distributed algorithm (Algorithm 6) proposed in Section 4.3.2.3 is then adapted to the ZigBee cluster tree structure, with special considerations on the addressing and routing mechanisms which make the resulted algorithms more efficient with respect to the amount of control messages involved. The adapted algorithms are evaluated via simulation in Section 5.3 and we conclude this chapter in Section 5.4.

5.1 The IEEE 802.15.4 and ZigBee Tree

A general description on the IEEE 802.15.4 standard [80206] has been given in Section 2.1.2.2. Specifically designed for low power devices, it allows devices to sleep during the inactive period in a superframe bounded by the beacons. The sleep mode, which is only available when the network works on a synchronized tree structure, is appealing when energy saving is desired. On the contrary, if the network works on a peer-to-peer mode, the lack of synchronization makes the devices keep alive in order to receive from neighbors.

Based on the 802.15.4, the ZigBee specification [zig08] defines addressing, network maintenance and routing for unicast, multicast or broadcast packets at network layer. We have discussed the basic functionalities provided by the ZigBee specification in Section 2.1.2.3. Especially, the cluster tree routing does not employ any routing discover packets thus it is more appealing for energy constrained networks such as WSNs.

Building multi-user WSN based on ZigBee network is straightforward. However, some choices have to be made to fit the special requirements of multi-user WSN. In a WSN, the routers are usually at the same time the sensor nodes, thus are supposed to be battery powered. This energy issue advocates for the tree operation mode proposed by ZigBee. Network maintenance when multiple users join and leave can be easily implemented in the ZigBee infrastructure with full conformance. For example, a user may use the ZigBee join procedure to connect to the network and follow the leave procedure when it moves outside the parent and later reconnect to other nodes with the rejoin procedure.

Furthermore, a user may act as either end devices or routers in the ZigBee network. If mobility is considered, the movement of the user router may result in reconstruction of the subtree rooted at itself. In this study, we confine the users to act only as end devices, thus the backbone of the cluster tree can be kept intact even when the users move.

A user initiates a query in order to collect data from the sensors. Motivated by the application scenarios introduced previously, we suppose the query is sent to all sensor nodes within a certain number of hops to the user then all such sensors will send data back to the user at the required rate repeatedly until modified by another query. The queries are sent via hop-bounded flooding on the tree and data is sent back via unicast cluster tree routing, both supported by the ZigBee specification. Specifically, the former can be done by checking the query message header for the *radius* field on the router, and rebroadcasting the message if radius is positive, to its on-tree neighbors except the one from which the message is received. For the latter, we assume sensors send a copy of the same data to each querying user rather than using multicast. Figure 5.1 illustrates an example multi-user WSN with two users and a query at radius of 3 hops.

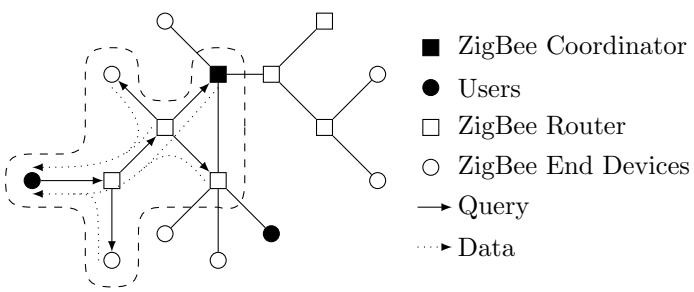


Figure 5.1: Multi-user WSN based on ZigBee tree structure.

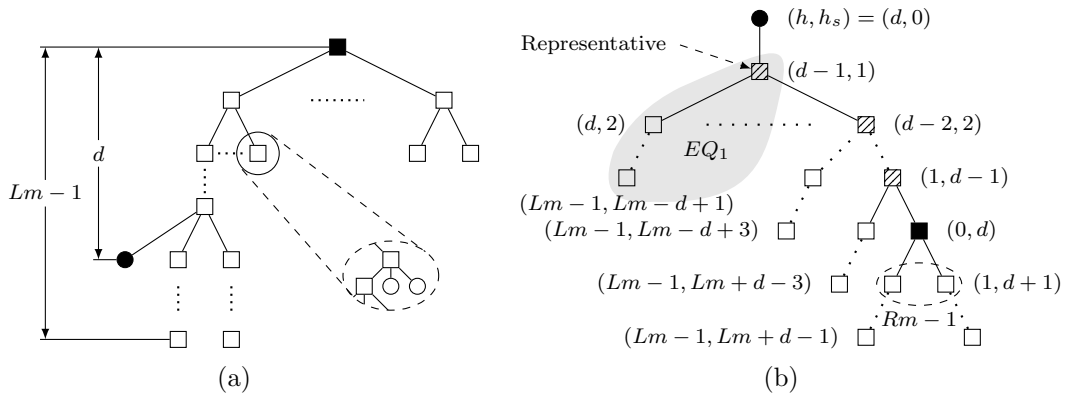


Figure 5.2: Two views of a ZigBee routing tree. (a) Tree T, rooted at the coordinator, (b) Tree T_s, rooted at the user, labels assigned to devices.

5.2 Algorithms Adapted to the ZigBee Network

The distributed algorithms (the Algorithm 6) for both MMKP-MNU and MMKP-MMF have been proposed in Chapter 4. The basic idea is to solve a MCKP at each congested sensor and then feed the results back to the related querying users. The adaptation of these algorithms to the ZigBee tree network consists of only a new traffic estimation mechanism that exploits the special property offered by the cluster tree addressing and routing mechanisms.

5.2.1 Traffic estimation with the ZigBee tree

When the sensor notices a congestion is about to happen, the traffic estimation is employed to calculate optimal query allocation for the users querying it. Traffic could be estimated through various ways and here we propose a local estimation profiting the ZigBee address structure. The aim is to give an estimation about the increment of the number of devices involved if the query range becomes one hop larger. Intuitively, the additional traffic load comes from those devices.

Consider the ZigBee network which is a tree T rooted at the ZigBee coordinator, as shown in Figure 5.2(a). For the querying user s, the whole network could be seen as a tree T_s rooted at itself. Thus for a router, the additional traffic passing through comes from the to-be-covered devices that are his descendent on T_s. A logical structure of T_s is illustrated in Figure 5.2(b). We assign a label (h, h_s) to each device in the network, where h and h_s denote the depth of the device on T and T_s, respectively. Note that every device knows h on joining the network. On the other hand, for a given device, h_s is actually its hop distance to s, which could be obtained simply by inspecting the received query message originated from s.

Then we divide the routers (including the ZigBee Coordinator) into several equivalent classes according to their labels. Each equivalent class consists of a sub-tree rooted at a router on the path from the user to the coordinator. Obviously, there

$$\Delta_R^n(i,h_s,k) = \begin{cases} Rm^{k-h_s+1}, & \text{if } h_s - 1 \le k \le Lm - d + 2i - 2 \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

$$\Delta_R^r(1,1,k) = \begin{cases} Rm\Delta_R^n(1,2,k) + \Delta_R^r(2,2,k), & \text{if } 0 < k \le Lm + d - 2 \\ 1, & \text{if } k = 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

$$\Delta_R^r(i,i,k) = \begin{cases} (Rm-1)\Delta_R^n(i,i+1,k) + \Delta_R^r(i+1,i+1,k), & \text{if } i - 1 < k \le Lm - d + 2i - 2 \\ 1, & \text{if } k = i - 1 \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

are d such equivalent classes. Let EQ_i be the ith equivalent class and we have:

$$EQ_i = \{r | h - h_s = d - 2i\}, (1 \le i \le d). \tag{5.1}$$

The router with $h_s = i$ is referred to as the *representative router* of the equivalent class, noted as r_i. For example the routers with label $(d-1,1),(d,2),\ldots,(Lm-1,Lm-d+1)$ belongs to EQ_1 and $(d-1,1)$ is the representative router. As shown in Figure 5.2(b), the routers of EQ_1 are grouped into a shadowed area.

Now consider a router in $r \in EQ_i$ with label (h,h_s), let $\Delta_R(i,h_s,k)$ be the number of additional routers and $\Delta_E(i,h_s,k)$ the number of end devices that will be handled by r when the query range increases from k to $k+1$. If every device send data at constant rate b, the additional traffic load $\Delta_T(i,h_s,k)$ should be:

$$\Delta_T(i,h_s,k) = b\left(\Delta_R(i,h_s,k) + \Delta_E(i,h_s,k)\right). \tag{5.2}$$

Now we derive $\Delta_R(i,h_s,k)$, then $\Delta_E(i,h_s,k)$ could be obtained as the following:

$$\Delta_E(i,h_s,k) = \Delta_R(i,h_s-1,k-1)(Cm - Rm). \tag{5.3}$$

In order to calculate $\Delta_R(i,h_s,k)$, we further separate the non-representative routers and the representative routers and denote the corresponding Δ_R function as Δ_R^n and Δ_R^r.

Then we have (5.4) for a non-representative router, where $1 \le i \le d$ and $i+1 \le h_s \le Lm - d + 2i - 1$. For a representative router, which implies $h_s = i$, we have (5.5) and (5.6) for $i = 1$ and $i > 1$ cases, respectively.

One remark about the estimation proposed above is that it obtains an upper bound of the additional traffic and this upper bound is reached only when the network address is fully used by the devices. However, this condition is hardly met in practice, thus the estimation usually deviates from reality. As a countermeasure, we propose to apply a scaling parameter to the estimated traffic to account for address utilization. For example, routers may obtain such a scaling parameter from the number of addresses allocated to its children and its available address block.

The new traffic estimation method could be applied directly by each node k to calculate $M(k)$. As a result, Algorithm 5 and 6 could be used with minor modifications.

Table 5.1: Evaluation metrics.

Parameter		Definition
T_{app}	=	Query data receive bit rate at users
γ	=	$\frac{T_{app}}{\text{Query data sent bit rate}}$
O_{mac}	=	$\frac{\text{MAC layer control message sent bit rate}}{T_{app}}$
O_{app}	=	$\frac{\text{Application layer control message sent bit rate}}{T_{app}}$
I	=	$\frac{\text{sum of query level of users}}{\text{number of users}}$, at a certain simulation time.
I^*		same as I, obtained by an exact algorithm.
f	=	$\frac{\left(\sum_{i=1}^{m} x_i\right)^2}{m \sum_{i=1}^{m} x_i^2}$, m: number of queries, x_i: query radius of user i.

5.3 Performance Evaluation

In this section, we evaluate the proposed algorithms by comprehensive simulations. We implemented the proposed algorithms and the basic functionalities of the ZigBee network layer on top of IEEE 802.15.4 implementation [ZL] in the ns2 simulator [ns2].

5.3.1 Evaluation metrics

We will evaluate the following metrics: query data throughput (T_{app}), query data arrival ratio (γ), MAC layer control message overhead (O_{mac}), application layer control message overhead (O_{app}), query radius and fairness index (f). The definitions are presented in Table 5.1.

5.3.2 Simulation setup

Table 5.2 summarizes the simulation parameters. We evaluate our algorithm in two kinds of networks: a grid network and a network with uniformly deployed nodes. The size of the network are chosen to cover a small network and a large network, with 50 nodes in 100 by 100 meters square, 100 nodes in 140 by 140 meters square, respectively.

We also select two representative values of the queried data rate: for a light load network, each node on query send data at 100bps, which corresponds to sending a 28 bytes of data every 2.24 seconds; for a heavy load network, the query data rate is 800bps or 28 bytes of data every 0.42 seconds. We use this fixed data message

Table 5.2: Simulation parameters.

Topology	1: 50 nodes at 100m×100m, 2 users 2: 100 nodes at 140m×140m, 4 users
Node distribution	Grid with 14m interval / Uniform
MAC layer	IEEE 802.15.4, $BO = SO = 6$, $Lm = 10$, $Cm = 3$, $Rm = 3$
MWK layer	ZigBee cluster tree routing
Wireless Tx/Rx	15m transmission range, two ray ground, omniscient antenna model
Bandwidth	250kbps at 2.4GHz band
Query data rate	Light load: 100bps, High load: 800bps
Simulation time	Query start: 70s, query stop 300s, simulation stop: 350s

length in our simulation, following the default data length defined in TinyOS [Lev] for CC2420 transceiver.

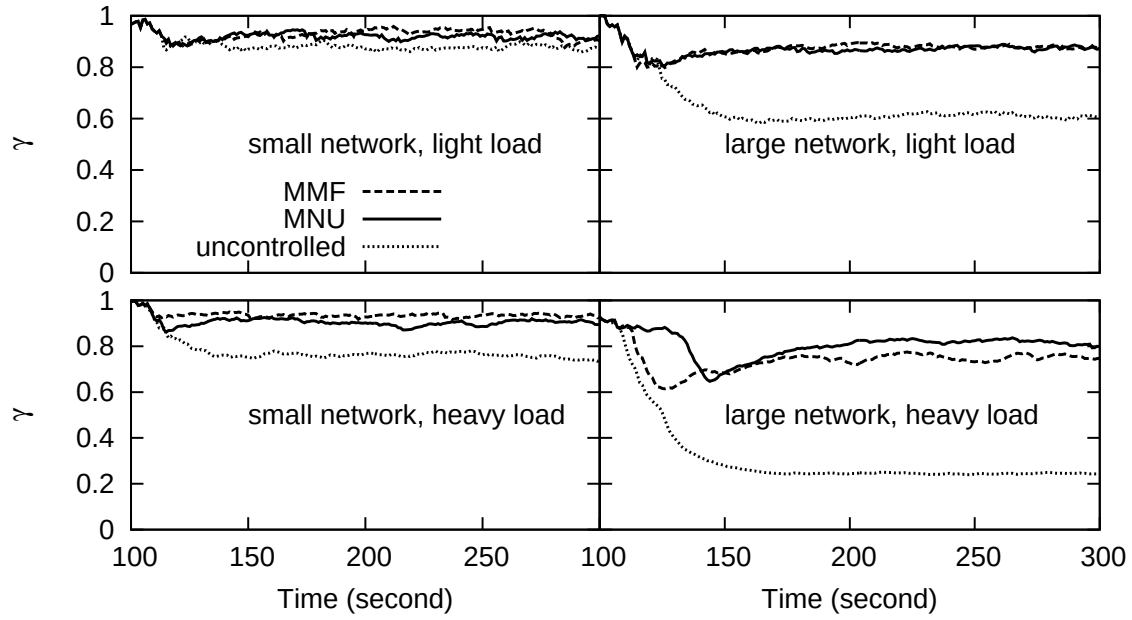
We start the nodes at the beginning of the simulation with random jitters and the nodes will continue trying to join the network until it succeeds, except node 0 who will create the ZigBee network. The network formation takes about 60 seconds thus we start the queries at 70 seconds. To let the simulation shut down gracefully, we stop all the queries 50 seconds before the simulation stops.

5.3.3 Query data arrival ratio

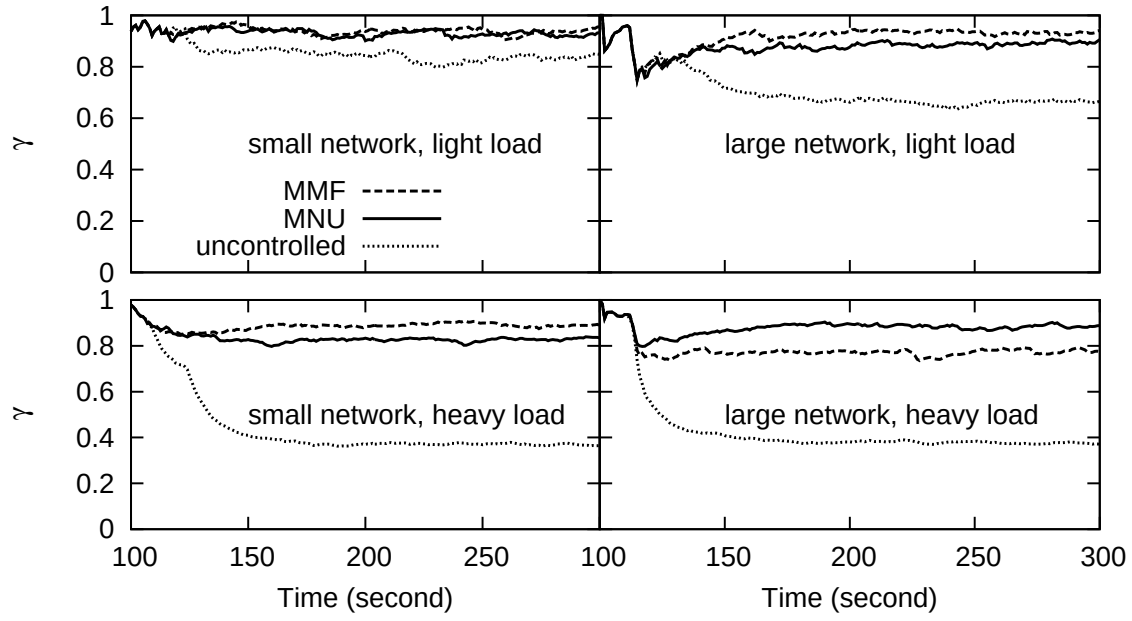
Let us consider how well multiple queries are handled in the network controlled by the distributed algorithms, by inspecting the query data arrival ratio. As comparison, we also run the network without control algorithm. The results are presented in Figure 5.3. We see that the uncontrolled network delivers the queried data at a relatively low ratio than the controlled network. The arrival ratio dropped very quickly after the queries are diffused in the network. For a heavy load network ($b = 800$), less of congestion control is generally detrimental: the arrival ratio could be as low as 0.3 for large grid network and around 0.4 for both large and small uniform networks. This is unfavorable. On the contrary, a query radius control with either MNU or MMF objective could prevent the network from congestion, thus resulted in higher data arrival ratio. In a small network with light query load, the query control did not gave noticeable improvement on the arrival ratio compared with uncontrolled case. Actually, there is almost no congestion in the network under this case and the effect of the query control is not obvious.

5.3.4 Query data throughput

The aggregated throughput of the query data provides more information about the performance of the network. Figure 5.4 presents this result. Intuitively, the



(a) Grid



(b) Uniform

Figure 5.3: Application data arrival ratio.

control algorithm will confine the query radii from going too large, so will limit the total number of sensor nodes under query. This may result in a lower throughput. However, if the network is congested, the throughput may be even lower than an uncongested network with smaller queries. This phenomenon can be observed in the large grid network under heavy load case (bottom right plot of Figure 5.4(a)).

5.3.5 Control message overhead

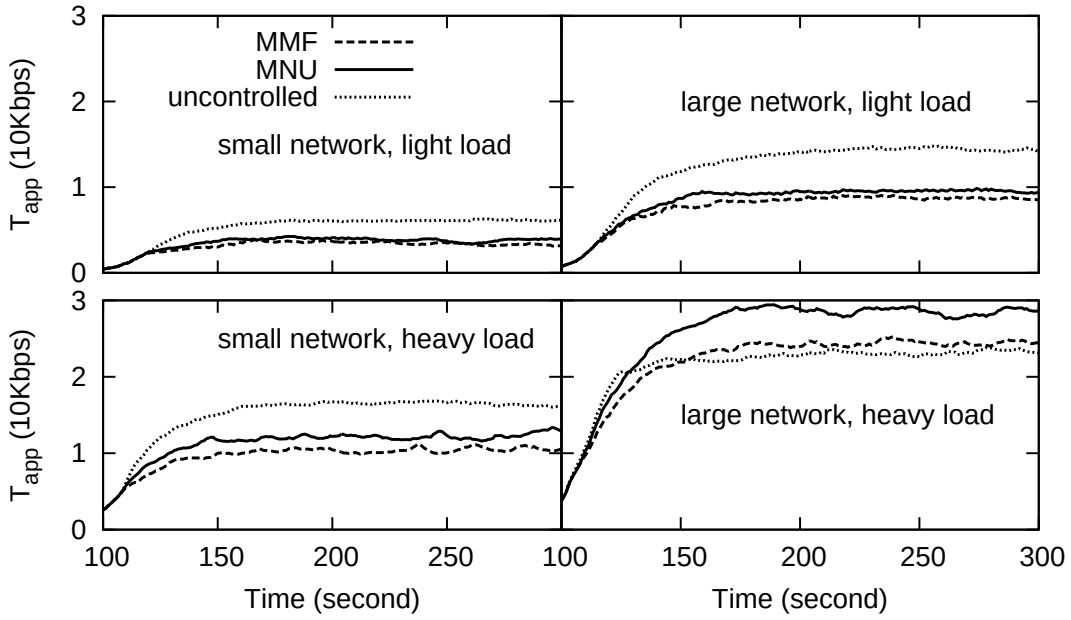
We are interested in how much it costs to achieve query control in the proposed query scheme and the distributed algorithm. We shall investigate this issue by inspecting both the MAC layer and the application layer control overhead. Note that the network layer does not impose any overhead after the network formation thanks for the cluster tree routing. Even during the network formation period, the network layer requires only several message exchange for one device to join the network. This overhead is negligible comparing with those in the normal operation period.

In MAC layer, the overhead comes from the beacon message, the acknowledgment and the ARP packets, while in application layer, it comes from the query message and the adjust-level message as described in Algorithm 6. Figure 5.5 shows the overhead for both layers in different network settings. The values could be interpreted as how many bits of control message is needed for successfully delivering one bit of query data. At MAC layer, we observe a sharp decrement of the overhead when the query load goes high, for both grid and uniform networks.

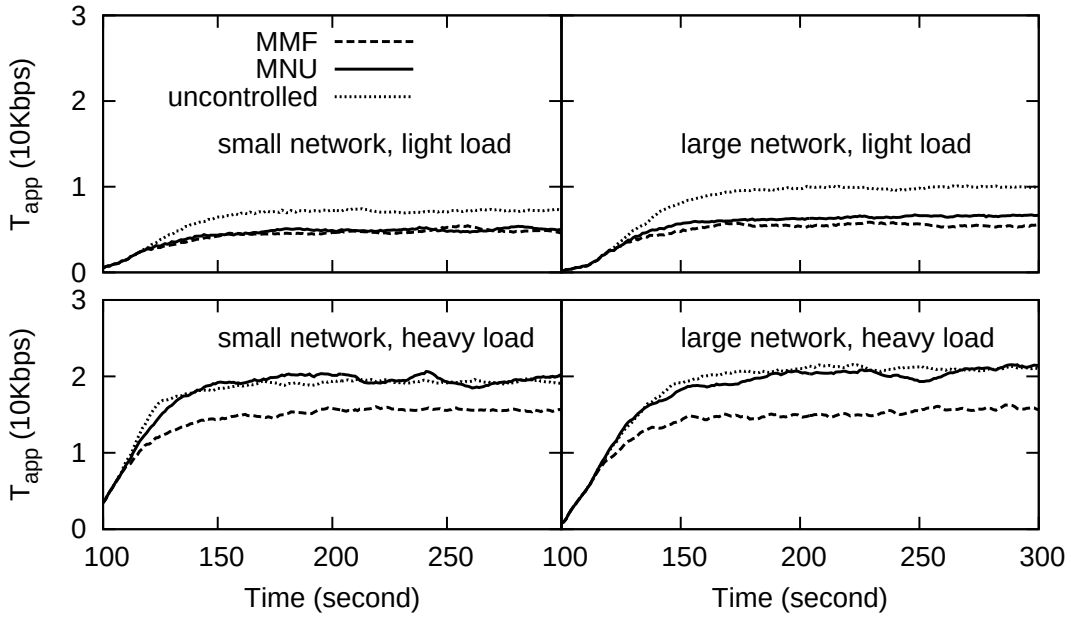
In application layer, the uncontrolled networks always have negligible overhead, independent to the network size, the node distribution or the query load. In contrast, the control overhead goes abruptly for networks with either MNU or MMF when the query load is light. However, under heavy load case, the overhead could be kept almost constant when the network size increases. Combined with Figure 5.3 and 5.4 previously discussed, the query control mechanisms are more efficient under heavy load case than under light load case.

5.3.6 Query range and fairness index

The average query range of the 4 users in the large network, obtained by both the distributed algorithms and an external problem solver is plotted in Figure 5.6. The external problem solver obtains an optimal solution for both MNU and MMF with the traffic information traced from the uncontrolled network. We see that for MMF case, the approximation is quite near to the optimal. While for MNU in a light load network, the optimal is further above what the distributed algorithm can achieve. Furthermore, we can observe that the difference between optimal and simulation values is larger under light load cases than heavy load cases, under MNU cases than MMF cases. The explanation is as follows. Normally, the 'select the minimum' strategy used in Algorithm 6 gives a user more chances to follow a smaller query range, especially when the users have more possibility to choose different query range values. This is actually a cavity of the 'select the minimum' strategy. By inspecting more carefully on the local solutions found on the congested sensors, we

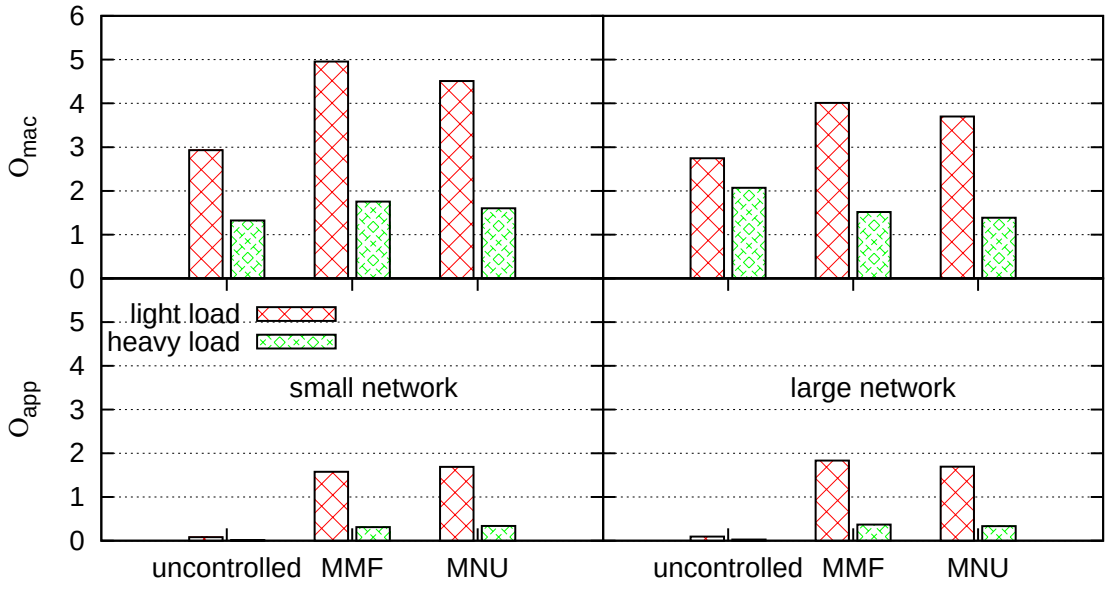


(a) Grid

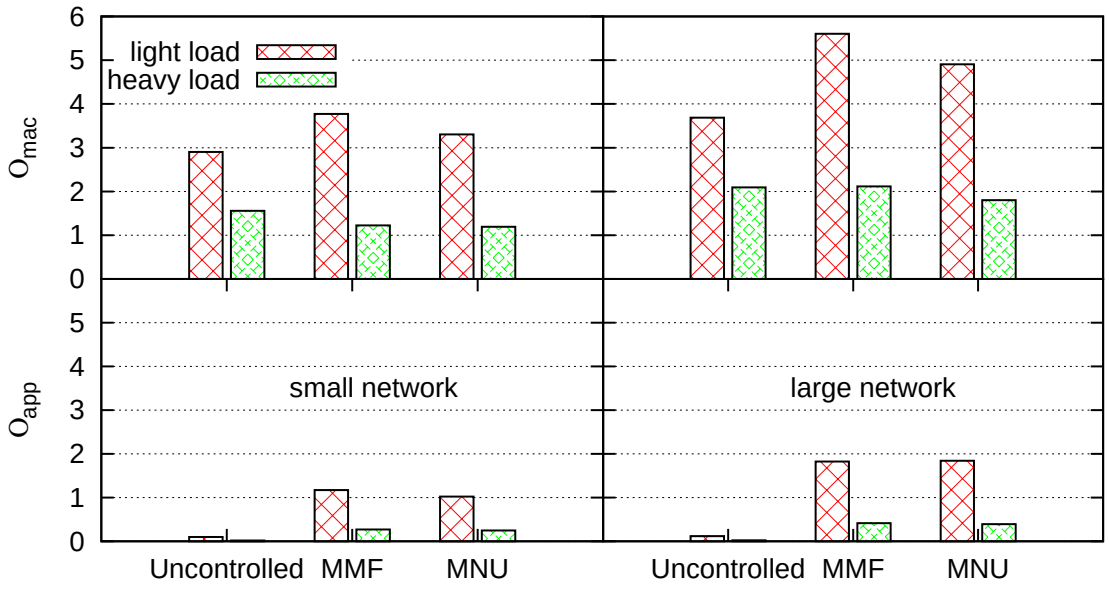


(b) Uniform

Figure 5.4: Aggregated application data throughput.

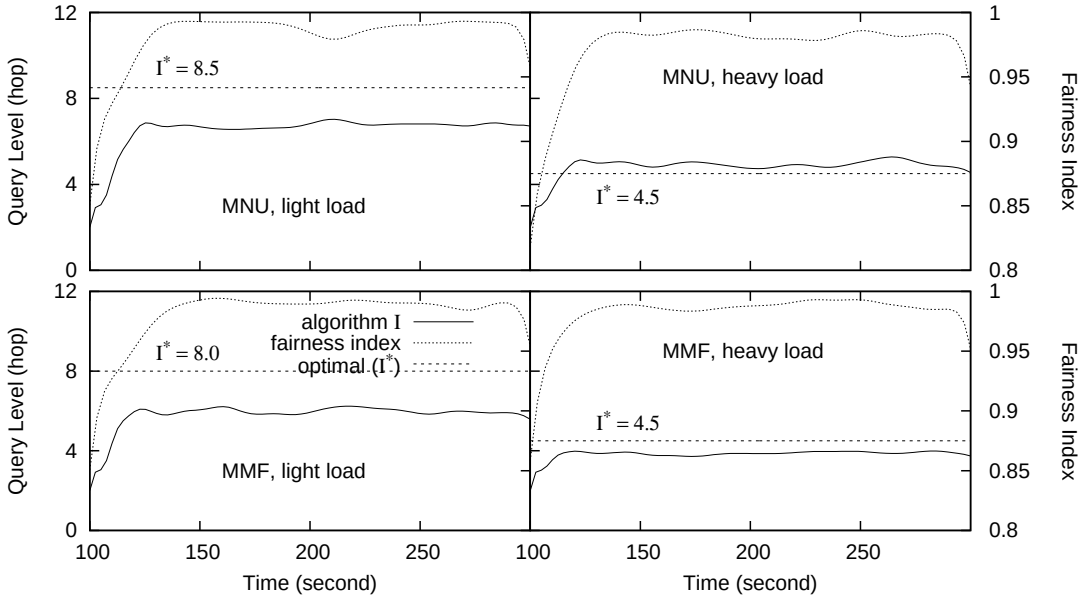


(a) Grid

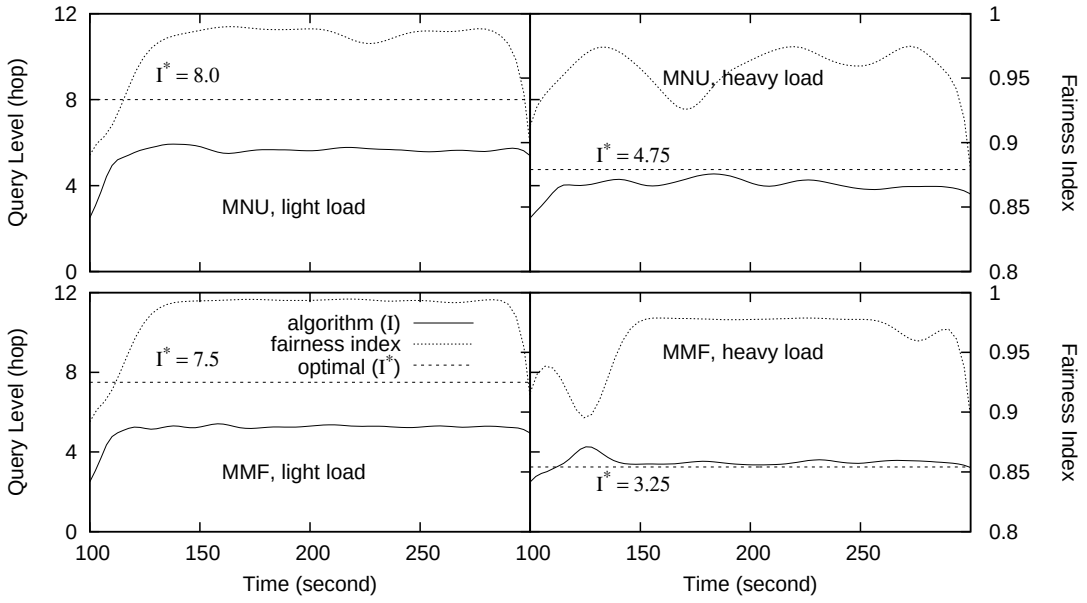


(b) Uniform

Figure 5.5: Protocol overhead.



(a) Grid



(b) Uniform

Figure 5.6: Query radius evolution of users.

found that the obtained query ranges are quite dispersed sometimes. However the average is near to the optimal value, *e.g.* at certain time in a simulation run, the query range calculated on two sensors could be $(6, 5, 10)$ and $(5, 10, 6)$ for 3 users, both resulting an average of 7. However, this diversity is eliminated at the users by always choosing the minimal, *e.g.* users have to choose $5, 5, 6$ as their final results. This problem becomes less obvious when the local solution gives users similar query ranges. Finally, one may wonder why the approximation over-performs the optimal in the MMF heavy load case. This is actually due to the linear increment used to further explore the potential larger feasible query range. If the local solution suggested by the congested sensors is already very near the optimal, the exploration procedure may generate transient congestion state in the network. We leave these problems for future works.

Finally, fairness index of the 4 users in the network is plotted for MNU and MMF. Since MMF generally gives similar query range to the users, its f values are higher than those of MNU under the same network situation. This could be examined by comparing the f plots in Figure 5.6 vertically, justifying the optimization objective of MMKP-MMF.

5.4 Summary

Mobile user network structure hence the resulted direct interaction between users and sensors can be implemented on IEEE 802.15.4 and ZigBee standards in this chapter. Algorithms for MMF and MNU, proposed in Chapter 4, have been adapted to the ZigBee cluster tree in this chapter. By exploiting the ZigBee cluster tree structure, the computation was done fully locally. Simulation results have shown that: firstly, the new algorithms do well in congestion control with little overhead and they are especially efficient in large networks with heavy queries; secondly, MMKP-MMF algorithm works better than MMKP-MNU in approximating their optimal solutions on the ZigBee cluster tree.

Chapter 6

The Problem Behind: Empirical Study On The Hard MMKP Instances

"Nothing in life is to be feared. It is only to be understood."
– Marie Curie

We have formulated the discrete version of the query allocation problem for the mobile-user wireless sensor networks with the multidimensional multiple choice knapsack problem in Chapter 4. An overview on the KP and MMKP problems has been given in Section 2.3. Exact and heuristic algorithms for MMKP proposed in previous works have been discussed in Section 2.3.1 and Section 2.3.2, respectively.

In this chapter, we focus on this hard combinatorial problem itself. We develop our idea in this chapter with the following sections: A brief introduction is given in Section 6.1, mainly to motivate the empirical study presented in this chapter. Then we give a brief survey of existing benchmarking methods in Section 6.2. Then in Section 6.3, we propose new methods to generate MMKP benchmark instances. Section 6.4 is dedicated to evaluating the difficulty of these instances using the existing exact algorithm and solvers. Finally, we draw conclusion and propose important future works in Section 6.5.

6.1 Introduction

Multidimensional Multiple choice Knapsack Problem (MMKP) is one of the most complex members of the Knapsack Problem (KP) family. It could be stated as follows: We are given m classes with each class i containing n_i items. The jth item of class i has profit p_{ij}. Each item has l dimensions of weight, and the weight at dimension k is denoted as w_{ijk}. The knapsack has capacity c_k on each dimension k. The goal is to select one item in each class to maximize the sum of their profits and

to keep the total weight on each dimension no more than the corresponding capacity. Formally, MMKP could be expressed with an integer programming model:

$$\text{(MMKP)} \quad \text{maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} p_{ij}x_{ij} \tag{6.1}$$

$$\text{subject to} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{ijk}x_{ij} \leq c_k, k = 1, \ldots, l \tag{6.2}$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, i = 1, \ldots, m \tag{6.3}$$

$$x_{ij} \in \{0, 1\}, i = 1, \ldots, m, j = 1, \ldots, n_i \tag{6.4}$$

where the binary variable x_{ij} indicates the jth item of class i is selected or not. For clarity, we assume all classes have the same number of items in this study, *i.e.* $n_1 = \cdots = n_m = n$.

MMKP has many applications. It has been used to model the Quality of Service (QoS) management problem in computer networks [LLRS99] and the admission control problem in the adaptive multimedia systems [SIH05, Kha98, KLMA02]. Various other resource allocation problems can also be mapped directly to MMKP, please refer to [KPP04, PHD05] and the references wherein.

Most academic efforts related with MMKP have been put on finding heuristic algorithms due to the NP-hard nature of the problem [KLMA02, MJS97, HMS06, ARK+06]. However, computing exact MMKP solutions can also be of interest when the computation time constraint is not critical, *e.g.* exact solution is the most valuable benchmark for the heuristic algorithms. More important, when the exact algorithms can solve most problem instances quickly and only fall into exponential time in rare cases, they can be applied to some practical problems. Thus, for any exact algorithm, it is important to know how often it falls into the trap.

Great efforts have been taken in analyzing the structure of many KP family members, *e.g.* simple KP, Bounded KP (BKP), Multiple KP (MKP), Multiple Choice KP (MCKP), Multidimensional KP (MDKP) *etc.* Comprehensive discussions on these problems could be found in [HMS75, KPP04]. It has been demonstrated that all of them are highly structured. Exploiting their special structural properties usually leads to efficient algorithms that are able to solve certain category of problem instances in reasonable time, although the problems are NP-hard. Moreover, it appears that some instances are particularly hard to solve not because of the size of the problem (number of input variables or the magnitude of the variable values), but because of the special combination of variable values, *i.e.* even a small problem instance can require a long time to solve. Generally, the relationship between the profits and weights of the items plays a very important role in the solution time of certain problem instances. Besides, a very important observation has been made on simple KP that the relationship between the capacity and the weight also has great impact on the hardness of the instances [Pis99].

In contrast, little work has been done on MMKP in analyzing its structure. To the best of our knowledge, besides many proposed algorithms, the only theoretical analysis is presented in [DW98]. The authors show that the proportion of the dominated variables can be estimated as a probability function of the number of dimensions and the number of items in each class. The results could be used to reduce the problem size during the pre-process stage. However, it is still unclear how the parameters such as profit, weight and capacity interact in making the problem difficult. Furthermore, almost all MMKP algorithms proposed in the literature are evaluated against very limited problem instances which may belong to special easy cases.

In this chapter, we study the relationship between various parameters of the MMKP such as profit, weight and capacity in order to identify the key factors that make a hard instance. Furthermore, uncorrelated, weakly correlated and strongly correlated cases for items within each class, between classes and across multiple dimensions are investigated. To the best of our knowledge, no such work has been reported in the current literature. A systematic method to generate comprehensive MMKP test instances is proposed. Several groups of instances generated with the proposed methods are evaluated with the exact algorithm and solvers. The experiments show that many instances are several orders of magnitude harder than traditionally used ones in terms of the computing time. These hard MMKP instances usually have medium knapsack capacity and high correlation between weights and profits. The experiments also suggest that instances with similar set of profits across classes and with strong correlations between weights and profits are hard to solve.

Two exact algorithms are currently available for solving the MMKP, namely the BBLP algorithm [Kha98, KLMA02] and the EMKP algorithm [Sbi07], which have been discussed in Section 2.3.1. Besides, two integer programming solvers, ILOG CPLEX [cpl] and GLPK [glp] exist. However, we have identified some fundamental problems of the EMKP algorithm [Sbi07]. First of all, the EMKP algorithm employs basically a sequential search strategy, *i.e.* the development of a certain node depends on its previous brother nodes. This prevents pruning the infeasible nodes effectively as some feasible nodes may have to be developed from them. Secondly, the EMKP algorithm selects the *best* node to develop at each round, but the paper does not state how the selection is made. A common way is to select the node with the highest upper bound. However, for the infeasible nodes which are kept in the search tree, there is no natural way to calculate an upper bound. Finally, the sequential search strategy implicitly requires generating the brother node for every node who has a brother. This is true even for the unpromising nodes whose upper bound is below the lower bound. However, EMKP tries to reduce the search space by pruning these unpromising nodes (line 17 of the algorithm in [Sbi07]). This will break the searching process and the algorithm will not be able to find the optimal solution which have to be reached through the pruned node. Even worse, the better the quality of the lower bound is (approaching the optimal from below), the more often the improper prune may happen. As a result, we will consider only the BBLP algorithm and two generic MIP solvers GLPK and CPLEX in this study.

6.2 Existing Methods to Generate Benchmark Instances

It is very important to test the algorithms with problem instances in order to know their performance in practice. When algorithms are tackling a particular problem, ideal instances for performance evaluation are those from traces of the real world. However, as MMKPs usually originate from a diverse applicative background, typical instances from a certain domain may hardly be reasonable for another. Moreover, there is no systematic report on real world MMKP problem instances in the literature. In contrast, test instances can be generated to cover a much wider range of instance types. As a result, generated instances play an important role in benchmarking the algorithms and have been used in most KP and MMKP related researches. In the following, we first describe how the KP instances are usually generated, then we give a brief review on the current method to generate the MMKP instances. The latter is basically a straightforward extension of the former.

6.2.1 Generating KP instances

In order to generate a KP instance with certain number of items, the idea is to first assign values to both profit and weight of each item, then to set the capacity of the knapsack. Several groups of instances have been identified for KP considering the correlation between the profits and weights [Pis05].

- *uncorrelated instances.* For this category, the profit of an item is independent to its weight. A commonly used method is to select profits (p_j) and weights (w_j) randomly in a certain interval, *e.g.* $[1, R]$. These instances are generally easy to solve.

- *weakly correlated instances.* In weakly correlated instances, the profit of an item is related with its weight, *e.g.* to select w_j randomly in $[1, R]$ and p_j in $[w_j - R/10, w_j + R/10]$ while ensuring $p_j \geq 1$.

- *strongly correlated instances.* For these instances, the profit of an item is a linear function of its weight plus a certain shift, *e.g.* to select w_j randomly in $[1, R]$, but let $p_j = w_j + R/10$. This category of KP instances are generally hard to solve.

- *sub-set sum instances.* For this category of instances, the profit of an item is a linear function of its weight. As a result, only weight need to be considered when packing the knapsack.

Instances with other types of correlation can be defined similarly [Pis05].

Finally, the capacity of the knapsack is set to a certain percentage of the sum weight. However, this has been shown to be inadequate as it generates the easiest KP instances under certain situations [Pis99]. Thus the idea consists of generating

a series of S test instances with the capacity c of the hth instance selected as:

$$c = \frac{h}{S+1} \sum_{j=1}^{n} w_j. \tag{6.5}$$

6.2.2 Generating MMKP instances

To generate an MMKP instance with a given number of classes, a given number of items in each class and a given number of dimensions, the problem is to assign a profit value to each item, a weight value to each item at each dimension, and finally, a capacity value to each dimension of the knapsack. This can be done in various ways.

In [KLMA02], weights of items are uniformly selected in the interval $[0, R]$ indifferent to the dimensions or which class the item belongs to, where R denotes the maximum resource consumption by an item. Let P denote the maximum cost of unit resource, then the value $R \times P$ could be considered as the maximum cost (weight) of an item. Then uncorrelated instances are generated with profits assigned according to the item index in the class:

$$p_{ij} = \mathcal{U}\left(0, l \times \frac{R}{2} \times \frac{P}{2}\right) \times \frac{j+1}{n_i}. \tag{6.6}$$

For correlated instances, the profit is a linear function of the sum weight:

$$p_{ij} = \sum_{k=1}^{l} P_k w_{ijk} \times \mathcal{U}\left(0, l \times 3 \times \frac{R}{10} \times \frac{P}{10}\right), \tag{6.7}$$

where $P_k = \mathcal{U}(0, P)$ is a uniform random number between 0 and P.

Finally, the capacity of dimension k is set to half of the maximum possible resource consumption:

$$c_k = \frac{1}{2} \times m \times R. \tag{6.8}$$

The same set of instances have been used in [Kha98, PHD05, ARK+06, CA06]. Instances generated with the same principle have been used in [HMS04, HMS06, CH08]. These instances are available at the *OR Benchmark Library* [OR-].

In [Sbi07], test instances are generated as follows: p_{ij} and w_{ij} are randomly selected in intervals $[0, 150]$ and $[0, 50]$, respectively. Capacities are set as:

$$c_k = \frac{1}{2} \sum_{i=1}^{m} \left(w_{ik}^{\min} + w_{ik}^{\max}\right), \tag{6.9}$$

where

$$w_{ik}^{\min} = \min_{1 \le j \le n} \{w_{ijk}\}, \tag{6.10}$$

$$w_{ik}^{\max} = \max_{1 \le j \le n} \{w_{ijk}\}. \tag{6.11}$$

In [AHHS05], domain related values are considered in the test instances. The number of classes, items in each class and dimensions are set according to a typical Video on Demand (VoD) system. The variable values are also set with respect to the typical values of a VoD system. For example, the weight of each item is set to the typical resource consumption of a session. The value is then scaled by a random number chosen from the interval $[0.75, 1.25]$ to mimic system dynamics. Similarly, the capacities are set to typical available resources scaled by a random value chosen from the interval $[0.95, 1.05]$.

Although these instances have been widely used in the literature, our computational results show that they are insufficient in demonstrating the performance of the algorithms. Table 6.1 presents the time used to solve the first few instances in the OR benchmark library with CPLEX, GLPK and the BBLP algorithm. Here we emphasize on the relative solution time across the instances. Notably, instances I3 and I4 take much more time than I5 and I6, despite they are smaller than the latter. This actually implies that not only the size of the instance, but also the structure play very important role in the solution time.

Table 6.1: Solution time (second) of OR benchmark library instances I1 to I6.

Inst	m	n	l	CPLEX	GLPK	BBLP
I1	5	5	5	0.005	0.028	0.016
I2	10	5	5	0.006	0.029	0.033
I3	15	10	10	1.983	16.036	67.260
I4	20	10	10	31.045	1383.251	1532.059
I5	25	10	10	0.018	0.046	0.660
I6	30	10	10	0.204	0.190	2.369

6.3 New Methods to Generate MMKP Problem Instances

Experiences from KP suggest that the correlation between profits and weights is critical to the hardness of an instance. Extending this idea to MMKP, we need to handle the correlation between the profits and multiple dimensions of weights. One direct way is to select the profit for each item then select the weights according to the profits. Given the number of classes (m), the number of items in each class (n, assuming all classes have the same number of items), and the number of dimensions (l), the MMKP is denoted as $P(m, n, l)$. We will also refer to a mapping from a class of n items to n values informally as a *generating function*.

6.3.1 Generating the Profits

In order to select the profits for items in each class i, we first bound the profits with two parameters $p_i^{\min}$ and $p_i^{\max}$ and select profit values within the interval. This could be done in various ways and here we define some generating functions for the profits.

Uniform Generating Function Uniform random profits are natural in many real world problems and are widely used in the literature. In uniform generating function, we draw profit uniformly and randomly within the interval. We denote the uniform generating function as:

$$p_{ij} = \mathcal{U}\left(p_i^{\min}, p_i^{\max}\right). \tag{6.12}$$

Linear Generating Function Items with linear profits are less studied in the literature. However, this kind of profit value assignment is actually quite common. For example in the QoS adaption problem [AHHS05], the QoS levels are usually mapped to the profit of items and their values are often consecutive integers. Also in the multi-hop query allocation problem [HLS09], the query range is mapped to the profit and is measured in hop numbers which take also consecutive integer values.

In the linear generating function, we assign p_{ij} with a linear function of the item index j, *i.e.*

$$p_{ij} = \frac{j-1}{n_i - 1}\left(p_i^{\max} - p_i^{\min}\right) + p_i^{\min}. \tag{6.13}$$

For clarity, we use a short hand notation for this linear generating function as follows:

$$p_{ij} = \mathcal{L}\left(p_i^{\min}, p_i^{\max}\right). \tag{6.14}$$

Applying the Profit Generating Functions The generating functions should be applied on each single class to generate the profits. Obviously, one can apply the same function to all classes or change the functions for each class. For the uniform generating function, even when it is applied to all classes with the same parameters, the random nature of the function will give different values for profits in different classes. On the contrary, when the linear generating function is applied to all classes with the same parameters, all classes will have the same profit vector for their items. Therefore, besides applying the same generating function to all classes, we further propose two ways to use the generating functions. The first one is to reproduce the random profit vector generated by a uniform generating function on all classes. This is typically the case when several users (classes) can access the same set of objects (items) with varying quality of service (profits) but the cost of accessing them differs (weights). We explicitly denote the profits generated via this way as:

$$p_{ij} = \mathcal{R}\left(\mathcal{U}\left(p_1^{\min}, p_1^{\max}\right)\right). \tag{6.15}$$

Here, $\mathcal{R}$ signifies Reproducing the first generated profit vector for other classes. The second way to apply the generating functions is to take into account the class index i when deciding the interval from which the values are taken for each class, *e.g.* $\mathcal{U}(10(i-1), 10i)$ or $\mathcal{L}(10(i-1), 10i)$. When the uniform generating function is applied this way, the resulted profits in each class is still randomly selected but profits of different classes are dispersed into different intervals. While the linear generating function is applied, the profits are linearly assigned in different intervals. We denote this special application of generating functions as:

$$p_{ij} = \mathcal{C}(F), \tag{6.16}$$

where F is a generating function with different parameters for different classes and $\mathcal{C}$ signifies that the generating function is Class-dependent.

6.3.2 Generating the Weights

To generate the weights, we can apply a certain correlation on the generating function for each dimension. In particular, we define the following generating functions.

Uncorrelated Generating Function In uncorrelated generating function, we simply assign weights with values uniformly and randomly selected within a certain interval:

$$w_{ijk} = \mathcal{U}\left(w_{ik}^{\min}, w_{ik}^{\max}\right). \tag{6.17}$$

Weakly Correlated Generating Function This generating function is motivated by previous results on KP instances [Pis05]. The motivation is to slightly associate profits to weights, but still with a degree of freedom for each dimension. In our proposal, weights are assigned according to:

$$w_{ijk} = \mathcal{U}\left(\max\left(0, p_{ij} - \frac{p_i^{\max}}{\delta}\right), p_{ij} + \frac{p_i^{\max}}{\delta}\right). \tag{6.18}$$

We will use the following shorthand notation:

$$w_{ijk} = \mathcal{W}(\delta). \tag{6.19}$$

Strongly Correlated Generating Function Strongly correlated generating function is also motivated by previous results where the correlation between profits and weights is strong:

$$w_{ijk} = p_{ij} + \frac{p_i^{\max}}{\delta}. \tag{6.20}$$

We use the following short hand notation for this function:

$$w_{ijk} = \mathcal{S}(\delta). \tag{6.21}$$

Inversed Strongly Correlated Generating Function For inversed strongly correlated generating function, weights are assigned according to:

$$w_{ijk} = p_i^{\max} - \frac{p_{ij}}{\delta}, \tag{6.22}$$

and will be referred to as:

$$w_{ijk} = \mathcal{I}(\delta). \tag{6.23}$$

Note that the inversed strongly correlated generating function is not interesting to be used alone. Interesting cases occur when both strongly correlation and inversed strongly correlation coexist on different weight dimensions. Intuitively, these instances are hard to solve because careful trade-off between weights across dimensions has to be made. Although we are not aware of any realistic MMKP problems of this type, they are still interesting from a theoretical point of view.

Applying the Weight Generating Functions Similar to the profit generating functions, one could apply the same generating function with the same parameter to all dimensions. But it is also possible to apply the same function with different parameters or even different generating functions for dimensions. In addition to simply applying the same generating function with the same parameters on all dimensions, here we propose two ways to apply the weight generating functions. The first one is to include the dimension index k as a parameter of the generating function so that the weight for a dimension k can be chosen in a range that depends on k for the uniform generating function, or the parameter δ can be a function of k for weakly, strongly and inversed strongly correlated generating functions. It is convenient to use a shorthand as follows:

$$w_{ijk} = \mathcal{D}(F), \tag{6.24}$$

where F can be, for example, $\mathcal{U}(1, 10k)$ for the uniform generating function, or $\mathcal{W}(k+5)$ and $\mathcal{S}(k+5)$ for weakly correlated and strongly correlated generating functions, respectively. Here, $\mathcal{D}$ signifies that the generating functions are Dimension-dependent. We could also apply different generating functions to different dimensions, for example, we will generate instances with the inversed strongly correlated generating functions on some dimensions and strongly correlated generating function on others. Under this case, we denote:

$$w_{ijk} = \mathcal{D}(F_1 F_2 \dots), \tag{6.25}$$

where $F_1, F_2, \dots$ are the generating functions in used.

6.3.3 Generating the Knapsack Capacities

Finally, knapsack capacities are generated by extending (6.5) to multiple classes and multiple dimensions. We generate a series of the S instances and the capacity of the kth dimension of the hth generated instance ($h = 1, 2, \dots, S$), denoted as c_k^h, is dispersed from the minimum possible weight to the maximum possible weight:

$$c_k^h = \frac{h}{S+1}\left(\sum_{i=1}^{m} w_{ik}^{\max} - \sum_{i=1}^{m} w_{ik}^{\min}\right) + \sum_{i=1}^{m} w_{ik}^{\min}. \tag{6.26}$$

The parameter h will also be referred to as the *capacity level* of the instance in the series. Note that all the S instances do not need to have the same items (profit and weight assignment). However, in order to investigate the impact of the capacity level on the solution time, we let all S instances in the same series have the same profit and weight values. As a result, instances in a series differ from each other only by their capacities.

6.3.4 Summary of Instance Notations

We denote G-x-y a group of instances. The parameter x indicates the generating function that is chosen to allocate the profits, so x should be picked in

$\{\mathcal{U}, \mathcal{L}, \mathcal{R}, \mathcal{C}(\mathcal{U}), \mathcal{C}(\mathcal{L})\}$. In the same idea, the parameter y indicates how the weights are computed. The set of generating functions considered in this chapter is

$$\{\mathcal{U}, \mathcal{W}, \mathcal{S}, \mathcal{D}(\mathcal{U}), \mathcal{D}(\mathcal{W}), \mathcal{D}(\mathcal{S}), \mathcal{D}(\mathcal{SU}), \mathcal{D}(\mathcal{SI}), \mathcal{D}(\mathcal{SUI})\}.$$

And for the group names, we use the corresponding normal fonts instead of the calligraphic fonts used for the generating functions. For example G-U-W stands for the group of instances with profits generated with the uniform generating function $\mathcal{U}$ and weights generated with the weakly generating function $\mathcal{W}$. Among all combinations, we focus on the instances that either exhibit an interesting behavior, or appear to be standard families of instances. This subset of instance families are detailed in Table 6.2. Note in particular that we create two groups of instances using the $\mathcal{I}$ generating function for some dimensions. These two combinations have been chosen because, as we will show later, they exhibit especially interesting hardness nature. Other combinations are obviously possible and hard instances other than those discussed in this study must exist. Discover more such instances could be an interesting future work.

Table 6.2: Generating Functions for Instances $P(10, 5, 5)$

Group	Profits (p_{ij})	Weights (w_{ijk})		
		Uncorr.	Weakly Corr.	Strongly Corr.
G-U-$*$	$\mathcal{U}(1, 50)$	$\mathcal{U}(1, 10)$	$\mathcal{W}(10)$	$\mathcal{S}(10)$
G-L-$*$	$\mathcal{L}(1, 50)$			
G-U-D($*$)	$\mathcal{U}(1, 50)$	$\mathcal{U}(1, 10k)$	$\mathcal{W}(k+5)$	$\mathcal{S}(k+5)$
G-L-D($*$)	$\mathcal{L}(1, 50)$			
G-C(U)-$*$	$\mathcal{U}(10(i-1), 10i)$	$\mathcal{U}(1, 10)$	$\mathcal{W}(10)$	$\mathcal{S}(10)$
G-C(L)-$*$	$\mathcal{L}(10(i-1), 10i)$			
G-C(U)-D($*$)	$\mathcal{U}(10(i-1), 10i)$	$\mathcal{U}(1, 10k)$	$\mathcal{W}(k+5)$	$\mathcal{S}(k+5)$
G-C(L)-D($*$)	$\mathcal{L}(10(i-1), 10i)$			
G-R-$*$	$\mathcal{R}(\mathcal{U}(1, 50))$	$\mathcal{U}(1, 10)$	$\mathcal{W}(10)$	$\mathcal{S}(10)$
G-R-D(SU)	$\mathcal{R}(\mathcal{U}(1, 50))$	$\mathcal{S}(10), \forall k = 1; \mathcal{U}(1, 10), \forall k \in \{2, 3, 4, 5\}$		
G-R-D(SI)	$\mathcal{R}(\mathcal{U}(1, 50))$	$\mathcal{S}(10), \forall k \in \{1, 2\}; \mathcal{I}(10), \forall k \in \{3, 4, 5\}$		
G-L-D(SU)	$\mathcal{L}(1, 50)$	$\mathcal{S}(10), \forall k = 1; \mathcal{U}(1, 10), \forall k \in \{2, 3, 4, 5\}$		
G-L-D(SI)	$\mathcal{L}(1, 50)$	$\mathcal{S}(10), \forall k \in \{1, 2\}; \mathcal{I}(10), \forall k \in \{3, 4, 5\}$		
G-L-D(SUI)	$\mathcal{L}(1, 50)$	$\mathcal{S}(10), \forall k \in \{1, 2\}; \mathcal{U}(1, 10), k = 3; \mathcal{I}(10), \forall k \in \{4, 5\}$		

6.4 Experiment Study

6.4.1 Experiment Setup

We implemented the BBLP algorithm with C++ programming language and built the binary with GNU g++ version 4.3.0. For the standard solvers, we employed the ILOG CPLEX version 11.2.0 [1] and GNU GLPK version 4.31. For both solvers, we keep the default parameters related with the algorithms.

[1] The CPLEX is licensed to "AMPL Student Edition", which is able to solve problems with up to 300 variables and this is enough for our example problems.

All experiments have been carried out on the same computation platform, which is a Fedora 7 running on an IBM Thinkpad with an Intel Pentium M processor at 1.86GHz and with 1GB memory and 1GB swap space on the hard disk.

We generate the instances described in Table 6.2 with the proposed method, then we challenge the algorithms with these instances. In particular, we generate instances for $P(10, 5, 5)$, which are of the same size as I2 from the OR benchmark library. We have similar results for $P(5, 5, 5)$ and $P(15, 10, 10)$ which correspond to I1 and I3, respectively. However, the former is so easy that the differences are too small, while for the latter, most instances we generated can not be solved in reasonable time. Therefor, only results for $P(10, 5, 5)$ are presented in this chapter. Since both the number of input variables and the values that the variables take have impact on the solution time of an instance, we select the variable values within the same range as I2 so the effects of different variable values are minimized. Notice also that some of the generating functions previously defined have random factors. For groups using these generating functions, we generate 20 series for each group to account for the random effects. For the groups that contain only the deterministic generating functions, *e.g.* linear profits with strongly correlated weights, the parameters of the generating functions determine the uniqueness of the instance, so only one series is evaluated. For each series, we generate 100 instances, *i.e.* $S = 100$. The instance generating program, the configuration files, the generated instances, their solutions and the solution time are available at `http://enstb.org/~gsimon/Resources/MMKP/`.

Some generated instances may be infeasible while others may be too hard to be solved to optimal in reasonable time. For the latter case, the execution time of the algorithm is limited to 600 seconds [2]. As a result, the solution time that will be presented in the following part of this section could be the time for either obtaining the optimal solution, or asserting infeasibility, or the time used when the algorithm is aborted.

6.4.2 Average Solution Time

We first give an overview of the solution time of the generated instances to highlight the existence of hard instances. In Table 6.3, both average and solution times are presented where the average is taken across capacity levels and across multiple series, and the maximum is taken from the average values across multiple series. Comparing with the solution time of I2 presented in Table 6.1, we find that certain groups of instances such as G-L-W, G-L-S, G-L-D(W) and G-L-D(S), *etc.* are much harder.

We can roughly classify these instances into three categories as indicated in the three separated parts in Table 6.3. From top to bottom, results for instances generated with uncorrelated, weakly correlated and strongly correlated generating

[2] We implemented a timing mechanism in the BBLP algorithm. For CPLEX and GLPK, one can specify a time limit by feed the program with a command line parameter. However, the CPLEX and our BBLP implementation use the CPU time while the GLPK uses real time, thus the machine is dedicated to the simulations and the difference is minimized.

functions are listed and a clear trend of increasing solution time could be observed.

We conclude that high correlation between weights and profits generally makes an instance harder. If the profits are chosen according to the linear generating function, instances with weakly and strongly correlated weights become very hard even for the advanced solvers such as CPLEX and GLPK.

Table 6.3: Solution Time (second) of Instances.

Group	CPLEX		GLPK		BBLP	
	Avg.	Max.	Avg.	Max.	Avg.	Max.
G-U-U	0.0051	0.0400	0.0124	0.0770	0.0168	0.2180
G-U-D(U)	0.0068	0.1190	0.0148	0.1820	0.0220	0.5179
G-C(U)-U	0.0047	0.0380	0.0130	0.0810	0.0180	0.3090
G-C(L)-U	0.0046	0.0430	0.0124	0.1470	0.0173	0.3210
G-C(U)-D(U)	0.0053	0.0580	0.0141	0.1140	0.0251	0.2690
G-C(L)-D(U)	0.0058	0.0700	0.0139	0.1040	0.0213	0.2620
G-R-U	0.0052	0.0410	0.0126	0.1130	0.0157	0.1930
G-L-U	0.0056	0.0500	0.0134	0.1020	0.0194	0.3350
G-L-D(U)	0.0070	0.0670	0.0143	0.1700	0.0241	0.6559
G-U-W	0.0243	0.2510	0.0789	0.5049	0.2277	1.6068
G-U-D(W)	0.0282	0.1940	0.0788	0.7339	0.2382	2.4456
G-C(U)-W	0.0105	0.0310	0.0269	0.1560	0.3069	4.6063
G-C(U)-D(W)	0.0101	0.0390	0.0242	0.1840	0.2420	2.6426
G-C(L)-W	0.0121	0.0460	0.0310	0.2150	0.3371	4.8203
G-C(L)-D(W)	0.0118	0.0420	0.0304	0.2270	0.3295	4.7473
G-R-W	0.0437	0.7389	0.1239	1.3708	0.2940	6.9169
G-L-W	0.6814	23.6984	20.5166	598.0341	3.8108	61.2037
G-L-D(W)	0.3460	10.9843	4.2153	587.2337	2.0304	35.4606
G-U-S	0.0051	0.0360	0.0203	0.1130	1.5901	63.4574
G-U-D(S)	0.0094	0.0710	0.0268	0.1470	0.9968	31.5992
G-C(U)-S	0.0025	0.0100	0.0112	0.0350	2.4810	49.7544
G-C(U)-D(S)	0.0042	0.0130	0.0130	0.0810	1.1158	84.2942
G-R-S	0.0252	0.2810	1.1301	51.5342	56.0733	184.0700
G-R-D(SU)	0.0086	0.1770	0.0418	0.8419	35.0805	161.3765
G-R-D(SI)	0.0036	0.0220	0.3080	29.4125	45.3696	169.8772
G-L-S	0.0963	0.6049	227.6951	600.0000	44.0825	186.1967
G-L-D(S)	0.0386	0.7069	300.3657	598.0401	50.9872	186.3017
G-L-D(SU)	7.1919	117.7880	113.2350	594.3146	13.5551	101.1256
G-L-D(SI)	0.0503	0.6669	80.9059	555.7105	14.1198	152.3548
G-L-D(SUI)	9.4867	235.9370	79.9511	592.1970	12.6077	137.0432
G-C(L)-S	35.7737	384.3760	154.1470	597.6111	68.4683	209.6621
G-C(L)-D(S)	25.6856	422.6000	158.5421	598.0471	60.3072	209.4772

6.4.3 Capacity Level and Solution Time

Now we show the relationship between capacity level and solution time of the instances. Figure 6.1 presents the solution time of G-U-$*$ instances according to the capacity level. We can observe that for uncorrelated cases, instances with lower capacity levels are generally very easy while hardest instances emerge at capacity levels between 40 and 50. The easiest uncorrelated instances with lower capacity

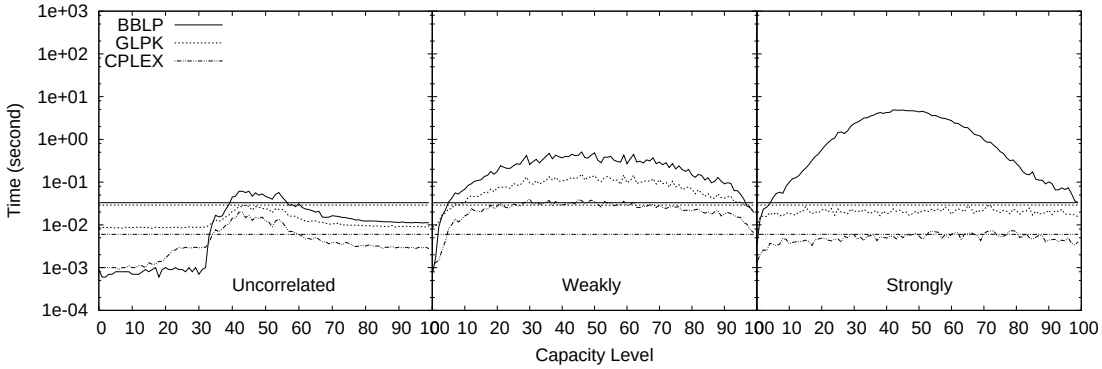


Figure 6.1: Solution times of G-U-∗ instances.

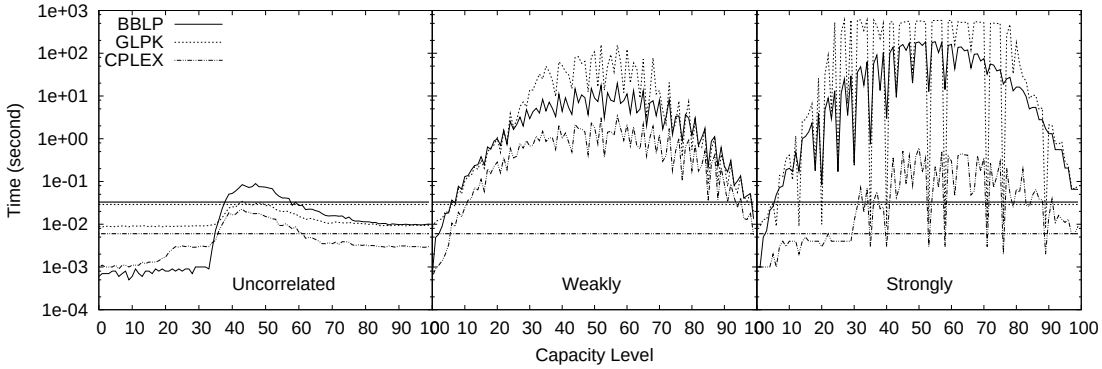


Figure 6.2: Solution times of G-L-∗ instances

level are due to their infeasibility while those with highest capacity level are trivial. While on the other hand, for weakly and strongly correlated cases shown in the middle and right most plots in Figure 6.1, respectively, the hardest instances usually appear at the center of the capacity level. Similar observations could be made from the G-L-∗ cases in Figure 6.2. However, when the linear generating function is used, the weakly and strongly correlated instances become harder.

Figure 6.2 demonstrates also high variability of relative hardness within one series. This is especially obvious for the G-L-W/S instances. Some non-trivial strongly correlated instance could be extremely easy for CPLEX, GLPK and sometimes also for BBLP. The very special combination of capacity level, profit and weight admits very efficient branch-and-bound operation. Furthermore, thanks to the special mechanisms employed by GLPK and CPLEX, these instances can be solved even faster. These mechanisms consist of pre-process that may possibly reduce the number of variables, various branching heuristics and various cutting algorithms. By applying the default parameters of CPLEX and GLPK, these advanced algorithms are enabled and both solvers apply them dynamically during the search process. However, it is quite surprising that BBLP is generally faster than GLPK on strongly correlated instances, and it even achieves similar performance as CPLEX does on weakly correlated instances. Both imply that the additional efforts taken by CPLEX or

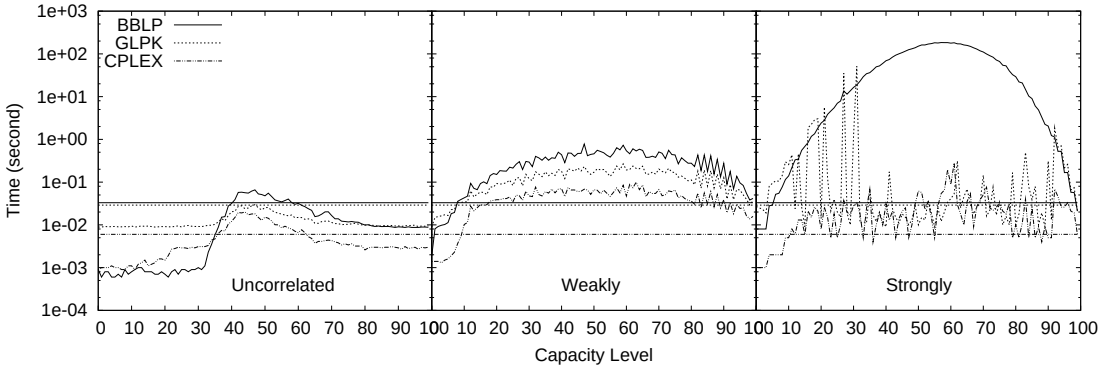


Figure 6.3: Solution times of G-R-∗ instances.

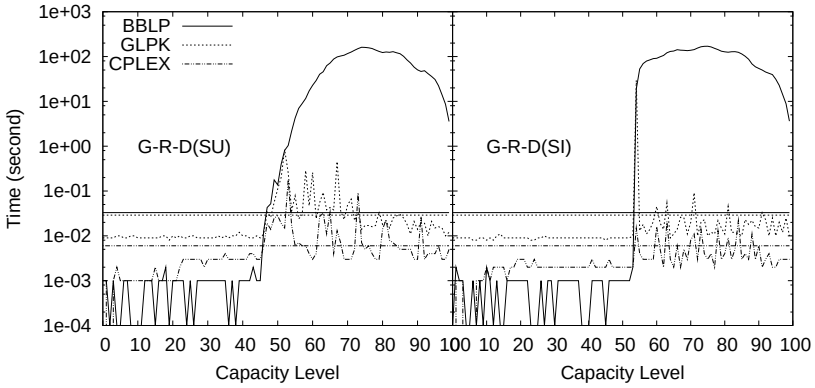


Figure 6.4: Solution times of G-R-D(∗) instances.

GLPK do not help much in solving these instances.

The positions of hard instances are hard to predict when certain correlations exist. Notably, the G-U-W/S, G-L-W/S and G-R-W/S (shown in Figure 6.3) generally have similar properties that the hardest instances appear at 50% capacity and the advanced algorithms could solve certain instances very quickly. However experiments on G-R-D(SI) and G-L-D(∗) (in Figure 6.4 and Figure 6.5, respectively) show different properties. For example, the inversed strongly correlated dimension gives a clear cut on the feasible instances, making the hardest ones appear at a shifted position. The hardest instances of G-R-D(SU) and G-L-D(SU) appear also at positions shifted to the higher capacity levels, as shown in the left most figures of Figure 6.4 and Figure 6.5. Therefore, a rule of thumb is to use the whole series to benchmark the algorithms, instead of with only a few samples.

6.4.4 Non-trivial Infeasible Instances

The instances may be infeasible and they appear often in uncorrelated cases. As we show in Figure 6.6, if an instance is infeasible, it is generally easy for all the three algorithms to detect this fact, partially due to the fact that the LP relaxation for these instances are also infeasible. However, there exist infeasible instances that

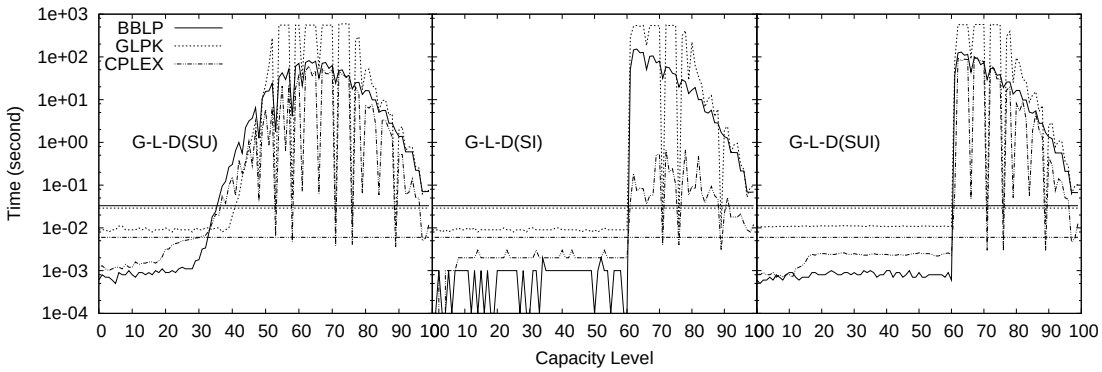


Figure 6.5: Solution time of G-L-D(∗) instances.

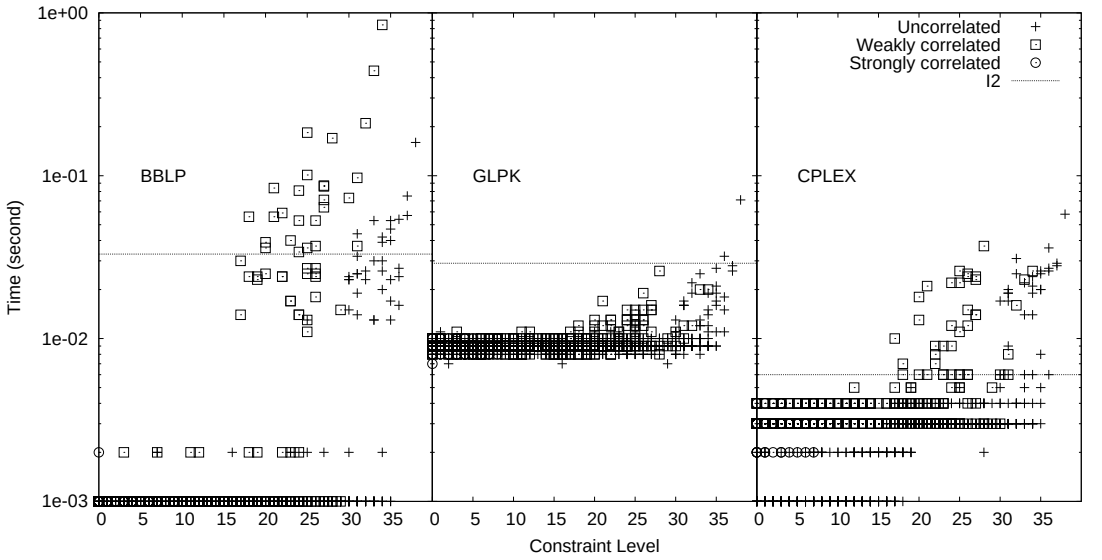


Figure 6.6: Nontrivial infeasible instances, G-C(U)-D(∗) as an example.

are *non-trivial* to detect. The same observation has also been claimed in [KPP04]. These hard infeasible instances usually appear at intermediate capacity levels at which both infeasible and feasible instances exist.

6.4.5 The Critical Dimension

In Figure 6.8(a), we could see that the solution time increases linearly with the number of dimensions for the considered $P(10, 5, *)$ G-L-U instances. However, if the dimensions have mixed correlation properties, the impact of strongly correlation dimension may be dominant. Actually, one strongly correlated weight dimension is enough to make the instance hard. On one hand, results for the G-L-∗ instances with a single dimension are presented in Figure 6.7, which are very similar to their multi-dimension counterparts presented in Figure 6.2, suggesting that instances with a single strongly correlated dimension is already hard. On the other hand, in Fig-

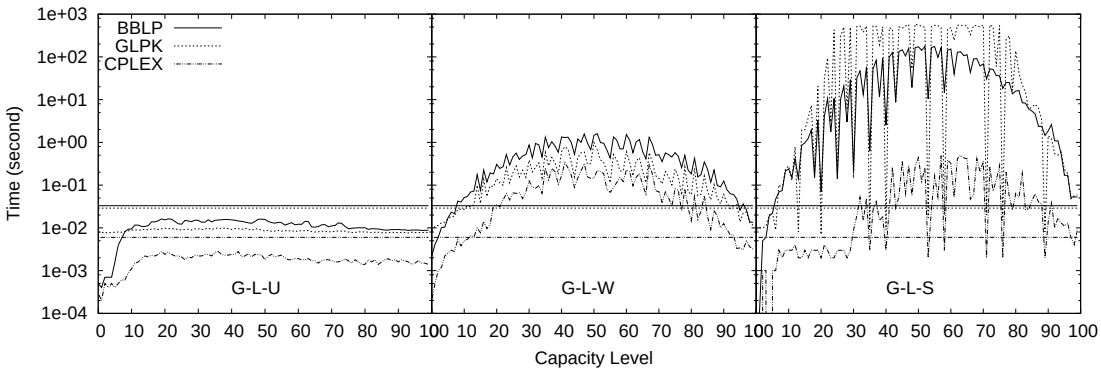
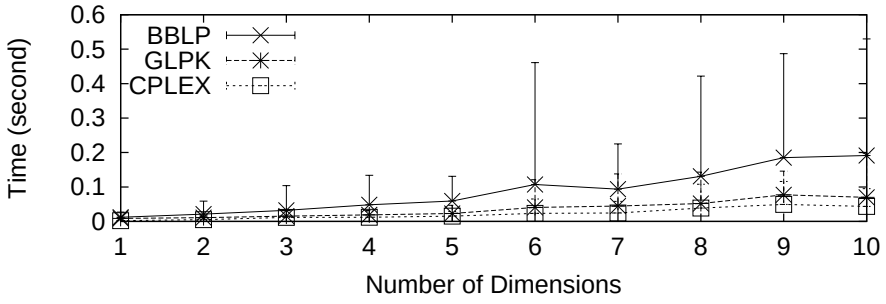


Figure 6.7: Solution times of single dimensional G-L-$*$ instances.

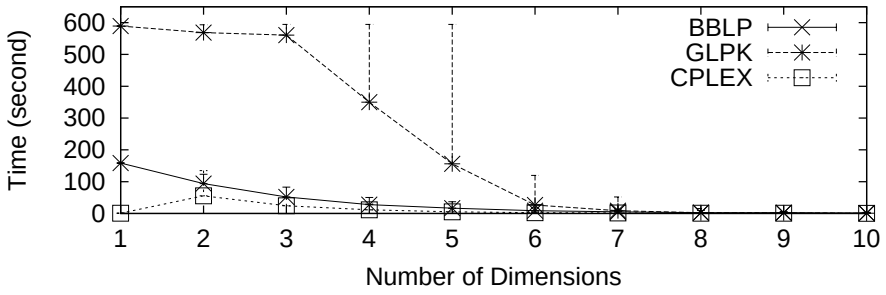
ure 6.8(b), solution time of instances with a single strongly correlated dimension and several uncorrelated dimensions are presented. We find that the solution time generally decreases when more uncorrelated dimensions are added and finally, when there exist many uncorrelated dimensions, the effects of a strongly correlated dimension may be diminished.

6.5 Conclusion

We have proposed systematic methods to generate more comprehensive MMKP instances for benchmarking the algorithms. Several categories of MMKP instances have been produced to demonstrate that some MMKP instances are hard. Experiments on these hard instances with present exact algorithm and solvers also revealed some special structure of the problem. Briefly, the instance is hard to solve when all classes contain the same profit vector and the weights are correlated with the profits. Besides, one strongly correlated dimension of weight is able to render the resulted instances hard to solve. Finally, certain categories of instances are very hard for all considered algorithm and solvers: BBLP, GLPK and CPLEX, even many advanced branching and cutting algorithms are employed by the two generic solvers. Special structural properties of these instances deserve further investigation.



(a) G-L-U instances



(b) G-L-D(SU) instances, with a single strongly correlated dimension.

Figure 6.8: Solution Time vs. Number of Dimensions for $P(10, 5, *)$ instances.

Chapter 7

Epilogue

"The golden age is before us, not behind us."
– William Shakespeare

Wireless sensor networks are becoming a reality. Large scale deployment that provides real-time in-site information to mobile users could be envisioned in the near future. Direct access of mobile users to the sensor nodes simplifies the overall network architecture and is able to keep the traffic local, which is critical for a scalable network. Fairness and efficiency should be simultaneously considered when optimizing the operation of such networks. While most current literature studies these issues with emphasis on the sensor nodes from a networking point of view, our vision is that the fairness of users is especially important when the users are clients of a service providing network. Following this vision, we study the fairness issues in wireless sensor network from a user's point of view.

7.1 Conclusion

We have identified and studied the fair query allocation problem for the multi-user wireless sensor networks. Several related issues, *i.e.* the capacity of general wireless ad-hoc networks and wireless sensor networks, the medium access and networking layer standards for wireless sensor networks, the multidimensional multiple choice knapsack problem and its algorithms and the fairness definitions are briefly surveyed. This part of survey study has provided us a sound knowledge based on which the following aspects of the multi-user query allocation problem have been studied.

(i) Max-min fair query allocation problem in a WSN with densely deployed sensor nodes is defined and discussed. The analysis is based on a disc-shaped continuous-diameter query model and a stream based traffic model. Under these assumptions, the query region of a user is limited by the bandwidth of both the sensors and the users. Thus, user have to cooperate with the sensors in order to achieve the

desired results. Explicit expression of max-min fair query radius under two-user case is derived and the problem under multi-user case is solved with a distributed heuristic. Extensive simulations show the effectiveness of the proposed algorithms and reveal some interesting new problems as well.

(ii) Fair query allocation among the users is also studied with a hop based query model. Under this case, the discrete value of the query radius no longer promises the existence of the max-min fair solution, thus lexicographical max-min fairness has to be exploited instead. We also found it is convenient to model the lexicographical max-min fairness problem as a Multi-dimensional Multiple choice Knapsack Problem. Furthermore, the traditional optimization objective, which maximizes the sum utility of all items in the knapsack, could also be used to maximize the overall query range of users in our case. Based on these observations, we propose a unified framework for the problem description. This framework, firstly, is able to formulate different optimization problems originated in multi-user WSNs; and secondly, implies simple implementation of a uniformed algorithm solving both problems. Extensive simulations have been carried out to evaluate the performance of the algorithms. Different properties between the two optimization objectives are discussed.

(iii) In order for the study above being practically meaningful, we investigate the feasibility of reformulating the problem and implementing the proposed solutions in a IEEE 802.15.4/ZigBee based wireless sensor network. The ZigBee cluster tree mode is considered because of its energy efficiency. Besides, the cluster tree and the related addressing and routing mechanisms allow a fully localized computation when solving the query allocation problem. The proposed distributed algorithms are shown to be effective in approaching the optimal solutions and in controlling the congestion.

(iv) The multidimensional multiple choice knapsack problem has be used to formulate the query allocation problem for a multi-user wireless sensor network with hop-based queries. Many experiments have shown bizarre properties of the MMKP instances, *i.e.* the time used in solving the instances may vary greatly such that smaller instances takes much longer time than larger instances. In the last part of this thesis, we investigated this issue by experiments. A systematic method for generating MMKP instances is proposed and several groups of instances which represent a variety of correlation types between the problem parameters are generated. These instances are tested with the existing BBLP algorithm as well as two optimization software tools, namely the GLPK and CPLEX. The results show that linear profits and high correlation between weights and profits make the instances specially hard to solve for all the three algorithm/solvers, even though some advanced integer programming mechanisms are integrated in the two solvers.

7.2 Limitations and Perspectives

The whole study is based on the vision that large scale wireless sensor network is to be deployed to provide real-time in-site services to multiple mobile users who are able to access the sensor nodes directly. Besides the fairness issues studied in this

thesis, there are several other debatable issues considering this very special network architecture.

Applicability The first question is whether or not the so-called mobile user WSN will be proven useful and indeed get deployed. From current application deployment experiences, many WSNs are of a very limited size with respect to the number of nodes (*e.g.* a dozen of nodes) and the geographical area they are covering (hundreds of square meters). From a technical point of view, the major difficulty for large WSNs are connectivity and scalability. While many researchers are busy solving such problems, other are questioning the needs for a large continuously connected network. They argue that many independent small wireless sensor networks together will be sufficient in providing pervasive services to users. With this debate in mind, we find that the mobile user WSN may satisfy both sides. On one hand, it could scale to large number of nodes covering vast geographical area, while on the other hand, it does not need to be fully connected as the mobile users are always able to retrieve data from sensor nodes which are sufficiently near. As a result, we have a strong feeling that the mobile user WSN at least provides a promising way to future pervasive sensing applications.

Networking or Service? Another interesting issue is should we separate or integrate the networking issues with the service issues? Traditionally, the WSNs are responsible for gathering the raw data (probably with limited in-network processing), while the data are provided to the users via a back-end server. As a result, most researches study the networking aspect without considering the users. We have studied the user aspect with networking limitations as constraints, thus effectively mixed both issues into one problem formulation. While the proposed mobile user WSN architecture is novel and feasible for applications where the significance of the gathered data has strong spacial-temporal dependence, *i.e.* data is meaningful for a short period of time and highly related to where the data is from and they do not need to be stored in general, it is *infeasible* for scenarios where the data should be saved for any reason. For the latter case, nevertheless, it is possible to support the mobile users with the traditional sink based architecture.

Multicast and In-network Processing Our problem formulation and the proposed algorithms work with either shortest path routing or cluster tree hierarchical routing and we consider only unicast routing without in-network processing. In reality, many routing or data gathering protocols exist and some have in-built data aggregation mechanisms to reduce the data traffic. For routing protocols that multicast same copy of data to multiple destinations, or routing protocols with in-network aggregation, our model is no longer valid. As a result, new traffic model is needed under such situations which could be an interesting direction for the future works.

Variable Bandwidth We have based our study on a theoretical capacity results of wireless sensor networks, which have its own limitations. Notably, the results are

scaling laws obtained as an asymptotic value when the number of sensors in a fixed area goes to infinity. In reality, the effective bandwidth between each pair of sender and receiver depends on many factors, *e.g.* the modulation scheme, the transmission power, the transmission state of nearby pairs, the medium access mechanism in use, *etc.* The current problem formulation with a constant shared bandwidth as constraints is not able to reflect these factors. While complex model is required to handle more details, the current solutions could be extended without too much effort. The measurement based approach proposed in this thesis is able to adapt to the dynamics in the bandwidth. To investigate the performance of query allocation algorithms under dynamic bandwidth situation is another perspective on which the current work could be extended.

Mobility Although we have emphasized very much on the mobile users, we have not tackled the issues posed by the mobility in this thesis, *i.e.* all the models, analysis, algorithms are based on static sensors and users. This prevents a direct application of the proposed mechanisms to a real scenarios. In order for this study to be more practically significant, we shall extend our proposals to a mobile user environment, which may not be trivial especially when the convergence of the distributed algorithms has to be enforced.

Security Let mobile users access the sensor nodes directly may raise security issues. Obviously, certain service contract has to be enforced between the users and the network operator. Simple authentication protected by lightweight encryption mechanisms should be implemented on both the sensors and the users.

Why MMKP is Hard? Finally, considering the MMKP problem itself, the empirical study presented in this thesis is only a first step towards understanding its structural properties. We have shown that linear profits and strong correlation between profits and weights make a hard MMKP, however, more investigation is needed to answer why. While the BBLP exact algorithm and the GLPK and CPLEX solvers are used to show these hard instances, further testing these instances with existing heuristics may provide more hints to the question mentioned above. Although some experiences have already shown that certain heuristics are sometimes very efficient in solving the hard instances with special properties, this topic may deserve in-depth study.

Bibliography

[80206] IEEE 802.15.4 standard, September 2006.

[AB06] Bassam Aoun and Raouf Boutaba. Max-min fair capacity of wireless mesh networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.

[AHHS05] S. Alam, M. Hasan, M. Hossain, and A.S.M. Sohail. Heuristic solution of mmkp in different distributed admission control and qos adaptation architectures for video on demand service. *Broadband Networks, 2005 2nd International Conference on*, pages 896–903 Vol. 2, Oct. 2005.

[AMSK01] M. Mostofa Akbar, Eric G. Manning, Gholamali C. Shoja, and Shahadat Khan. Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In *ICCS '01: Proceedings of the International Conference on Computational Science-Part II*, pages 659–668, London, UK, 2001. Springer-Verlag.

[ARK+06] Md. Mostofa Akbar, M. Sohel Rahman, M. Kaykobad, E. G. Manning, and G. C. Shoja. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Comput. Oper. Res.*, 33(5):1259–1273, 2006.

[AS01] Mani B. Strivastava Andreas Savvides, Chih-Chieh Han. Dynamic fine-grained localization in ad-hoc networks of sensors. In *ACM Mobicom*, 2001.

[BBB04] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: sensor networks in agricultural production. *Pervasive Computing, IEEE*, 3(1):38–45, Jan.-March 2004.

[Bou06] J.-Y. Le Boudec. Rate adaptation, congestion control and fairness: a tutorial. In *Technical report.*, 2006.

[CA06] Maria Chantzara and Miltiades E. Anagnostou. Mvrc heuristic for solving the multi-choice multi-constraint knapsack problem. In *International Conference on Computational Science (1)*, pages 579–587, 2006.

[CES04] David Culler, Deborah Estrin, and Mani Srivastava. Overview of sensor networks. *IEEE Computer*, 37(8):41–49, 2004.

[CFX07] Shigang Chen, Yuguang Fang, and Ye Xia. Lexicographic maxmin fairness for data collection in wireless sensor networks. *IEEE Trans. on Mobile Computing*, 6(7):762–776, 2007.

[CH08] N. Cherfi and M. Hifi. A column generation method for the multiple-choice multi-dimensional knapsack problem. Comput. Optim. App. online first, Springer Netherlands, 2008.

[CK07] Jie Chen and Xenofon Koutsoukos. Survey on coverage problems in wireless ad hoc sensor networks. In *the IEEE SouthEastCon Conf.*, March 2007.

[CM06] Canfeng Chen and Jian Ma. Mobile enabled large scale wireless sensor networks. In *International Conference on Advanced Communication Technology (ICACT)*, 2006.

[cpl] `http://www.ilog.com/products/cplex/`.

[CW04] Mihaela Cardei and Jie Wu. *Handbook of Sensor Networks*. CRC Press, Boca Raton, FL, USA, 2004.

[DW98] M. E. Dyer and J. Walker. Dominance in multi-dimensional multiple-choice knapsack problems. *Asia-Pacific Journal of Operational Research*, 15(2):159–168, 1998.

[EB04] Cheng Tien Ee and Ruzena Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 148–161, New York, NY, USA, 2004. ACM.

[GG61] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[GG63] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem-part ii. *Operations Research*, 11(6):863–888, 1963.

[GK00] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[GK04] Y. Ganjali and A. Keshavarzian. Load balancing in ad hoc networks: Single-path routing vs. multi-path routing. In *IEEE INFOCOM*, 2004.

[glp] `http://www.gnu.org/software/glpk/`.

[HB01] Xiao Long Huang and Brahim Bensaou. On max-min fairness and scheduling in wireless ad-hoc networks: analytical framework and implementation. In *ACM MobiHoc*, 2001.

[Hir08] Chaitr S. Hiremath. *New Heuristic And Metaheuristic Approaches Applied To The Multiple-choice Multidimensional Knapsack Problem.* PhD thesis, School of Graduate Studies, Wright State University, 2008.

[HJB04] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 134–147, New York, NY, USA, 2004. ACM.

[HLS09] Bing Han, J. Leblet, and G. Simon. Query range problem in wireless sensor networks. *Communications Letters, IEEE*, 13(1):55–57, January 2009.

[HMS75] Cornelis A. De Kluyver Harvey M. Salkin. The knapsack problem: A survey. *Naval Research Logistics Quarterly*, 22(1):127–144, 1975.

[HMS04] M. Hifi, M. Michrafy, and A. Sbihi. Heuristic algorithms for the multiple-choice multi-dimensional knapsack problem. *J Operat Res Soc*, 55(12):1323–1332, 2004.

[HMS06] M. Hifi, M. Michrafy, and A. Sbihi. A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, 33(2-3):271–285, 2006.

[HS07] Bing Han and Gwendal Simon. Fair capacity sharing among multiple sinks in wireless sensor networks. In *the IEEE MASS Conf.*, pages 1–9, Oct. 2007.

[JCH98] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301, 1998.

[Kel97] Frank Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[Kha98] Shahadat Khan. *Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture.* PhD thesis, University of Victoria, 1998.

[KLMA02] Shahadat Khan, Kin F. Li, Eric G. Manning, and Md. Mostofa Akbar. Solving the knapsack problem for adaptive multimedia systems. *Stud. Inform. Univ.*, 2(1):157–178, 2002.

[KMT98] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. 49, 1998.

[KPP04] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, October 2004.

[LC07] Chae Y. Lee and Hee K. Cho. Discrete bandwidth allocation considering fairness and transmission load in multicast networks. *Computers and Operations Research*, 34(3):884–899, 2007.

[Lev] Philip Levis. TEP 111: message_t. `http://www.tinyos.net/tinyos-2.x/doc/pdf/tep111.pdf`.

[LH05] Jun Luo and Jean-Pierre Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *IEEE INFOCOM*, 2005.

[LLRS99] Chen Lee, John Lehoczky, Ragunathan (Raj) Rajkumar, and Dan Siewiorek. On quality of service optimization with discrete qos options. In *RTAS'99: Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium*, page 276, Washington, DC, USA, 1999. IEEE Computer Society.

[LMC04] Chae Y. Lee, Young P. Moon, and Young Joo Cho. A lexicographically fair allocation of discrete bandwidth for multirate multicast traffics. *Computers and Operations Research*, 31(14):2349–2363, 2004.

[LMFJ+04] Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, 2004.

[LY06] Mo Li and Baijian Yang. A survey on topology issues in wireless sensor networks. In *the 2006 International Conf. on Wireless Networks (ICWN)*, June 2006.

[MA06] Rene Muller and Gustavo Alonso. Efficient sharing of sensor networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.

[MDMLN03] Daniel Marco, Enrique J. Duarte-Melo, Mingyan Liu, and David L. Neuhoff. On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data. In *IEEE/ACM IPSN*, 2003.

[MFA07] Guoqiang Mao, Barış Fidan, and Brian D. O. Anderson. Wireless sensor network localization techniques. *Comput. Netw.*, 51(10):2529–2553, 2007.

[MJS97] Martin Moser, Dusan P Jokanović, and Norio Shiratori. An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 80(3):582–589, 1997.

[Mos96] Martin Moser. Declarative scheduling for optimally graceful qos degradation. In *ICMCS '96: Proceedings of the 1996 International Conference on Multimedia Computing and Systems (ICMCS '96)*, page 0086, Washington, DC, USA, 1996. IEEE Computer Society.

[MT90] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[MW00] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Netw.*, 8(5):556–567, 2000.

[NPar] D. Nace and M. Pioro. A tutorial on max-min fairness and its applications to routing and load-balancing in telecommunication networks. *IEEE Communications Surveys and Tutorials*, to appear.

[ns2] `http://www.isi.edu/nsnam/ns/`.

[OPT05] Wlodzimierz Ogryczak, Michal Pioro, and Artur Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of Telecom. and Information Tech.*, 3:43–56, 2005.

[OR-] `ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/OR-Benchmark.html`.

[OS06] Wlodzimierz Ogryczak and Tomasz Sliwinski. On direct methods for lexicographic min-max optimization. In *the ICCSA Conf.*, volume 3982 of *LNCS*, pages 802–811, May 2006.

[PHD05] R. Parra-Hernandez and N.J. Dimopoulos. A new heuristic for solving the multichoice multidimensional knapsack problem. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 35(5):708–717, Sept. 2005.

[Pis99] David Pisinger. Core problems in knapsack algorithms. *Oper. Res.*, 47(4):570–575, 1999.

[Pis05] David Pisinger. Where are the hard knapsack problems? *Comput. Oper. Res.*, 32(9):2271–2284, 2005.

[PT07] Meng-Shiuan Pan and Yu-Chee Tseng. The orphan problem in zigbee-based wireless sensor networks. In *Proceedings of the 10th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 95–98, 2007.

[RB06] Bozidar Radunovic and Jean-Yves Le Boudec. A Unified Framework for Max-Min and Min-Max Fairness with Applications. *ACM/IEEE Trans. on Networking*, 15(5):1073–1083, 2006.

[RC08] Bhaskaran Raman and Kameswari Chebrolu. Censor networks: a critique of "sensor networks" from a systems perspective. *SIGCOMM Comput. Commun. Rev.*, 38(3):75–78, 2008.

[RD] Gallager Robert and Bertsekas Dimitri. *Data Networks*. Prentice Hall, 2nd edition.

[RGGP06] Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, and Konstantinos Psounis. Interference-aware fair rate control in wireless sensor networks. In *ACM SIGCOMM*, 2006.

[RSZ04] Cauligi S. Raghavendra, Krishna M. Sivalingam, and Taieb Znati, editors. *Wireless Sensor Networks*. Springer, 2004.

[SARN08] Abu Zafar Shahriar, M. Mostofa Akbar, M. Sohel Rahman, and Muhammad Abdul Newton. A multiprocessor based heuristic for multi-dimensional multiple-choice knapsack problem. *J. Supercomput.*, 43(3):257–280, 2008.

[Sbi07] Abdelkader Sbihi. A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. *Journal of Combinatorial Optimization*, 13(4):337–351, 2007.

[SIH05] Md Waselul Haque Sadid, Md Rabiul Islam, and S M Kamrul Hasan. A new strategy for solving multiple-choice multiple-dimension knapsack problem in pram model. In *Asian Applied Computing Conference*, 2005.

[SK04] Avinash Sridharan and Bhaskar Krishnamachari. Max-min fair collision-free scheduling for wireless sensor networks. In *IEEE International Performance Computing and Communications Conference (IPCCC)*, 2004.

[SK07] Avinash Sridharan and Bhaskar Krishnamachari. Maximizing network utilization with max-min fairness in wireless sensor networks. In *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2007.

[SOBAA03] Yogesh Sankarasubramaniam, Özgür B. Akan, and Ian F. Akyildiz. Esrt: event-to-sink reliable transport in wireless sensor networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 177–188, New York, NY, USA, 2003. ACM.

[ST00] Saswati Sarkar and Leandros Tassiulas. Fair allocation of discrete bandwidth layers in multicast networks. In *the IEEE INFOCOM Conf.*, volume 3, pages 1491–1500, Mar. 2000.

[SY04] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *Network, IEEE*, 18(4):45–50, July-Aug. 2004.

[Toy75] Yoshiaki Toyoda. A simplified algorithm for obtaining approximate solutions to zero-one programming problems. *Management Science*, 21(12):1417–1427, 1975.

[WDM01] J. Widmer, R. Denda, and M. Mauve. A survey on TCP-friendly congestion control. *IEEE Network*, 15(3):28–37, 2001.

[WEC03] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. Coda: congestion detection and avoidance in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 266–279, New York, NY, USA, 2003. ACM.

[WSL+06a] Chonggang Wang, Kazem Sohraby, Victor Lawrence, Bo Li, and Yueming Hu. Priority-based congestion control in wireless sensor networks. In *SUTC '06: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06)*, pages 22–31, Washington, DC, USA, 2006. IEEE Computer Society.

[WSL+06b] Chonggang Wang, Kazem Sohraby, Bo Li, Mahmoud Daneshmand, and Yueming Hu. A survey of transport protocols for wireless sensor networks. *IEEE Network*, 20(3):34–40, 2006.

[YCNCC06] Ch. Ykman-Couvreur, V. Nollet, Fr. Catthoor, and H. Corporaal. Fast multi-dimension multi-choice knapsack heuristic for mp-soc run-time management. In *System-on-Chip, 2006. International Symposium on*, pages 1–4, Nov. 2006.

[YMG08] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.

[ZG04] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004.

[zig08] ZigBee alliance document 053474r17: Zigbee specification, Jan. 2008.

[ZL] J. Zheng and J. Myung Lee. Low rate wireless personal area networks - ns2 simulation platform. `http://www-ee.ccny.cuny.edu/zheng/pub/`.

Publications

International Journals

[1] Bing Han, Jimmy Leblet, Gwendal Simon, Hard Multidimensional Multiple Choice Knapsack Problems, an Empirical Study, to appear in *Computers & Operations Research*.

[2] Bing Han, Jimmy Leblet, Gwendal Simon, Query Range Problem in Wireless Sensor Networks, in *IEEE Communications Letters*, Vol. 13, No. 1, pp. 55-57, January 2009.

International Conferences

[3] Yiping Chen, Bing Han, Jimmy Leblet, Gwendal Simon, Gilles Straub, Network-Friendly Box-Powered Video Delivery System, to appear in *the 21st International Teletraffic Congress (ITC 21)*.

[4] Bing Han, Gwendal Simon, Optimizing Multi-hop Queries in ZigBee Based Multi-sink Sensor Networks, in *the 10th International Conference on Distributed Computing and Networking (ICDCN'09)*, Hyderabad, India, January 3-6, 2009. (Regular paper, acceptance rate: 24/176)

[5] Bing Han, Gwendal Simon, Fair Capacity Sharing Among Multiple Sinks in Wireless Sensor Networks, in *the Fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'07)*, Pisa, Italy. October 8-11, 2007. (acceptance rate: 67/265)

[6] Bing Han, Dongliang Xie, Le Tian, Biao Ren, Shiduan Cheng, An Adaptive Location Service for Wireless Sensor Networks with Mobile Sinks, in *the International Conference on Wireless and Mobile Communications, (ICWMC'06)*, Bucharest, Romania, July 29-31, 2006.

Internal Workshops

[7] Bing Han, Jimmy Leblet, Gwendal Simon, On Sharing the Capacity of Hybrid Ad-Hoc Networks, in *the Autonomous and Spontaneous Networks Symposium*, Telecom ParisTech, Paris, November 20-21, 2008. (poster)

[8] Bing Han, Gwendal Simon, On the Optimality of Max-min Fair Query in Multi-Sink Wireless Sensor Networks, in *the Internal Workshop of Programme Initiative of GET, on Réseaux Spontanés*, Paris, Octobre 30-31, 2007.

Working Papers

[9] A Box-Powered Video Delivery Network Paying Attention to Network Operators, with Yiping Chen, Jimmy Leblet and Gwendal Simon.

[10] Optimal Distribution of Media Descriptions in Large Networks, with Jacob Chakareski, Pascal Frossard, Jimmy Leblet and Gwendal Simon.

Glossary

–A–	
ACK	ACKnowledgement
AIMD	Additive Increase/Multiplicative Decrease
AODV	Ad hoc On-Demand Distance Vector Routing
ARP	Address Resolution Protocol
ASK	Amplitude-Shift Keying
–B–	
BBLP	Branch and Bound with Linear Programming (algorithm for MMKP)
BKP	Bounded Knapsack Problem
BPSK	Binary Phase-Shift Keying
–C–	
CAP	Contention Access Period
CCA	Clear Channel Assessment
CCP	Complementary Constructive Procedure (Hifi's MMKP Algorithm)
CENS	Center for Embedded Networked Sensing
CDMA	Code Division Multiple Access
CFP	Contention Free Period
CGBA	Column Generation Based Algorithm (Sbihi's MMKP Algorithm)
CH1	Hiremath's Heuristic for MMKP, version 1
CH2	Hiremath's Heuristic for MMKP, version 2
C-HEU	Convex-hull based HEU (Akbar's MMKP Algorithm)
CP	Constructive Procedure (Hifi's MMKP Algorithm)
CPU	Central Processing Unit
CRCW-PRAM	Concurrent Read Concurrent Write Parallel Access Machine
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
–D–	
DARPA	Defense Advanced Research Projects Agency

Der_Algo	Derived Algorithm (Hifi's MMKP Algorithm)
DIS	the DIStributed algorithm for max-min fairness
DoD	Department of Defense
DSN	Distributed Sensor Network
–E–	
EMKP	Exact algorithm for MMKP (Sbihi's Algorithm)
–F–	
FDMA	Frequency-Division Multiple Access
FFD	Full Functional Device
FLTS	First Level Tabu Search (Hiremath's MMKP Algorithm)
–G–	
GLPK	GNU Linear Programming Kit
GLS	Geographic Location Service
GPS	Global Positioning System
GTS	Guaranteed Time Slot
–H–	
HEU	HEUristic Algorithm for MMKP (Khan's Algorithm)
HMMKP	Heuristic for MMKP (Parra-Hernández's Algorithm)
–I–	
IEEE	Institute of Electrical and Electronics Engineers
I-HEU	Incremental HEU (Akabar's MMKP Algorithm)
ISM	Industrial, Scientific and Medical
–K–	
KP	Knapsack Problem
–L–	
LOCAL	the LOCAL algorithm for max-min fairness
LP	Linear Programming
LP-WPAN	Low Power Wireless Personal Area Network
–M–	
MAC	Media Access Control
MEMS	Micro-Electro-Mechanical System
MCKP	Multiple Choice Knapsack Problem
MDKP	Multiple Dimension Knapsack Problem
M-HEU	Modified HEU algorithm (Akabar's MMKP Algorithm)
MIP	Mixed Integer Programming
MKP	Multiple Knapsack Problem
MMF	Max min Fairness

MMKP	Knapsack Problem
MNU	Max Network Utility
MRLS	Modified Reactive Local Search (Hifi's MMKP Algorithm)
MSE	Mean Squared Error
MVRC	Maximizing Value per Resource Consumption heuristic (Chantzara's MMKP Algorithm)

–N–

NACK	Negative ACKnowledgement
NIB	Network Information Base
NP	Non-deterministic Polynomial
NSF	National Science Foundation
NWK	NetWorK (layer)

–O–

OPT	the OPTimal algorithm for max-min fairness
OR	Operations Research

–P–

PAN	Personal Area Network
PDA	Personal Digital Assistant
PHY	PHYsical (layer)

–Q–

QoS	Quality of Service
QPSK	Quadrature Phase-Shift Keying

–R–

RFD	Reduced Functional Device
RLS	Reactive Local Search (Hifi's MMKP Algorithm)

–S–

SEM	Standard Error of the Mean

–T–

TCP	Transmission Control Protocol
TDMA	Time-Division Multiple Access
TTDD	Two-Tier Data Dissemination

–U–

–V–

VLSI	Very-Large-Scale Integration
VoD	Video on Demand

–W–

WSN	Wireless Sensor Networks

–Z–

ZC	ZigBee Controller
ZED	ZigBee End Device
ZR	ZigBee Router