DISTRIBUTED ROUTING ALGORITHMS FOR SENSOR NETWORKS

THÈSE N^{O} 3420 (2005)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut de systèmes de communication

SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Guillermo BARRENETXEA

ingénieur de télécommunication, Universidad Publica de Navarra, Espagne
de nationalité espagnole

acceptée sur proposition du jury:

Prof. M. Vetterli, Dr B. Beferull-Lozano, directeurs de thèse
Prof. G. Karlsson, rapporteur
Prof. J.-Y. Le Boudec, rapporteur
Prof. B. Plattner, rapporteur

Lausanne, EPFL
2006

"Non gogoa, han zangoa"
Esaera zaharra

Abstract

Recent advances in wireless communications and computing technology are enabling the emergence of low-cost devices that incorporate sensing, processing, and communication functionalities. A large number of these devices are deployed in the field to create a sensor network for both monitoring and control purposes. Sensor networks are currently an active research area mainly due to the potential of their applications. However, the deployment of a working large scale sensor network still requires solutions to a number of technical challenges that stem primarily from the constraints imposed by simple sensor devices: limited power, limited communication bandwidth, and small storage capacity. In view of all these particular constraints, we require a new paradigm for communication, which consists of new algorithms specifically conceived for sensor networks.

This thesis concentrates on the routing problem, that is, moving data among different network locations, and on the interactions between routing and coding, that is, how sensors code the observations. We start by designing efficient and computationally simple decentralized algorithms to transmit data from one single source to one single destination. We formalize the corresponding routing problem as a problem of constructing suitably constrained random walks on random graphs and derive distributed algorithms to compute the local parameters of the random walk that induces a uniform load distribution in the network. The main feature of this routing formulation is that it is possible to route messages along all possible routes between the source and the destination node, without performing explicit route discovery/repair computations and without maintaining explicit state information about available routes at the nodes.

A natural extension to the single-source/single-destination scenario is to consider multiple sources and/or multiple destinations. Depending on the structure and goal of the network, nodes exhibit different communication patterns. We analyze the problem of routing under three different communication models, namely uniform communication, central data gathering, and border data gathering. For each of these models, we derive capacity limits and propose constructive routing strategies that achieve this capacity.

An important constraint of sensor networks is the limited storage capacity available at the nodes. We analyze the problem of routing in networks with small buffers. We develop new approximation models to compute the distribution on the queue size at the nodes which provide a more accurate distribution than the usual Jackson's Theorem. Using these models, we design routing algorithms that minimize buffer overflow losses.

Routing in large and unreliable networks, such as sensor networks, becomes prohibitively complex in terms of both computation and communication: due to temporary node failures, the set of available routes between any two nodes changes randomly. We demonstrate that achieving robust communications and maximizing the achievable rate per node are incom-

patible goals: while robust communications require the use of as many paths as possible between the source and the destination, maximizing the rate per node requires using only a few of the available paths. We propose a family of routing algorithms that explores this trade-off, depending on the degree of reliability of the network.

The performance of routing algorithms in sensor networks can be significantly improved by considering the interaction of the source coding mechanism with the transport mechanism. We jointly optimize both the source coding and the routing algorithm in a common scenario encountered in sensor network, namely, real-time data transmission. We demonstrate that the combination of specially designed coding techniques, such as multiple description coding, and multipath routing algorithms, performs significantly better that the usual routing and coding schemes.

In summary, this thesis revisits the classic routing problem in the light of distributed schemes for networks with resource-limited nodes.

Résumé

Les récents progrès des communications sans fils et des technologies informatiques ont permis l'apparition de dispositifs à bas coûts intégrant les fonctionnalités de captage, de traitement et de communication. Un grand nombre de ces dispositifs sont deployés dans la nature afin de créer un réseau de capteurs à des fins aussi bien de contrôle que de monitorisation. Le fort potentiel d'applications des réseaux de capteurs en font un domaine de recherche très actif.

Cependant, la conception d'un réseau de capteurs de grande taille se heurte à un certain nombre de difficultés techniques provenant des contraintes imposées par les capacités réduites des capteurs individuels: basse puissance, capacité de communication réduite et faible capacité de stockage. Afin de surmonter ces difficultés, de nouveaux paradigmes de communication sont nécessaires: il y a un besoin de nouveaux algorithmes spécifiques aux réseaux de capteurs. Cette thèse se concentre sur la question du routage, à savoir le transfert de données entre différentes parties du réseau, et son interaction avec le codage, à savoir, comment les capteurs codent leurs observations.

Dans un premier temps nous avons conçu un algorithme décentralisé simple et efficace pour transmettre les données d'une source unique à une destination unique. Dans ce but, on reformule le problème du routage en celui d'imposer les contraintes appropriées à des chemins aléatoires sur des graphes (aléatoires). En suite nous dérivons des algorithmes distribués pour trouver les paramètres locaux des marches aléatoires qui induiront un certain propriété de distribution de la charge dans le réseau. La caractéristique principale de cette formulation est que nous sommes capables de transmettre des messages entre une source et une destination par tous les chemins possibles, en nous affranchissant d'une reconnaissance explicite du chemin au préalable, et sans devoir tenir à jour une information sur l'état des chemins à chaque noeud.

Une extension naturelle de ce scenario est de considérer plusieurs sources et/ou destinations. Selon la structure et la fonction du réseau, les noeuds présentent différents motifs de communication. Nous avons traité le problème dans le cadre de trois modèles de communication : la communication uniforme, "central data gathering" et "border data gathering". Dans le cadre de chacun de ces modèles, nous avons étudié les limites de capacités et avons explicité les stratégies de routage qui permettent de les atteindre.

Une limitation courante des réseaux de capteurs est leur faible capacité de stockage aux noeuds du réseau. Nous avons donc analysé le problème du routage dans des réseaux avec faible mémoire tampon, ce qui a necessité de nouveaux modèles pour obtenir la distribution de la taille des queues aux noeuds. Nous avons ainsi proposé une alternative au traditionnel théorème de Jackson, qui permet une évaluation plus précise des distributions. Ce faisant, nous avons obtenu des algorithmes qui minimisent les pertes du au débordement de la mé-

moire tampon.

Nous nous sommes également intéresés au problème du routage dans des réseaux défaillants, où chaque noeud est susceptible, temporairement et de façon aléatoire, de tomber en panne. Nous montrons alors que, d'une part robustesse des communications et d'autre part maximisation du traffic par noeud , sont des objectifs contradictoires. La robustesse implique l'utilisation d'un nombre de chemins (entre la source et la destination) aussi grand que possible, tandis que maximiser l'utilisation d'un noeud incite à n'utiliser qu'un nombre restreint de chemins possibles. Nous montrons comment obtenir différents compromis, selon le degré de fiabilité du réseau.

Enfin, nous nous sommes également intéresés à l'interaction entre le codage à la source et le mécanisme de transport dans le réseau de capteurs. Nous nous sommes dans un premier temps intéresé à l'utilisation des techniques de codage par descriptions multiples pour la transmission de données en temps réel à travers des réseaux de capteurs : nous avons formulé et résolu analytiquement un problème de réseau qui optimise simultanément le codage à la source et le mécanisme de routage. Nous avons ensuite considéré les aspects pratiques de l'utilisation des techniques de codage par descriptions multiples dans des réseaux de capteurs à grande échelle, où le nombre de chemins disponibles entre une source et une destination est grand.

Acknowledgments

I have been so lucky to have two thesis advisors, Martin Vetterli and Baltasar Beferull-Lozano. Martin's fresh ideas and broad knowledge have been of invaluable support for the completion of this thesis, as well as the motivation boosts I received after every highly intense weekly meetings. During this four years, Martin has not only revealed his scientific qualities but also his human qualities: merci pour les heures de divan. Baltasar's enthusiasm and determination helped me to find my way out of the maze of hypothetical ideas where I so frequently got lost. Thanks for all these long meetings at late hours spent in polishing raw ideas, and additionally, for those exquisite paellas and fideuas (Thanks also to Julia!)

I'm grateful to all the past and present members of LCAV for the great working atmosphere during the past four years. In particular, I thank Henri, as an officemate, for changing my research vision by showing me that things that work in theory do not necessary work in practice (in fact, rarely do), and as a friend, for all the great time outside the office. I would also like to thank the rest of the PSE gang, Andrea and Titi, for very interesting research discussions, as well as for Andrea's mountain tours and Titi's guitar lessons. If everything goes wrong, I can always earn my living with "No woman no cry". Special thanks goes also to Jocelyne, for her fast and efficient response to all the administrative problems I had. I would like also to thank the members of my thesis committee for the comprehensive review of my thesis and the constructive discussion during the thesis defense.

If there is someone to blame for me taking this adventure is Rakel, because she convinced me how cool communications systems were, and disintegrated my future as economist.

This thesis would not have been possible without the correct balance between work and leisure. During all this time, mountains have always been my refuge and my source of energy. Fortunately, while hanging on a rope or sliding down a white slope, there is no much time to think about research. I would like to thank all the people that shared these special feelings with me, especially Josep, Marie, Thomas, and Joao.

I would also like to thank Alex, my allergy mate, not only for adapting his lunch habits to my very constrained restaurant choice, but also for being a constant support in the roller-coaster of my motivation. It has been great to share this long ride with you! My gratitude goes also to Faye, for her valuable corrections.

Many thanks go to Historias, Brujula, Moster, and Marcos, for frequently reminding me how hard working is during Sunday mornings and for all the great memories that I'll retain of these intense four years. To my basque friends, Alberto, Kintxo, Tiko, and Davitxu, for making me feel as the prodigal son. Even being more that one thousand kilometers away, I have the comfort of knowing that they are always close to me. And to Txiki, for recalling me that doing things at the last moment is not necessarily the best solution, and most importantly, for providing me the energy I needed to write this thesis.

Last but not least, I thank my family for their continuous support since the day I started this long trip now twelve years ago.

Contents

List of Figures

List of Tables

Glossary of Notation and Acronyms

We include here the notation used in the thesis. In those few cases where a symbol has more than one meaning, the context (or a specific statement) resolves the ambiguity.

$\|X\|$	Cardinality of set X
$[i_k, j_k]$	Coordinates of node d_k in a lattice
(d_i, d_j)	Link between nodes d_i and d_j
$\{s_k, t_k\}$	Labels of node d_k in a lattice
$\mathcal{A}(d_i, T_i)$	Set of ancestors of d_i in the tree T_i
B	Number of bits per sample
$B_d = B/M$	Number of bits per description
C	Network capacity
C_l	Link capacity
$d_k \in V$	Node k of the network
$d_{BS} \in V$	Base station
D	Distortion
D_s	Side Distortion
D_c	Central Distortion
$Diag(l)$	l-th Diagonal of the lattice: set nodes $[i_k, j_k] \in V$ such that $i_k + j_k = l$.
$Diag(d_k)$	Set of nodes that belong to the same diagonal as d_k
$\mathcal{D}(d_i, T_i)$	Set of descendants of d_i in the tree T_i
$DF(\Pi)$	Distribution factor of Π
E	Set of links in the network
$f(\Pi)$	Fairness of routing algorithm Π
$F^{\Pi}(d_i, d_j, d_k)$	Traffic generated at d_i with destination d_j that flows through d_k according to Π
F_τ	Cumulative distribution function of τ
G	Graph representing the network
$h(d_k, d_l)$	Shortest distance in hops between nodes d_k and d_l
$h_e(d_k)$	Shortest distance in hops from d_k to the nearest node on the boundary of the lattice
H	Packet header size in bits
K	Packet size in bits

$l = (d_k, d_l) \in E$	Communication channel between nodes d_k and d_l
$\mathcal{L}$	Laplace transform
$\overline{L}(N)$	Average distance in hops between any source/destination
L_Q	Maximum Hamiltonian paths length
M	Number of descriptions
$n(d_k) = \lvert\varphi^n(d_k)\rvert$	Number of neighbors of d_k
n_{avg}	Average number of neighbors per node
N	Total number of nodes in the network
$N_1 = \sqrt{N}$	Number of nodes per dimension in lattice networks
q_k	Queue associated to node k
Q	Buffer size per node
$\mathcal{R}$	Rate per node
$\mathcal{R}^{\Pi}_{\text{sup}}$	Maximum achievable rate using Π
R_{tx}	Transmission range
$s(d_i, d_j)$	Euclidean distance between d_i and d_j
S_p	Number of samples per packet
$S(t)$	Total number of packets in the network at time t
$\text{SPR}(d_i, d_j)$	Set of nodes that belong to any shortest path between d_i and d_j
t_{Bo}	Backoff time
T	Sampling period
T_{BO}	Backoff window size
$T(d_i, d_j)$	Probability of node d_i communicating to node d_j
V	Set of nodes in the network
$w_{\Pi}(d_k)$	Weight of d_k
$\alpha = \frac{\mathcal{R}}{C}$	Relative capacity
$\delta(d_i, d_j) = [\delta_x, \delta_y]$	Least displacement from d_i to d_j
Δ	Maximum Tolerable Delay
$\Gamma^{\Pi}(\delta_x, \delta_y, d_c)$	Traffic generated by all pair of nodes with least displacement $[\delta_x, \delta_y]$ that flows through d_c
$\lambda^{\Pi}_{d_k}$	Average traffic arrival rate to d_k according to Π
λ^{Π}_{l}	Average traffic rate through link l according to Π
λ^{Π}_{q}	Average traffic rate to queue q according to Π
μ	Average service rate
$\nu(d_k)$	Next hop at d_k
$\varphi^n(d_k)$	Set of neighbors of d_k
$\varphi^l(d_k)$	Set of links connected to d_k
$\pi_{d_k} = \{p_1, p_2, \ldots, p_{n(d_k)}\}$	Forwarding probabilities from d_k to any of its neighbors
Π_S	Stationary load distribution
Π	Routing algorithm
τ_0	Average loss probability threshold
ϕ	Shape of constrained spreading algorithm
ρ	Utilization factor
$\tau(d_i, d_j)$	Delay that a packet experiences to travel from d_i to d_j
ζ	UEP trade-off between side and central distortions

CSMA	Carrier Sense Multiple Access
CTS	Clear To Send
DBF	Distributed Bellman-Ford
DGT	Data Gathering Tree
JSCC	Joint Source Channel Coding
MAC	Media Access Control
MD	Multiple Description
MDSQ	Multiple Description Scalar Quantization
MST	Minimum Spanning Tree
pdf	Probability Density Function
PDF	Probability Distribution Function
RTS	Request To Send
SPR	Shortest Path Region
SPT	Shortest Path Tree
TSP	Traveling Salesman Problem
UEP	Unequal Error Protection
UTD	Uniform data gathering Traffic Distribution algorithm
UTD-Q	Uniform data gathering Traffic Distribution algorithm - finite Queues
UTD-SP	Uniform data gathering Traffic Distribution algorithm - Shortest Path
SD	Single Description

Chapter 1

Introduction

1.1 The emergence of sensor networks

Recent advances in computing technology and wireless communications are enabling the emergence of small and inexpensive devices incorporating communication, processing, and sensor functionalities. Some of these devices are commercially available for a low price [66]. Although we are still far from truly inexpensive devices, the decreasing price of wireless hardware is contributing to the proliferation of sensor networks for monitoring and control purposes: a large quantity of these devices is deployed in the field to create a densely distributed network of embedded signal sensors, processors and controllers.

One of the main reasons for the current rapid development of sensor networks is the potential of its applications and its relevance in various research fields. Sensor networks applications range from important societal issues such as environmental and habitat monitoring, traffic control, emergency scenarios, and health care, to economical issues such as production control and structure monitoring [27; 41; 16; 28]. Sensor networks have also a great potential as a research tool in experimental sciences: They facilitate the acquisition, processing, and interpretation of data that with the current centralized measurement systems would be very difficult and expensive. In addition to this, sensor networks allow data harvesting in scenarios of difficult access or in adverse environments, and at spatial densities that are much finer than with previous approaches.

However, the development of a working large scale sensor network still requires solutions to a number of technical and theoretical challenges, due mainly to the constraints imposed by the wireless sensor devices: Common devices used in sensor networks are generally very limited in power, communication bandwidth, processing capabilities, and storage capacity. Consequently, these devices present a high degree of unreliability, and information loss as well as temporary failures are common in the network.

In view of all these particular features, sensor networks require a new paradigm for communications: we need new tools (theories, heuristics, designs) specifically conceived for sensor networks. Of particular interest for this thesis is the routing problem, that is, moving data among different network locations.

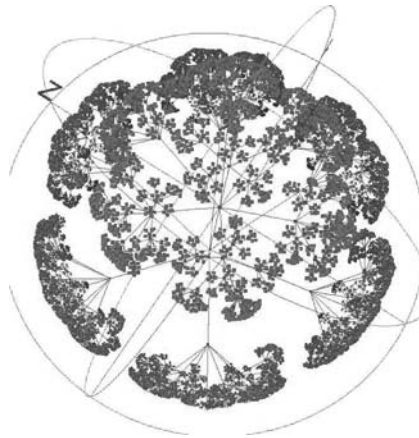


Figure 1.1: Nodes and links associated to a particular ISP. Note the hierarchical approach to deal with a large number of clients. (Source: `http://www.caida.org/`).

1.2 A new communication paradigm

The task of a sensor network typically consists of measuring a given physical phenomena, generating some information about the phenomena, and transmitting it to one or more base stations [1]. These base stations gather all the information generated in the network to analyze it, store it, and eventually take some actions. The problem of routing information in sensor networks is very challenging mainly due to the limited capabilities of the devices that compose these networks: limited amount of storage, processing power, and energy resources.

Traditional routing algorithms, developed usually for small size networks and/or more powerful nodes, become prohibitively complex in terms of both communication and computational complexity. For instance, due to the large number of devices in a sensor network, it may be not possible to build and maintain a global addressing scheme.

Thus, traditional IP-based protocols may not be applicable directly to sensor networks. Furthermore, scale issues have been dealt so far with by means of hierarchical routing. The most common approach to group a large client community is setting up a hierarchy with different hierarchical levels, where the clients are located at the bottom level. When any client is unable to route one packet, it is redirected to the next hierarchy level node. This hierarchical routing leads to a tree structure. On example of this tree structure is shown in Figure 1.1, which shows nodes and links for a particular ISP.

This hierarchical routing is clearly not suitable in the context of sensor networks. First, this routing scheme defines one single shortest path between any pair of nodes in the topology. If a node on this path fails, that particular source/destination pair would remain disconnected for an amount of time that may not be acceptable for some applications. Second, nodes and

links in high levels of the hierarchy have to sustain most of the network traffic. In networks where all the nodes have the same capabilities, nodes high in the hierarchy constitute the network bottleneck and a single point of failure.

In addition, the lack of reliability of sensor networks makes the routing problem even harder: the information sent within the network can be lost and any previously established route can break down due to node or communication failures. The high degree of unreliability of the individual devices, combined with the large number of devices strongly calls for multipath routing techniques, i.e., techniques in which data flows simultaneously along multiple routes. However, achieving multipath routing in sensor networks is a difficult task. First, searching a large space of possible routes (derived from having a large number of nodes with high density) may prove computationally prohibitive for low complexity devices. Second, with temporary node failures that result in routes being created and destroyed all the time, the communication overhead required to maintain an accurate picture of available routes might be prohibitive.

Most of the algorithms that have been developed in the past are not fully decentralized, nor fully self-organized, nor fully scalable. In this new scenario, the communication and computational complexity involved in traditional solutions based on centralized schemes makes necessary the use of distributed solutions: we need distributed ways of sensing, processing, and communicating.

This thesis concentrates on distributed routing algorithms, where routing decisions at each node are only based on local information: That is, information a) nodes has access to directly, b) nodes can compute by exchanging messages with its nearest neighbors *only*, and c) carried by each packet. We analyze the routing problem for common traffic scenarios encountered in sensor networks. For each of these scenarios, we study capacity limits and derive constructive routing strategies that achieve this capacity.

We design routing algorithms to deal with common constraints imposed by sensor networks. Particularly, we study packet losses caused by the limited storage capacity of sensor nodes and propose routing algorithms that minimize overflow losses. The analysis and design of routing algorithms for finite buffer networks requires solving the associated queueing network problem for which there are not analytical solutions even for simple cases [10]. In practice, several approximations are used to make the analysis tractable. Most of these approximations are based on Jackson's independence assumption, which works well under low rate but degrades rapidly as the rate increases. We propose a new approximation model that captures the dependence between the queue distributions of different nodes and allows to design simple routing algorithms that minimize overflow losses. We investigate also the problem of routing in unreliable networks with frequent temporary node failures under different failure models. We propose distributed routing algorithms that maximize the information rate transmitted in the network.

In unreliable networks such as sensor networks, specially designed coding techniques can increase the amount of *useful* information that is transmitted in the network. By jointly optimizing the source coding and the routing algorithms, we can significantly increase the information rate transmitted by the nodes. Particularly, we investigate the interaction of the source coding and the routing mechanisms in two common scenarios in sensor networks: data gathering and real time signal transmissions.

1.3 Related work

Sensor networks have attracted many research efforts over the past few years [61; 32], and many new algorithms have been proposed for the routing problem [2]. Almost every routing protocol can be classified according to the purpose they serve such as energy conservation, traffic balance, robustness, and throughput maximization.

For energy conservation, routing algorithms that equalize nodes energy and distribute the traffic evenly maximize network lifetime [35]. Upadhayay et al. observed that a balanced distribution of traffic has a greater impact on system performance than the adaptivity or efficiency of the algorithm [71] and presented a minimal fully adaptive routing algorithm that creates a balanced and symmetric traffic load in the network. The problem of load distribution was also studied by Hajeck [30].

Multipath routing techniques have been found to be a good strategy under the unreliable conditions of sensor networks to increase the robustness against failures [67; 59; 58]. The principle of these algorithms consists in flowing data simultaneously along multiple routes. Routing using multiple paths has also been studied in the context of high-speed networks [44], where it has been proposed as a way of reducing queueing delays in a manner analogous to adaptive routing [47], of dealing with transmission errors and of dealing with system failures [7; 5]. Parallel multiple route computations have been proposed as a mechanism to provide Quality of Service (QoS) in ad-hoc networks [14].

The usual deployment of devices into the sensed area frequently consists of a regular structure that results into a lattice sensor network, or a perturbation of it [22]. Capacity analysis of regular lattice networks has been addressed by various researchers [40; 50; 48]. Using graph cut methods, Sun and Modiano [69] investigated the spare capacity, which is the capacity needed on each link to recover from a link or node failure, in an NxN torus topology under a uniform all-to-all traffic model. Routing in lattice networks has been thoroughly studied in the context of distributed parallel computation [18], where Maxemchuk [43] analyzed various routing schemes through simulation.

Previous works that analyze the finite buffer limitation at the nodes are mainly based on Jackson's independence assumption [10]: the queue associate to each node in the network is analyzed independently of other queues. Harchol-Balter and Black [31] considered the problem of determining the distribution on the queue sizes induced by a routing algorithm in lattice networks. They reduced the problem into a product-form Jackson queue network and analyze it using standard queueing theory techniques. Mitzenmacher [49] approximated the system by an equivalent Jackson network with constant service time queues. He provided bounds on the average delay and the average number of packets for square lattices.

To reduce the complexity of the queueing network analysis, all the aforementioned papers make an independence approximation and consider the analysis of just one queue (node). We propose a new analysis that captures the dependence between the distributions on the queue size of different nodes. Using this analysis, we provide simple rules to design routing algorithms that maximize throughput for limited buffer nodes.

Leighton [40] analyzed the performance of simple routing algorithms in square grid and torus networks. Based on probabilistic reasoning, he provided bounds on the tail of the delay and queue size distributions. This analysis requires that the queue policy is "further first" instead of first-in first-out.

In deflection routing [43; 13], packets are forwarded to their preferred route (for instance, a shortest path) and when there is no storage available for a packet on the path to its destina-

tion, the packet is deflected to another path. Deflection routing exploits network diversity by reducing the number of times a packet is deflected. These routing techniques are especially suitable for ultra fast networks since packet buffering is completely avoided.

Finite buffers have also been considered in the study of scheduling algorithms for packet transmission in regular topologies under a uniform all-to-all traffic model [70; 73]. Sun and Modiano [70] showed that giving more priority to those packets with the shortest distance in number of hops to its destination achieves the highest throughput. Varvarigos and Bertsekas [73] evaluated the throughput of two different packet scheduling policies and observed that small buffer sizes can achieve throughput close to that of the infinite buffer case. In our work, packet scheduling is fixed (FIFO) and the focus is on analyzing optimal routing algorithms that achieves the maximum throughput. Particularly, we show that the performance of different routing algorithms differs significantly when the buffers are limited and that the required buffer to achieve a throughput close to that of the infinite buffer size reduces with the network size.

Neely, Rohrs and Modiano [53; 54] presented equivalent models for multi-stage tree networks of deterministic service time queues that reduce the analysis of tree network to the analysis of a simpler two-stage equivalent model.

For random wireless networks, Gupta and Kumar [29] studied the transport capacity and concluded that the total end-to-end capacity per node is $\mathcal{O}\left(1/\sqrt{N}\right)$, where N is the number of nodes. Duarte-Melo and Liu [17] studied the capacity and scalability issues of the data-gathering communication model in a wireless sensor network, giving upper bounds as well as constructive lower bounds by assuming that there exists a schedule that determines what subset of nodes can transmit simultaneously during which time slot. However, the problem of deriving optimal channel access schedules for multihop networks is NP-complete [4; 19]. Even sub-optimal schedules are highly non trivial to generate, especially in a decentralized manner. Under the same assumptions, Marco et al. [42] investigated the capability of large-scale sensor networks to measure and transport a two-dimensional field, where the quality of the reconstructed field is limited by the ability of the encoder to compress the data to a rate less that the capacity of the network. They concluded that as the density increases, any data compression scheme is insufficient to transport the required amount of data for a given quality.

The problem of designing a Media Access Control (MAC) for wireless networks to achieve proportional fairness of media access that adapts to different traffic requirements have been studied in [75; 51]. Nandagopal et al. showed that proportional fairness can be achieved without explicit global coordination by a backoff algorithm that optimizes a local utility function. Woo and Culler [75] proposed an adaptive rate control scheme that allows fair bandwidth allocation to the infrastructure for all nodes in a multihop network while being energy efficient. Mergen and Tong [48] analyzed the effects of the MAC on the network capacity. Particularly, they studied the effect of multi-packet reception capability on the capacity of wireless networks with regular structures under the uniform traffic model. For this scenario, they presented an optimal routing algorithms and a MAC scheme achieving the maximum capacity.

Many network communication problems do not allow to use the separation theorem, thus joint source channel coding (JSCC) can bring substantial improvements. McCanne et al. considered the use of JSCC in the context of multicast packet video [45; 46]. Alasti et al. [3] studied the use of JSCC in networks with congestion problem. They investigated the problem of a simple network represented by a set of parallel queues with congestion problems and

showed that double description coding significantly improves the overall average end-to-end distortion at high network loads compared to single description coding systems.

The first theoretical results in multiple description coding were provided by El Gamal, Cover and Ozarow [20; 55] for the case of Gaussian source, mean squared error distortion and two descriptions. An achievable region for the binary symmetric source with many descriptions was derived in [74]. Achievable rate regions of the multiple description problem with more than two descriptions have been determined for the *symmetric* case by Pradhan, Puri and Ramchandran [62; 64]. Vaishampayan proposed in [72] a simple procedure for designing multiple descriptions scalar quantizers with remarkably good asymptotic properties. For an excellent tutorial on multiple description coding refer to [25].

1.4 Thesis outline and contributions

In the complex distributed scenario of sensor networks, there are several interesting topics to investigate, spanning from traditional signal processing such as information coding, to communication and information theory issues such as transmission protocols and capacity bounds. Among all these, this thesis focuses on: (a) data routing, that is, efficient and computationally simple decentralized algorithms to move data among different network locations, and (b) information coding, that is, how sensors code the observations and how this interacts with routing.

In Chapter 2, we start by analyzing the point-to-point *routing* problem, that is, the problem of moving data from one single source to one single destination within the network. To overcome the size and complexity limitations of sensor networks, we propose a completely different approach to the traditional routing algorithms: we formalize the routing problem as a problem of constructing suitably constrained random walks on random lattices. We specify a desired stationary load distribution in the network and then define a distributed algorithm for computing the local parameter of a random walk that induces such distribution. To the best of our knowledge this is the first piece of work that deals with the routing of messages without any notion of discovering / maintaining / repairing explicitly described routes.

A natural extension to this scenario is to consider multiple source and/or multiple destinations, which is the object of Chapter 3. We start by studying routing problems on regular lattice graphs. Particularly, we focus on the analysis and design of routing algorithms that maximize the throughput per node for different communication patterns representing different network functionalities. For each case, we establish the fundamental limits of transmission capacity, and provide optimal routing algorithms for which the rate per node is equal to this maximum transmission capacity.

One of the important limitations of sensor networks is the constrained buffer space available on the nodes for the temporary storage of packets [75]. We show in Chapter 4 that the performance of different routing algorithms differs significantly when finite buffers are considered. In Chapter 4, we focus on the problem of routing in networks where nodes have small buffers and design routing algorithms that minimize overflow losses. This requires solving the queueing problem associated to the network and computing the distribution on the queue size at the nodes. We propose alternative approximation models to the usual Jackson's Theorem to obtain a more accurate distribution on the queue size at the nodes. Using this approximation, we analyze and design optimal routing algorithms for finite buffer networks that maximize the throughput per node. We also show that the required buffer to achieve a

throughput close to that of the infinite buffer size reduces with the network size.

Another limitation of sensor networks is the high degree of unreliability of the individual devices that compose these networks, which gives rise to frequent temporary failures. The object of Chapter 5 is to investigate the effect of the unreliability on the routing problem. We show that achieving robust communications and maximizing the achievable rate per node are incompatible goals: while robust communications require the use of as many paths as possible between the source and the destination, maximizing the rate per node requires using only a few of the available paths. We propose the use of a particular combination of two routing algorithms, the first one being optimal when there are no node failures at all and the second one being appropriate when the probability of node failure is high. The combination of these two routing algorithms defines a family of randomized routing algorithms, each of them being suitable for a given probability of node failure.

Most of the routing algorithms proposed in the previous chapters rely in one way or another on the regular topology of the network model considered. Motivated by the insights gained in lattice networks, we study in Chapter 6 the problem of routing in random topology networks. We start by studying network capacity and optimal routing algorithms for both infinite and finite buffers at the nodes for a collision-free network model. Even if random networks are very different in nature from grid networks, we show that similar principles can be applicable to analyze the buffer occupancy distribution at the nodes. Using approximation models, we study routing algorithms that minimize the packet overflow and consequently, maximize the achievable rate. We show that the problem of finding the optimal routing algorithm that achieves network capacity is an NP-hard problem. We propose an approximation algorithm to minimize overflow losses that achieves a maximum rate three times higher on average than the rate achieved with the usual shortest path routing algorithm. Finally, we discuss the wireless case, where packet transmissions among different nodes can interfere and result in a packet collision. We show that the overflow loss process is quite similar in both collision-free and wireless models, and propose a simple method to reduce overflow losses that consists in adapting the backoff window size linearly with the distance to the base station. This mechanism allows to significantly decrease overflow losses in the network.

Chapters 2 and 5 show that multipath routing is a good strategy in unreliable networks: by exploiting the space diversity of the network and making the data flow along multiple routes, we are able to overcome node failures and reduce the unpredictability of sensor networks. In Chapter 7, we show that we can exploit this multipath property even further by using an appropriate source coding mechanism. We investigate the interaction of the routing mechanism and source coding techniques such as Multiple Description (MD) coding. Particularly, we study the use of MD coding in two different scenarios: real time data transmission and central data gathering in a network with constrained buffers. First, in a real time data transmission scenario, we analytically compute the distortion at the receiver for a simple network model and show that the combination of multiple description coding and multipath routing performs significantly better than the usual single path and single description scheme. Second, we investigate the optimal transmission strategy in a sensor network performing a central data gathering and show that, depending on the network conditions, there exists an optimal number of descriptions that minimizes distortion. In this chapter we also consider the practical aspects of using MD coding in large sensor networks where the set of available paths between any source-destination pair is large. While most of the previous work studies MD coding from a rate-distortion point of view, we concentrate on the practical aspects of implementing an MD coding scheme in a sensor network. The results obtained in both

real time data and constrained buffer networks indicate the benefit of combining MD coding techniques and multipath routing in large and dense unreliable networks.

We conclude in Chapter 8 with a summary and a discussion about future research.

1.5 Performance Evaluation

Given the difficulty in performing actual measurements in real wireless networks, we evaluated all our algorithms through simulations. Instead of specifying the simulation environment for each graph, in this section we review the programs we use in this thesis.

For simulations in lattice networks (Chapters 2, 3, 4, and 5) we have created a simple simulator in C capable of dealing with sensor networks of large sizes.

To drive the simulations in random and wireless networks (Chapter 6), we use NAB (Network in A Box) [33], a network simulator targeted at wireless ad hoc and sensor networks that was designed for the purpose of simulating large-scale networks. NAB is written in Ocaml, a high-level yet highly efficient programming language.

To establish accurate simulation results, we perform multiple independent replications of each experiment [57] and compute the mean, the variance, and the confidence intervals. For the mean and the variance, we use the following formulas:

$$\overline{X} = \frac{1}{k}\sum_{i=1}^{k} X_i,$$

$$S^2 = \frac{1}{k-1}\sum_{i=1}^{k}(X_i - \overline{X})^2,$$

and to compute the $(1-\gamma)*100$ confidence intervals we use:

$$ci = \frac{\sqcup^{-1}\left(1-\frac{\gamma}{2}\right)S}{\sqrt{k}},$$

where k is the number of replications and $\sqcup^{-1}$ is the inverse cumulative function of the student distribution with ∞ degrees of freedom. All experimental curves obtained through simulation include the 95% confident intervals (if not specified otherwise) for each data point. In the case of graphs where the variance of each data point is very small, confidence intervals have been omitted for clarity.

Chapter 2

Point-to-Point Multipath Routing Algorithms for Large Scale Sensor Networks

2.1 Introduction

Implementing a basic packet forwarding service on a network of sensors is a challenging problem. The first difficulty to overcome is the high degree of unreliability: the individual nodes that compose sensor networks are subject to frequent temporary failures. This unreliability becomes more critical in the case of large sensor networks. With many error-prone nodes on any individual route, there is a high probability that some node along any particular route will fail. Moreover, traditional routing algorithms usually developed for smaller size networks become prohibitively complex in terms of computation (searching a large space of possible routes) and communication (a node failure/recovery translates into routes being destroyed/created).

However, we show in this chapter that working with large networks allows us to overcome the lack of reliability inherent in sensor networks. The high degree of unreliability of the network combined with a large number of nodes, strongly calls for multipath routing techniques capable of exploiting the path diversity present in the network. How to provide such multipath routing in large networks is the fundamental problem we address in this chapter.

To deal with large networks, we focus on decentralized algorithms, that is, algorithms based only on local information. Routing decisions at each node are only based on information nodes have about the state of the network, and possibly, on information carried by the packet. In this way, the complexity of the routing algorithm is independent of the size of the network. Our main insight is the use of randomized algorithms to implement multipath routing at essentially the cost of having each node implement independent routing decisions plus some minimal overhead.

This work is mainly inspired by Kelly's modeling of interacting particle systems using random walks [36]. At a microscopic level, the behavior of a particle can be described using a random walk model. At a macroscopic level, the same behavior is described in terms of global quantities such as temperature, voltage, etc. If we identify particles with packets and

the network with the medium, the routing problem consists of finding the best way to drive packets from one location to another. More specifically, we formalize the routing problem as a problem of constructing suitable constrained random walks on graphs. The target is to achieve a desired stationary load distribution in the network. Once this desirable load distribution is specified, we define a distributed algorithm for computing the local parameters of a random walk that induces such distribution.

We construct constrained random walks on a particular family of random graphs that we have chosen as an abstraction for the behavior of large sensor networks. Such random walks define a large class of algorithms for each node in the network to execute, so as to route packets to any destination. We denote by $\varphi^n(d_k) = \{u_1, \ldots, u_{n(d_k)}\}$ the set of neighbors of d_k and by $n(d_k) = |\varphi^n(d_k)|$ its cardinality. Let $\pi_{d_k} = \{p_1, p_2, \ldots, p_{n(d_k)}\}$ be real numbers such that $p_i \geq 0$ and $\sum_i p_i = 1$ (a pdf on the neighbors of d_k). When a packet reaches node d_k, the next hop is chosen by tossing a die whose i-th face occurs with probability p_i, and the packet is forwarded over the link (d_k, u_i). By making different assumptions on the topology of the underlying network, on its dynamics, and on constraints imposed on the local pdfs, we are able to explore a large and structured space of possible routing schemes.

In this chapter, we compute the local parameters of random walks (i.e., the π_{d_k}) with all the desired decentralization properties mentioned above that induces a suitable load distribution for different network abstractions. To the best of our knowledge, this is the first piece of work that dealt with the routing of messages without any notion of discovering / maintaining / repairing explicitly described routes.

This chapter is organized as follows. In Section 2.2, we study the routing problem in a square lattice and derive the local parameters of a random walk that achieves the most uniform traffic distribution in the network. In Section 2.3, we consider the construction of constrained random walks on a less structured graphs, namely a random lattice where a random subset of nodes has been deleted. We show that in this random model, we can still define suitable random walks with a certain load balancing property. Finally, in Section 2.4, we study the problem of routing in random dynamic graphs by considering a time-varying version of the static random network of Section 2.3.

2.2 Constrained random walks on regular static graphs

We start by designing suitable constrained random walks for a static graph with a regular structure, namely the square lattice (Figure 2.1). This model is simple enough to allow us to obtain simple closed form expressions for the sought distributions, yet at the same time it is rich enough to allow us to explore issues related to scalability versus numbers of nodes.

2.2.1 Network model

We consider a square lattice of size $N_1 \times N_1$ nodes (Figure 2.1) described by the graph $G(V, E)$. Given a set S, let $|S|$ denote the cardinality of the set S. The $N_1 \times N_1$ square lattice contains $|V| = N_1^2 = N$ nodes and $|E| = 2N_1(N_1 - 1)$ links.

A node $d_k \in V$, $k \in [0, N-1]$, can be characterized by its local coordinates $[i_k, j_k]$ in the lattice. We denote by $h(d_k, d_l)$ the shortest distance in hops between d_k and d_l, that is: $h(d_k, d_l) = |i_k - i_l| + |j_k - j_l|$. We denote by $h_e(d_k)$ the shortest distance in hops from d_k to the nearest node on the boundary of the lattice. The l-th *diagonal* of G is the set of all nodes $[i_k, j_k] \in V$ such that $i_k + j_k = l$, and is denoted by $Diag(l)$. By a slight abuse of

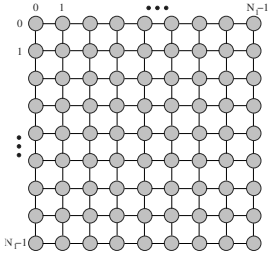


Figure 2.1: Square Lattice of size $N_1 \times N_1$. A node d_k is characterized by its coordinates $[i_k, j_k]$ in the lattice.

notation, we denote by $Diag(d_k)$ the set of all nodes that belong to the same diagonal as d_k, that is, $Diag(i_k + j_k)$. The size of a diagonal is given by $|Diag(d_k)|$. We assume that nodes are not aware of their geographic position in the network, i.e., nodes do not know their local coordinates. A link $l = (d_k, d_l) \in E$ represents a communication channel between nodes d_k and d_l. If there exists a link $l = (d_k, d_l) \in E$, we say that d_l and d_k are neighbors. We assume that all links have the same characteristics in terms of length and capacity.

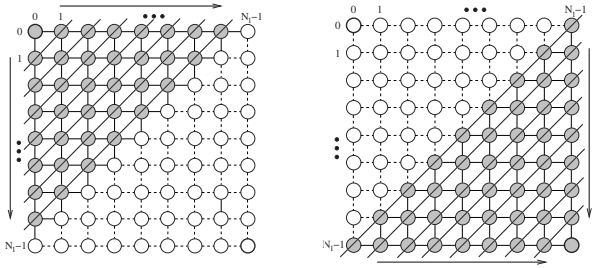
We assume that packets are injected at the source $d_s = [0, 0]$, and must travel hop by hop to the destination node $d_d = [N_1 - 1, N_1 - 1]$. Any interior node $[i_k, j_k]$ has 4 neighbors: $\varphi^n(d_k) = \{[i_k - 1, j_k],\ [i_k, j_k - 1],\ [i_k + 1, j_k],\ [i_k, j_k + 1]\}$. The first two are closer to the source, the latter two are closer to the destination. A completely general random walk on this lattice is specified by giving four numbers: $\pi_{d_k}[i_k - 1, j_k]$, $\pi_{d_k}[i_k, j_k - 1]$, $\pi_{d_k}[i_k + 1, j_k]$, and $\pi_{d_k}[i_k, j_k + 1]$, for all $d_k \in V$ (except at the boundaries, where the number of neighbors is smaller). We derive now the local parameters of the random walk that induces a suitable load distribution in the network.

2.2.2 Local parameters of the random walk

To effectively exploit whatever degree of *route diversity* the network provides, we require a certain "load balancing" condition on the stationary distribution Π_S induced by the constrained random walk. The load balancing condition we impose can be formulated as follows: if two nodes are located at the same distance from the source, then these nodes must carry the same traffic. That is, given $d_k = [i_k, j_k]$ and $d_l = [i_l, j_l]$ such that $i_k + j_k = i_l + j_l$, then $\Pi_S(d_k) = \Pi_S(d_l)$. In other words, all nodes belonging to a particular diagonal $Diag(l)$ share the load evenly. To ensure the avoidance of livelock conditions, we also impose the following condition on the random walks: if for some destination node d_d, we have some $d_l \in \varphi^n(d_k)$ such that $h(d_l, d_d) > h(d_k, d_d)$, then $\pi_{d_k}(d_l) = 0$. That is, we consider only "shortest-path" routing algorithms.

We compute now the local parameters π_{d_k} of the random walk that induces the desired load distribution. We begin by dividing the network into two regions (Figure 2.2):

- In the *expansion* stage, packets move across diagonals with an increasing number of



(a) Expansion Stage (b) Compression Stage

Figure 2.2: We divide the network into two stages: (a) *Expansion* stage: increasing number of nodes per diagonal (b) *Compression* stage: decreasing number of nodes per diagonal.

nodes and consequently, the density of packets per node decreases (Figure 2.2(a)).

- In the *compression* stage, packets move across diagonals with a decreasing number of nodes and consequently, the density of packets per node increases (Figure 2.2(b)).

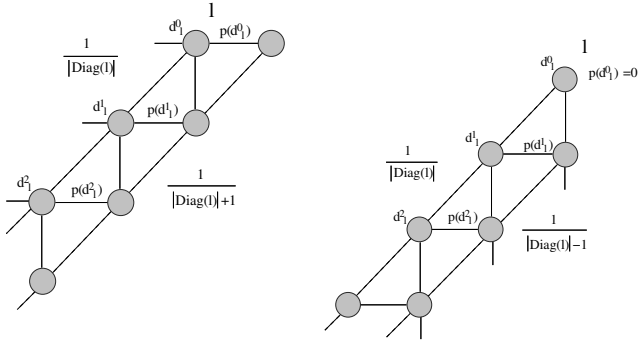
In the initial *expansion* stage, the number of nodes among which to spread the packet load increases, and therefore, the optimal load per node must decrease. After crossing the longest diagonal (corresponding to nodes with coordinates $i_k + j + k = N_1 - 1$) and entering the *compression* stage, the number of nodes on diagonals starts decreasing, and therefore the load per node must increase. It is straight forward to see that if $i_k + j_k < N_1 - 1$, d_k belongs to the expansion stage, while if $i_k + j_k \geq N_1 - 1$, d_k belongs to the compression stage

Note that other than boundary nodes, any node $d_k = [i_k, j_k]$ has exactly two neighbors with a shorter distance to the destination than his own, whose coordinates are $[i_k + 1, j_k]$ and $[i_k, j_k + 1]$. Therefore, in this particular topology, and under the shortest path condition, a random walk is defined by a single number p, the probability of choosing one of these two links. By convention, we define $p(d_k)$ to be the probability of d_k forwarding a packet to the neighbor that is closer to the boundary of the lattice ($1 - p(d_k)$ being the probability of forwarding to the other). Note that $p(d_k)$ characterizes completely the random walk.

Proposition 2.1 *The local parameters of the constrained random walk that achieves a uniform distribution on diagonals are given by:*

$$p(d_k) = \begin{cases} \frac{|Diag(d_k)| - h_e(d_k)}{|Diag(d_k)|+1}, & \text{if } i_k + j_k < N_1 - 1 \quad \textit{(expansion stage)}, \\ \frac{h_e(d_k)}{|Diag(d_k)|-1}, & \text{if } i_k + j_k \geq N_1 - 1 \quad \textit{(compression stage)}. \end{cases} \tag{2.1}$$

Intuitively, during the expansion stage we are imposing a penalty to use the nodes close to the straight line joining the source and the destination (large h_e) that can be used by a large



(a) Expansion (b) Compression

Figure 2.3: Forwarding probabilities.

number of nodes. Similarly, in order to avoid the "edge effect", during the compression stage we impose a penalty to those paths closer to the edge (small h_e).

Proof: The proof in both cases proceeds by induction on the diagonals. Consider first the expansion stage. The first diagonal corresponds to the source node d_s, and hence $|Diag(d_s)| = 1$ and $h_e(d_s) = 0$. It follows from (2.1) that $p(d_s) = 1/2 = (1 - p(d_s))$, achieving a uniform packet distribution over the second diagonal.

Let d_l^e be a node belonging to diagonal $Diag(l)$ at a distance e from the nearest node on the boundary, that is, $h_e(d_l^e) = e$. Assume (by induction) that we have already a uniform packet distribution over $Diag(l)$, and so the fractional load supported by each node is $1/|Diag(l)|$. The next diagonal has $|Diag(l)| + 1$ nodes, and the fractional load we want to achieve is $\frac{1}{|Diag(l)|+1}$. The situation is depicted in Figure 2.3(a). Therefore, for the set of nodes d_l^e with $0 \leq e \leq \frac{N_+}{2}$, their corresponding probability $p(d_l^e)$ has to satisfy:

$$\begin{aligned}
\frac{1}{|Diag(l)|} \cdot p(d_l^0) &= \frac{1}{|Diag(l)|+1}, \\
\frac{1}{|Diag(l)|} \cdot p(d_l^1) + \frac{1}{|Diag(l)|} \cdot (1 - (d_l^0)) &= \frac{1}{|Diag(l)|+1}, \\
&\cdots \\
\frac{1}{|Diag(l)|} \cdot p(d_l^e) + \frac{1}{|Diag(l)|} \cdot (1 - p(d_l^{e-1})) &= \frac{1}{|Diag(l)|+1}.
\end{aligned}$$

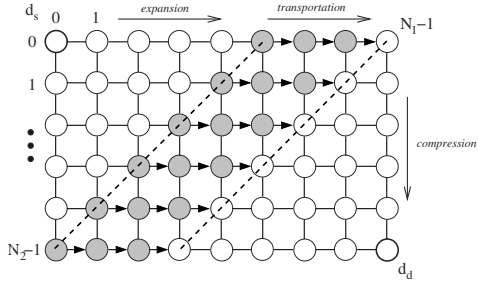


Figure 2.4: If the network is not square, there is also a *transportation* stage where the number of nodes in the diagonals remains constant.

Solving this system of equations, for any node d_k we obtain:

$$p(d_k) = \frac{|Diag(d_k)| - h_e(d_k)}{|Diag(d_k)| + 1}.$$

We proceed similarly for the compression stage: suppose we have the load uniformly distributed over a diagonal $Diag(l)$. The fractional load supported by each node is $1/|Diag(l)|$. The next diagonal has $|Diag(l)| + 1$ nodes, and the fractional load we want to achieve is $\frac{1}{|Diag(l)|-1}$. This situation is depicted in Figure. 2.3(b). For nodes d_l^e at distance $0 \leq e \leq \frac{N_1}{2}$ from the boundary, the corresponding probability $p(d_l^e)$ satisfies:

$$\begin{aligned}
p(d_l^0) &= 0, \\
\frac{1}{|Diag(l)|} \cdot p(d_l^1) + \frac{1}{|Diag(l)|} \cdot (1 - p(d_l^0)) &= \frac{1}{|Diag(l)| - 1}, \\
&\cdots \\
\frac{1}{|Diag(l)|} \cdot p(d_l^e) + \frac{1}{|Diag(l)|} \cdot (1 - p(d_l^{e-1})) &= \frac{1}{|Diag(l)| - 1}.
\end{aligned}$$

Finally, solving this system of equations yields:

$$p(d_k) = \frac{h_e(d_k)}{|Diag(d_k)| - 1}.$$

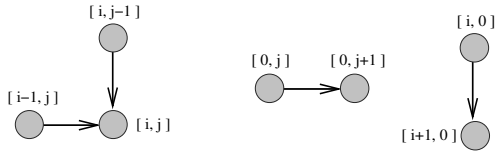
□

Note that if the source node d_s and destination node d_d are such that $|i_d - i_s| \neq |j_d - j_s|$, i.e. the network is not a square (Figure 2.4), we can still use the same probability distribution derived above. The only difference is that, besides the *expansion* and *compression* stages, there is also a *transportation* phase where the number of nodes in the diagonals remains constant. During the *transportation* phase, the distribution has a trivial solution: $p(d_k) = 1$ or 0 depending whether $|i_d - i_s| > |j_d - j_s|$, or $|i_d - i_s| < |j_d - j_s|$. This is illustrated in Figure 2.4.

2.2.3 Distributed computation of the local parameters

To compute the local parameters of the random walk (2.1), nodes require knowing their own lattice coordinates. However, we assume that nodes are not aware of their absolute position in the network. We can only expect that each node comes equipped with a unique identifier and that position information is discovered via communication among the nodes. In this section, we give a distributed algorithm to compute the lattice coordinates of the nodes.

We assume that nodes know their neighbor or neighbors whose distance to a given node d_i is smaller than its own. These nodes can be obtained by executing a distributed shortest path algorithm like Bellman-Ford [10]. To compute the lattice coordinates of any node d_k, it is important to note the following: the coordinates of d_k are uniquely identified by the coordinates of its neighbors whose distance to the source d_s is smaller than its own. This is illustrated in Figure 2.5.



(a) Two neighbors (b) One single neighbor

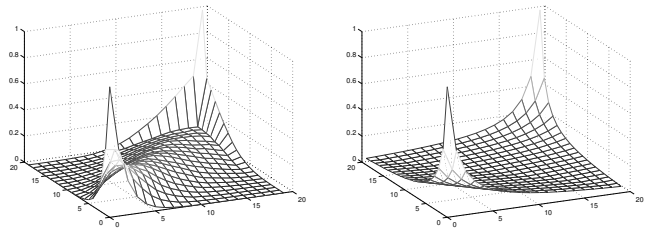
Figure 2.5: Recursive computation of network coordinates.

If d_k has two neighbors d_1 and d_2 with coordinates $[i_1, j_1]$ and $[i_2, j_2]$ whose distance from the source is smaller than its own (Figure2.5(a)), then the coordinates of d_k are given by $[i_k, j_k] = [\max(i_1, i_2), \max(j_1, j_2)]$. On the other hand, if d_k has only one neighbor d_1 with smaller distance to the source (Figure2.5(b)), this neighbor must have coordinates of the form either $[i_1, 0]$ or $[0, j_1]$, and then the coordinates of d_k are either $[i_1 + 1, 0]$ or $[0, j_1 + 1]$. Finally, the decision of which node is $[1, 0]$ and which node is $[0, 1]$ is made arbitrarily by the source. Once we have the geometrical location of any node, the distributed routing algorithm is trivial by using (2.1).

2.2.4 Simulation results

For illustration purposes, we compare the load distributions induced in the network by two different constrained random walks. First, we consider a simple random walk where, to decide which of the two feasible neighbors on a next hop to pick at each node, we flip a fair coin. Second, we use the constrained random walk defined by (2.1). Both distributions are shown in Figure 2.6. The simulation consists of 10000 messages transmitted in a network of size 20×20 injected at $d_s = [0, 0]$ with destination $d_d = [19, 19]$.

As expected, when using forwarding probabilities which are independent of the network location, most of the traffic is confined around the straight line joining the source and the destination. Using a constrained random walk with the local parameters defined above, we



(a) (b)

Figure 2.6: Load distribution in the network. a) Random walk based on tossing a fair coin to decide the next hop. b) Random walk using the local parameters derived in this section. The x and y axis represent the lattice coordinates of a node. The z-axis represents the number of packets carried by node $[x, y]$ normalized such that diagonals sum up to 1.

assign higher probability to nodes away from this main line during the expansion stage and to nodes close to it during the compression stage, such that the load distribution in the network becomes more uniform. Of course, at the source and destination there is still an unavoidable accumulation, but this is inherent in the bottlenecks at the source and destination.

Figure 2.7 shows the traffic distribution across the central diagonal $Diag(N_1 - 1)$, that is, across the set of nodes $[N_1 - i, i]$ for $i \in [0, \ldots, N_1 - 1]$, in a 20×20 lattice network. While using the constrained random walk with the local parameters defined above we achieve a uniform load distribution, the random walk based on tossing a fair coin load clearly overloads central nodes.

2.3 Constrained random walks on irregular static graphs

2.3.1 Network model

We now turn our attention to the construction of constrained random walks on a less structured graph than the square lattice. We introduce a new graph model that takes into account network irregularities by deleting a random subset of nodes from the square lattice, but in such a way that the resulting graph remains connected (Figure 2.8).

In the previous section, we derived the local parameters of a constrained random walk that induces a uniform traffic distribution on the diagonals of the square lattice. For this new network model, the concept of diagonal is still valid. However, achieving an exact load balancing as defined above will not be possible in general. This can be illustrated through the example depicted in Figure 2.8: if the load is uniformly distributed in $Diag(1)$, that is, $(1/3, 1/3, 1/3)$, then a uneven load of $(2/3, 1/3)$ will result in $Diag(2)$. Equivalently,

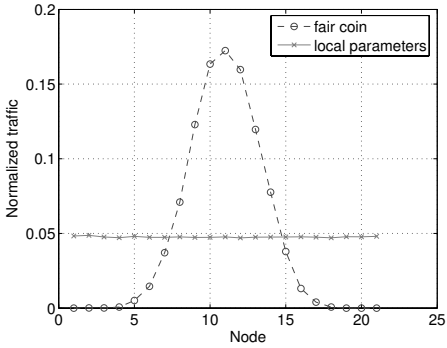


Figure 2.7: Load distribution in the central diagonal $Diag(N_1 - 1)$ induced by both random walks in a 20×20 regular lattice network.

to ensure an even load balance of $(1/2, 1/2)$ in $Diag(2)$, a uneven load of $(1/4, 1/4, 1/2)$ is required on $Diag(1)$. Therefore, a uniform load distribution simultaneously across all diagonals is not feasible.

The question that arises immediately is what is the load distribution that a random walk should induce in such an irregular graph. Even if we cannot achieve uniform load across diagonals, we still want to distribute load as fairly as possible between all nodes belonging to the same diagonal. It turns out that we can still define suitable random walks: If the number of routes is large, we can say something about the statistical distribution of where a packet will lie after h hops. Therefore, by choosing appropriate local parameters for the random walk, we should still be able to control the shape of this distribution, and steer it to one which is, if not exactly, at least approximately uniform across diagonals. Building on this intuition, we elaborate next on how to accomplish this.

2.3.2 A generalization of the lattice coordinates

It is important to note that the distributed algorithm we presented in Section 2.2.3 to compute the local coordinates of all nodes does not work in the case of irregular graphs: if any of the neighbors of a node is missing, we will not be able to identify the coordinates of the node. For this reason, we first introduce a generalization of the concept of lattice coordinates. We associate to each node d_k a new label $\{s_k, t_k\}$, where the first component s_k is the number of different available routes between d_k and the source, and the second component t_k is the number of different available routes between d_k and the destination. We say that two routes are different if they differ in at least one node (note that this is much weaker than requiring the routes to be disjoint, i.e., that all but the first and last nodes are different). These new labels can be easily computed in a distributed way: the number of different routes from a node to the source/destination can be obtained as the sum of the number of routes to the

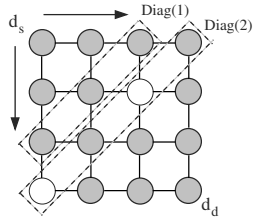


Figure 2.8: Irregular graph: a random subset of nodes has been removed from a square lattice. In general, a uniform load distribution across all diagonals simultaneously is not feasible.

source/destination from its neighbors whose distance to the source/destination is smaller than its own. Note that these new labels introduce a certain dependency with the network state and does not relate specifically to the location of the node.

The notions of expansion and compression stages can now be naturally generalized in this irregular network model using the new defined labels. We say that a node d_k labeled as $\{s_k, t_k\}$ belongs to the expansion stage when $s_k < t_k$. Equivalently, we say that the node belongs to the compression stage if $s_k \geq t_k$. We can also reformulate the forwarding probabilities of the random walk in terms of the new labels. Consider a node with label $\{s_k, t_k\}$ that has at most two neighbors to which it could forward data with labels $\{s_1, t_1\}$ and $\{s_2, t_2\}$. Then, the probability $p(d_k)$ of forwarding a packet to the node $\{s_1, t_1\}$ is defined as:

$$p(d_k) = \begin{cases} \frac{s_2}{s_1+s_2} & \text{if } s_k < t_k \quad \textit{(expansion stage)}, \\ \frac{t_1}{t_1+t_2} & \text{if } s_k \geq t_k \quad \textit{(compression stage)}. \end{cases} \tag{2.2}$$

Intuitively, during the expansion stage we make the forwarding probability to a given node inversely proportional to the number of routes between this node and the source. This is because, if we were successful in spreading the load evenly in earlier stages, then we would expect the load received by any node to increase with the number of routes from the source to that node (more routes mean more ways in which a packet could reach this node). During the compression stage, we make the forwarding probability to a given node proportional to the number of routes between this node and the destination. Since nodes can distribute the incoming load between all the available routes toward the destination, we make the supported load proportional to this number of routes.

2.3.3 Equivalence with lattice coordinates in the regular lattice

An important property of the labels defined above is that, if applied in the context of the regular square lattice, the computed forwarding probabilities are identical to those calculated in Section 2.2. It is in this sense that we call these labels a generalization of the lattice coordinates.

Proposition 2.2 *In the context of the regular square lattice, the local parameters of the con-*

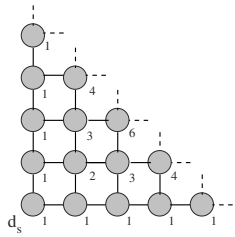


Figure 2.9: Pascal's Triangle: each node d_k is labeled with the number of different routes s_k to the source d_s.

strained random walk computed using the lattice coordinates of the nodes (2.1) and the node labels (2.2) are identical.

Proof: Consider the square lattice shown in Figure 2.1. The number of routes s_k from any node d_k in the expansion stage toward the source d_s represents an instance of the Pascal's Triangle problem, and hence s_k is given by the following combinatorial number:

$$s_k = \begin{pmatrix} |Diag(d_k)| - 1 \\ h_e(d_K) \end{pmatrix} \tag{2.3}$$

This is illustrated in Figure 2.9.

If now we substitute (2.3) in the local parameters of the random walk (2.2), in the case of a complete square lattice we obtain:

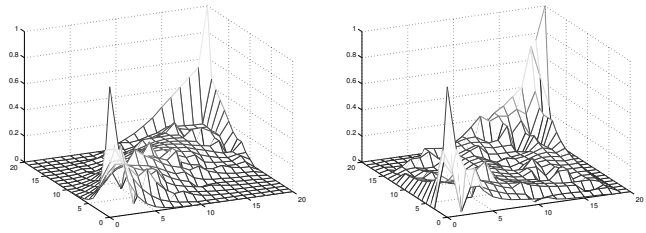
$$\begin{aligned} p(d_k) &= \tfrac{s_2}{s_1+s_2}, \\ &= \frac{\begin{pmatrix} |Diag(d_k)| \\ h_e(d_k)+1 \end{pmatrix}}{\begin{pmatrix} |Diag(d_k)| \\ h_e(d_k)+1 \end{pmatrix} + \begin{pmatrix} |Diag(d_k)| \\ h_e(d_k) \end{pmatrix}}, \\ &= \tfrac{|Diag(d_k)|-h_e(d_k)}{|Diag(d_k)|+1}. \end{aligned}$$

Similarly, for nodes in the compression stage, the number of joint paths toward the destination also forms a Pascal's Triangle. Substituting into (2.2) we obtain:

$$p(d_k) = \frac{t_1}{t_1+t_2} = \frac{h_e(d_k)}{|Diag(d_k)|-1}. \tag{2.4}$$

□

Therefore, labels based on the number of routes can be viewed as a generalization of the lattice coordinates, given that the packet forwarding probabilities induced are the same. Moreover, using these labels, the computation of the local parameters of the random walk (2.2) can be generalized for the case of non regular networks where nodes have more than



(a) (b)

Figure 2.10: Load distribution in the network. a) Random walk based on tossing a fair coin when there are two feasible neighbors. b) Constrained random walks using the local parameters computed in Section. 2.3.2. The x and y axis represents the network position of a node. The z-axis represents the number of packets carried by node $[x, y]$ normalized such that diagonals sum up to 1.

two feasible neighbors. If we denote by $p(d_k, d_i)$ the probability that d_k forwards a packet to d_i, where $d_i \in \varphi^n(d_k)$, then:

$$p(d_k, d_i) = \begin{cases} \frac{\sum_{d_j \in \varphi^n(d_k),\ d_j \neq d_i} s_j}{(|\varphi^n(d_k)|-1)\sum_{d_j \in \varphi^n(d_k)} s_j} & \text{if } s_k < t_k \quad \textit{(expansion stage)}, \\ \frac{t_i}{\sum_{d_j \in \varphi^n(d_k)} s_j} & \text{if } s_k \geq t_k \quad \textit{(compression stage)}. \end{cases} \tag{2.5}$$

2.3.4 Routing algorithm for the irregular lattice

In the case of a lattice with missing nodes, the algorithm for setting forwarding probabilities must be modified according to the number of neighbors that are on. Assume a given node $[i, j]$ and its two possible next hops, $[i + 1, j]$, and $[i, j + 1]$. We have three different cases depending on whether these nodes are on or off: if both nodes are on, we simply assign probabilities using (2.2). If there is only one neighbor on, we assign probability 1 to the only active node. Finally, if none of them are on, we assign probability 1 to a neighbor whose distance to the destination is strictly smaller than the distance from the current node.

The basic idea of this algorithm is that, if the process for deleting nodes is homogeneous (in the sense that the probability of having a node missing is independent of the location of the node, as it is the case when nodes are deleted independently), then we expect that load imbalances created by forwarding data to a single neighbor will in some cases cancel out. This issue is explored via simulations next.

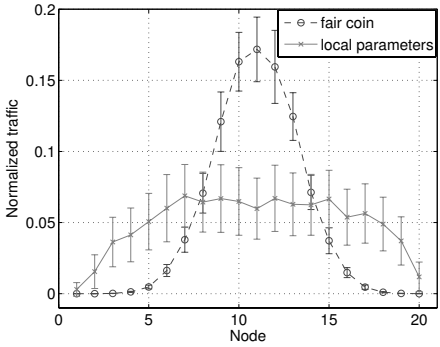


Figure 2.11: Average load distribution in the central diagonal $Diag(N_1 - 1)$ of a 20×20 random lattice network induced by both random walks for 1000 different random networks.

2.3.5 Simulation results

For illustration purposes, we compare the traffic distribution generated by a random walk based on tossing a fair coin to decide which of the two feasible neighbors on a next hop to pick with a constrained random walk whose local parameters are given in (2.2). Figure 2.10 shows the traffic distribution generated by both random walks in one irregular 20×20 square lattice where each node is ON with probability 0.95 and OFF with probability 0.05. The simulation consists of 10000 messages transmitted from node $[0, 0]$ to node $[19, 19]$.

Note that the constrained random walk with the proposed local parameters achieves a marked improvement in terms of load balancing, especially in comparison to the scheme based on tossing a fair coin. Note also that now loads on the diagonals are not uniform any more, although these plots suggest that the imbalance is not severe.

Figure 2.11 shows the average traffic distribution in the central diagonal $Diag(N_1 - 1)$ of a 20×20 random lattice network for 1000 different random lattices. Note that in the random case, the traffic distribution induced by the constrained random walk with the local parameters defined above is not perfectly uniform across the diagonal. The reason for this non-uniformity is the different number of possible paths that goes through a particular node of the diagonal. Nevertheless, this distribution is clearly more uniform that the distribution induced by the random walk based on tossing a fair coin, which overloads central nodes.

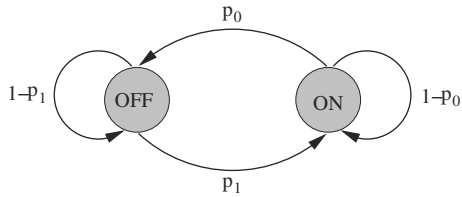


Figure 2.12: Nodes switch between ON and OFF states following a Markov chain model. Transitions are independent among nodes.

2.4 Constrained random walks on random dynamic graphs

2.4.1 Network model

Finally, we study the problem of routing in random dynamic graphs by considering a time-varying version of the static irregular lattice of the previous section: instead of deleting a set of nodes from the square lattice, we make nodes switch between ON and OFF states at discrete interval following a Markov rule as shown in Figure 2.12, where the time interval is equal to the time it takes to transmit one packet from one node to the following. We assume that transitions are independent among nodes. Note that under this Markov chain model across time, the stationary probabilities of a node to be ON or OFF are given by:

$$p_{ON} = \frac{p_1}{p_0 + p_1}, \tag{2.6}$$

$$p_{OFF} = \frac{p_0}{p_0 + p_1}. \tag{2.7}$$

2.4.2 Dynamic parameter computation

In Section 2.3, we presented a distributed labeling algorithm that adapts to the network topology which can be seen as a generalization over the lattice coordinates for irregular graphs. In the case of dynamic graphs, we adopt a dynamic version of this labeling algorithm: when a node changes state, this change will affect the labels of its neighbors (since the number of routes available from these nodes will change), which in turn will trigger changes to labels of node further apart. That is, upon a state transition in the network (a node recovery/failure), nodes will progressively update their labels according to the new graph. Using this dynamically updated labels, nodes compute the local parameters of the random walk using (2.2) as in the case of static graphs.

Two important aspects of this dynamic labeling algorithm is to understand how routing performance is affected by the delays in propagating information about updates of labels, and how sensitive this routing performance is to inaccuracies in the labels. As the network becomes more unpredictable (i.e. in which state transitions occur more often), labels are less accurate and consequently, the routing becomes less effective. We explore these aspects in the next section via simulations.

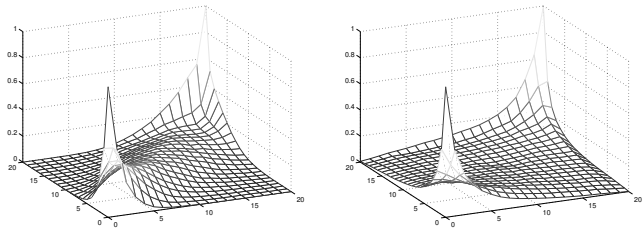


Figure 2.13: Load distribution in a dynamic network with transition probabilities $p_0 = 0.0125$ and $p_1 = 0.2375$ so that the stationary probability of a node to be OFF is $p_{OFF} = 0.05$. Left: random walk based on tossing a fair coin when there are two feasible neighbors. Right: constrained random walks using the local parameters computed in Section. 2.3.2. The x and y axis represents the network position of a node. The z-axis represents the number of packets carried by node $[x, y]$ normalized such that diagonals sum up to 1.

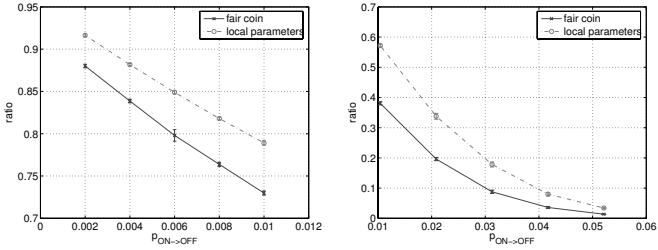
Finally, note that the routing problem in dynamic lattice networks under random failures will be studied in more detail in chapter 5.

2.4.3 Simulation results

As in previous sections, we compare first the traffic distribution generated by a random walk based on tossing a fair coin to decide which of the two feasible neighbors on a next hop to pick with the constrained random walk with the local parameters discussed above. Figure 2.13 shows the traffic distribution generated by both random walks in a dynamic irregular 20×20 square lattice with transition probabilities $p_0 = 0.0125$ and $p_1 = 0.2375$ so that the stationary probability of a node to be OFF is $p_{OFF} = 0.05$. The simulation consists of 10000 messages transmitted from node $[0, 0]$ to node $[19, 19]$.

Note that the constrained random walk with the proposed local parameters achieves a marked improvement in terms of load balancing. Interestingly, the load distributions achieved in the case of a dynamic random networks are much closer to uniform than those achieved in an irregular but static lattice. The reason is that, because of the ergodicity of the model considered for the network dynamics, the load distribution of the dynamic network is essentially the average of the load distributions of many irregular static random networks and it is this averaging effect what results in smoother, more balanced loads. For instance, the traffic distribution in the central diagonal $Diag(N_1 - 1)$ of dynamic networks is identical to the averaged traffic distribution in the central diagonal of irregular and static lattices.

Another important factor to measure the performance of the routing algorithm is the distribution of the time messages take to reach the destination. In the static case, this time is just the number of hops on a shortest route. However, in the context of dynamic networks,



(a) $p(OFF) = 0.01$ (b) $p(OFF) = 0.05$

Figure 2.14: Ratio of packets arriving to the destination without any additional delay as a function of the network *variability* in a 100×100 dynamic network with state probability $p(OFF) = 0.01$ and $p(OFF) = 0.05$.

this delay becomes random: as nodes go ON and OFF, the information about state transitions needs to propagate throughout the network, and this propagation takes time. Therefore, inaccurate state information can introduce randomness in transport delay in two forms. First, packets can be delayed at intermediate nodes if both $[i+1, j]$ and $[i, j+1]$ are OFF, and the current distance estimates from both $[i-1, j]$ and $[i, j-1]$ to destination are greater than that from $[i, j]$. In this case, a packet at $[i, j]$ waits a random amount of time until either the map of distances converges (and a new neighbor closer to destination can be identified), or until one of $[i+1, j]$ and $[i, j+1]$ turns ON again. Second, packets can also be delayed if both $[i+1, j]$ and $[i, j+1]$ are OFF, and at least one of the current distance estimates from either $[i-1, j]$ or $[i, j-1]$ to destination is smaller than from $[i, j]$. Note that both situations can arise only for short periods of time, while distance updates propagate.

For the same two random walks as before, we now compare the ratio of packets that arrive to the destination in the shortest time, i.e., the ratio of packets that does not get additional delay in the network. Figure 2.14 shows this ratio in a 100×100 dynamic network with stationary probabilities $p(OFF) = 0.01$ and $p(OFF) = 0.05$ as a function of p_0. Note that p_0 is an indicator of the *variability* of the network: higher values of p_0 imply frequent state transmissions and consequently, nodes often route packets based on inaccurate state information. Each simulation consists of 10000 messages transmitted from node $[0, 0]$ to node $[99, 99]$.

Note that in all cases, the random walk with the local parameters we defined exhibits the highest ratio of packets delivered without any additional delay. If a packet does not encounter a node with inaccurate information, it will arrive to the destination in the minimum number of hops. However, if a packet does encounter a node that recently underwent a state transition, it will likely get either delayed at that node, or misrouted. If state transmissions are unpredictable and uniformly distributed in all nodes, to minimize the number of packets being delayed at a given time, the best one can do is to distribute packets as balanced as

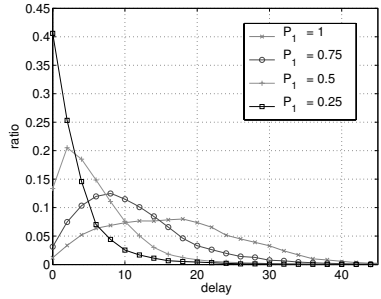


Figure 2.15: Transmission delay distributions as a function of the network *variability* in a 100×100 dynamic network with stationary probability $p(OFF) = 0.05$ and different transition probabilities p_0 and p_1.

possible in the network.

In Figure 2.15, we analyze the packet delay distribution induced by the random walk whose local parameter are discussed in Section 2.4.2 in a 100×100 dynamic network with state probability $p(OFF) = 0.05$ as a function of the network variability. Each simulation consists of 10000 messages transmitted from node $[0, 0]$ to node $[99, 99]$.

First note that if $p_0 = 0$, the graph is static, and hence all packets arrive to the destination without any additional delay. As the network becomes more unpredictable, i.e., as p_1 increases, state transitions are more frequent and less packets follow a shortest path due to inconsistent state routing. Consequently, the time distribution becomes wider.

2.5 Summary

In this chapter, we studied the problem of designing point-to-point routing algorithms for large scale sensor networks. First, we argued that complexity considerations make it natural to introduce an element of randomization in the problem formulation, and so we formulated the problem as one of defining suitable random walks on random dynamic graphs. Then, we presented random walk constructions in three different cases: a regular and static lattice, an irregular but still static lattice, and an irregular and dynamic lattice. For these three cases we designed distributed algorithms for computing the local parameters of the random walks that induced the most uniform load distribution in the network. Finally, we analyzed the performance of the proposed random walks via simulations. Using this routing formulation, we are able to route messages without explicitly discovering / maintaining / repairing routes.

A natural extension to this scenario is to consider multiple source and/or multiple destinations, which is the focus of the next chapter.

Chapter 3

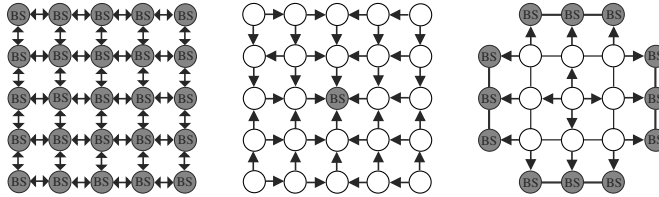
Multiple Sources and Destinations: Capacity Limits and Optimal Routing

3.1 Introduction

A natural extension to the point-to-point routing analyzed in previous chapter is to consider multiple sources and/or multiple destinations. Depending on the structure and goal of the network (monitoring, data collection, actuation), nodes exhibit different communication patterns. Particularly, we analyze three different communication models: uniform communication, central data gathering, and border data gathering. In uniform communication (Figure 3.1(a)), the probability of any node communicating to any other node in the network is the same for all pairs of nodes. It models a distributed control network where every node needs the information generated by all nodes in the network [26]. In central data gathering (Figure 3.1(b)), nodes send their data to one common node (base station) which collects the information generated in the network [34]. In border data gathering (Figure 3.1(c)), the information generated by all nodes is collected by the nodes located at the border of the network. This configuration models a situation that arises frequently in integrated devices: Nodes located on the borders can be easily connected to high-capacity transmission lines, while nodes inside the device are difficult to connect and can only communicate to neighbor nodes.

As in the previous chapter, we consider lattice networks, namely square lattice and torus lattice based networks. We choose these simple structures because they allow for a theoretical analysis while still being useful enough to incorporate all the important elements such as connectivity and scalability with respect to the size of the network. Moreover, lattice networks are widely used in regular settings like parallel computation [18], distributed control [15], satellite constellations [70], and wired circuits such as CMOS circuits and CCD-based devices [21]. Lattice networks are also known as grid [11] or mesh [68] networks.

We focus on the analysis and design of routing algorithms that maximize the throughput per node for the three network models described above. For each case, we establish the fundamental limits of transmission capacity, and then, we characterize and provide optimal routing algorithms achieving a rate per node equal to this maximum transmission capacity.



(a) Uniform communication (b) Central data gathering (c) Border data gathering

Figure 3.1: Communication models: a) uniform communication, b) central data gathering, and c) border data gathering.

We require the routing algorithms to have all the decentralization properties mentioned in the previous chapter. In addition, we require them to satisfy the property of being space invariant: the algorithm to route packets between any two nodes depends only on the relative position between them, not on their absolute positions. This is equivalent to assuming that nodes are not aware of their absolute position in the network.

The rest of the chapter is structured as follows. In Section 3.2, we introduce the network model and definitions. Section 3.3 studies the uniform communication model. We give capacity limits and design optimal routing algorithms for both torus and square lattices. In Section 3.4, we study the central data gathering model and in Section 3.5, the border data gathering. For both communication models we study capacity limits and derive constructive routing strategies that achieve this capacity.

3.2 Model and definitions

We assume that either the considered network is wired (e.g. a CMOS circuit) or if it is wireless, we assume contention is solved by the MAC layer. Thus, we abstract the wireless case as a graph with point-to-point links and transform the problem into a graph with nearest neighbor connectivity.

We consider graphs of size $N_1 \times N_1$ nodes (or vertices) that are either a square or torus lattice. The subscripts "s" and "t" denote the square and the torus lattices respectively. The square lattice (Figure 3.2(a)) is described by the graph $G_s(V, E_s)$ and the wrapped square or torus lattice (Figure 3.2(b)) by the graph $G_t(V, E_t)$. A torus lattice network is obtained from a square lattice network by adding some supplementary links between opposite nodes located at the border of the lattice. Figure 3.2 shows a $N_1 \times N_1$ square and a torus lattices for $N_1 = 5$.

The $N_1 \times N_1$ square lattice $G_s(V, E_s)$ contains $|V| = N$ nodes (or vertices) and $|E_s| = 2N_1(N_1 - 1)$ links (or arcs). The $N_1 \times N_1$ torus lattice $G_t(V, E_t)$ contains $|V| = N$ nodes and $|E_t| = 2N$ links.

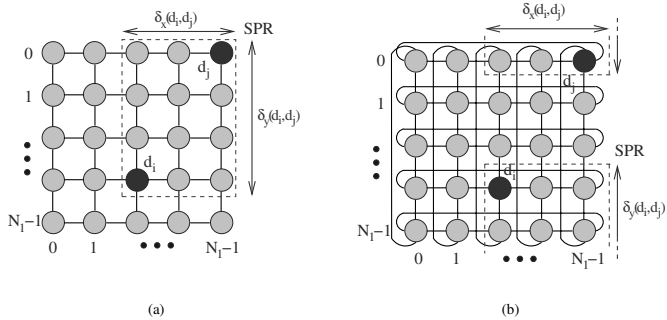


Figure 3.2: Network model and least displacement of nodes $\{d_i, d_j\}$ in (a) square and (b) torus lattices. The shortest path region SPR(d_i, d_j) between nodes d_i and d_j is delimited in both cases by a dashed square.

Every node $d_i \in [0, N-1]$ in the network can potentially be the source or the destination of a communication, as well as a relay for communications between any other pair of nodes. We assume that sources generate constant size packets with an average rate of $\mathcal{R}$ packets per time unit. We denote by $T(d_i, d_j)$ the probability of d_i communicating to d_j.

An arc or link $l \in E_{\{s,t\}}$ represents a communication channel between two nodes. In this work, we consider two cases for these communication channels, namely the half-duplex and the full-duplex case, depending upon whether both nodes may simultaneously transmit, or whether one must wait for the other to finish before starting transmission. In the case of half-duplex links, if two neighbor nodes want to use the same link, we assume that both have the same probability of capturing the link for a transmission.

The network model is similar to the one discussed in [70]. We assume that time is slotted and a one-hop transmission consumes one time slot. We denote by $\varphi^l(d_i)$ the set of links connected to the node d_i. The length of a path is defined as the number of links in that path. Moreover, we denote by $h(d_i, d_j)$ the length of the shortest path between nodes d_i and d_j. We define the *shortest path region* SPR(d_i, d_j) of a pair of nodes $\{d_i, d_j\}$ as the set of nodes that belong to any shortest path between d_i and d_j. For instance, SPR(d_i, d_j) in the square lattice is a rectangle with limiting corner vertices being d_i and d_j (Figure 3.2(a)).

For any pair of nodes $\{d_i, d_j\}$, we can view the lattice as an Euclidean plane map and consider d_j to be displaced from d_i along the X-Y Cartesian coordinates. We denote as $\delta_x(d_i, d_j) \in [-(N_1 - 1), N_1 - 1]$ and $\delta_y(d_i, d_j) \in [-(N_1 - 1), N_1 - 1]$ the relative displacements from d_i to d_j Figure 3.2(a)) and define the *least displacement* for these two nodes as:

$$\delta(d_i, d_j) = [\delta_x(d_i, d_j), \delta_y(d_i, d_j)]. \tag{3.1}$$

Because of the particular existing symmetry in the torus lattice, given two nodes $\{d_i, d_j\}$, there are several possible values for $\delta(d_i, d_j)$. We consider $\delta(d_i, d_j)$ to be the one with the

smallest norm (Figure 3.2(b)).

When packets arrive at a particular node or are generated by the node itself, they are placed into the buffer until the node has the opportunity to transmit them through the required link. We assume nodes have a unconstrained buffer for the temporary storage of packets.

A rate $\mathcal{R}$ is said to be *achievable* if the total number of packets in the network stays bounded over time.

Definition 3.1 *An N nodes network is said to have capacity $C_{\{s,t\}}^{\{u,cg,bg\}}(N)$ if any rate $\mathcal{R} = C_{\{s,t\}}^{\{u,cg,bg\}}(N) - \epsilon,\ \forall\ \epsilon > 0$, is achievable. Moreover, any rate $\mathcal{R} \geq C_{\{s,t\}}^{\{u,cg,bg\}}(N)$ is not achievable.*

The subscripts "s" and "t" indicates the network topology, square and torus lattices respectively. The superscripts "u", "cg", and "bg" denotes the traffic model, that is, uniform, central data gathering, or border data gathering respectively.

A routing algorithm Π defines how traffic flows between any source destination pair $\{d_i, d_j\}$. Shortest path routing algorithms are those where packets transmitted between any two nodes d_i, d_j can only be routed inside $\text{SPR}(d_i, d_j)$. We assume that routing algorithms are time invariant, that is, Π does not change over time. As in previous chapter, we assume that nodes are not aware of their absolute positions in the network.

Definition 3.2 *We say that a routing algorithm Π is space invariant if routing between any pair of nodes depends only on the relative position of the two corresponding nodes. That is, Π is space invariant if for all $\{d_i, d_j, d_k, d_l\} \in V$,*

$$\delta(d_i, d_j) = \delta(d_k, d_l) \ \rightarrow\ \Pi(d_i, d_j) = \Pi(d_k, d_l).$$

Definition 3.3 *A routing algorithm Π is said to have maximum achievable rate $\mathcal{R}_{sup}^{\Pi}(N)$ if any rate $\mathcal{R} = \mathcal{R}_{sup}^{\Pi}(N) - \epsilon,\ \ \forall\ \epsilon > 0$, is feasible in a network of N nodes using Π as routing algorithm. Moreover, any rate $\mathcal{R} \geq \mathcal{R}_{sup}^{\Pi}(N)$ is not feasible.*

We denote by $F^{\Pi}(d_i, d_j, d_k)$ the traffic generated at node d_i with destination node d_j that flows through node d_k according to a particular routing algorithm Π. Similarly, we denote by $\lambda_{d_k}^{\Pi}$ the average traffic arrival rate to node d_k according to Π:

$$\lambda_{d_k}^{\Pi} = \sum_{d_i \in V} \sum_{d_j \in V} T(d_i, d_j) F^{\Pi}(d_i, d_j, d_k). \tag{3.2}$$

By a slight abuse of notation, we denote by λ_l^{Π} the average traffic rate through link l according to a routing algorithm Π.

In the next sections, we study network capacity and routing algorithms that achieve the maximum $\mathcal{R}_{\text{sup}}^{\Pi}(N)$ for the three different communication models described above.

3.3 Uniform communication model

In the uniform communication model, the probability of any node communicating to any other node in the network is the same for all pairs of nodes, that is:

$$T(d_i, d_j) = \begin{cases} \frac{1}{N-1} & d_i \neq d_j, \\ 0 & d_i = d_j. \end{cases} \tag{3.3}$$

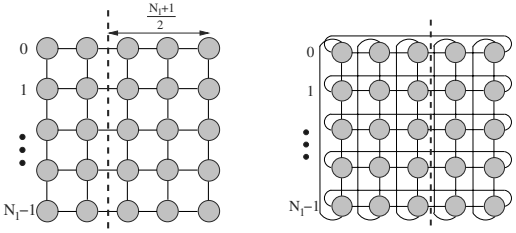


Figure 3.3: Bisections for a $N_1 \times N_1$ lattice network: Left: square grid, and Right: torus.

It models a distributed control network where every node needs the information generated by all nodes in the network [26]. We study now the network capacity and the optimal routing algorithm that achieve this upper bound.

3.3.1 Network capacity

Under the infinite buffer assumption, network capacity can be easily derived applying bisection arguments[1] [40; 50] to both torus and square lattices.

Lemma 3.1 *The network capacity $C^u_{\{s,t\}}(N)$ for the uniform communication model is given by:*

$$C^u_s(N) = \begin{cases} \frac{2\beta}{\sqrt{N}}\left(1-\frac{1}{N}\right), & \text{if } N \text{ is even}, \\ \frac{2\beta}{\sqrt{N}}, & \text{if } N \text{ is odd}, \end{cases} \tag{3.4}$$

$$C^u_t(N) = \begin{cases} \frac{4\beta}{\sqrt{N}}\left(1-\frac{1}{N}\right), & \text{if } N \text{ is even}, \\ \frac{4\beta}{\sqrt{N}}, & \text{if } N \text{ is odd}, \end{cases} \tag{3.5}$$

where β is equal to 1 for half-duplex links and 2 for full-duplex links.

The bisections that give these limits are shown in Figure 3.3. Note from (3.4) and (3.5) that, in both cases, the network capacity decreases with the square root of the total number of nodes, that is, with N_1. An alternative capacity proof that does not use a bisection argument can be found in appendix 3.A. As we will see in the next section, these upper bounds are actually tight and can be achieved by certain routing algorithms.

3.3.2 Optimal routing algorithms

Network capacity can indeed be achieved in both torus and square lattices by using suitable routing algorithms. In other words, $\max_{\Pi}\left\{\mathcal{R}^{\Pi}_{\text{sup}}(N)\right\} = C^u_{\{s,t\}}(N)$. The next proposition

[1] The bisection argument is based in the max-flow min-cut theorem which states that the maximal amount of a flow is equal to the capacity of a minimal cut.

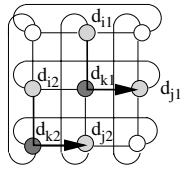


Figure 3.4: The source-destination pair $\{d_{i_1}, d_{j_1}\}$ generates traffic that flows across node d_{k_1} according to Π. If Π is space invariant, for any other node d_{k_2}, we can find another source-destination pair $\{d_{i_2}, d_{j_2}\}$ with the same least displacement as $\{d_{i_1}, d_{j_1}\}$ that generates exactly the same traffic across d_{k_2} as $\{d_{i_1}, d_{j_1}\}$ across d_{k_1}.

characterizes the class of shortest path routing algorithms that are optimal for torus lattice networks.

Theorem 3.1 *All shortest path space invariant routing algorithms achieve network capacity in a torus lattice network.*

$$\mathcal{R}^{\Pi}_{sup}(N) = C^u_t(N) \; \forall \, \Pi \in \{\textit{shortest path space invariant}\}.$$

Proof: Consider a space invariant routing algorithm Π. Given the structural periodicity of the torus, for every source-destination pair $\{d_{i1}, d_{j1}\}$ that generates traffic flowing across any particular node d_{k1}, there always exists another source-destination pair $\{d_{i2}, d_{j2}\}$ with the same least displacement as $\{d_{i1}, d_{j1}\}$ that generates exactly the same traffic flowing across some other node d_{k2} in the network (Figure 3.4). That is, $\forall\ d_{k2} \in V,\ \exists\{d_{i2}, d_{j2}\}$: $\{\delta(d_{i2}, d_{j2}) = \delta(d_{i1}, d_{j1})$ and $F^{\Pi}(d_{i2}, d_{j2}, d_{k2}) = F^{\Pi}(d_{i1}, d_{j1}, d_{k1})\}$.

Consequently, the average arrival rate to any node is constant:

$$\text{if } \Pi \in \{ \text{ space invariant } \}, \quad \lambda^{\Pi}_{d_k} = \lambda \quad \text{for all } d_k \in V. \tag{3.6}$$

Let $\overline{L}(N)$ be the average distance between a source and a destination for a given communication model described by $T(d_i, d_j)$:

$$\overline{L}(N) = \frac{1}{N} \sum_{d_i \in V} \sum_{d_j \in V} T(d_i, d_j) h(d_i, d_j).$$

Particularly, in a torus lattice under uniform traffic distribution (see Appendix 3.B):

$$\overline{L}(N) = \begin{cases} \frac{N\sqrt{N}}{2(N-1)}, & \text{if } N \text{ is even,} \\ \frac{1}{2}\sqrt{N}, & \text{if } N \text{ is odd.} \end{cases} \tag{3.7}$$

In the uniform communication model, all nodes generate packets at a constant rate $\mathcal{R}$. These packets take, on average, $\overline{L}(N)$ hops before reaching their destination. Therefore, the total traffic per unit of time generated in the network is given by $N\mathcal{R}\overline{L}(N)$. If Π is space invariant, all nodes have the same average rate (3.6) and the total traffic is uniformly distributed among all nodes. Therefore, the average arrival rate λ at any node is:

$$\lambda = \frac{N\mathcal{R}\overline{L}(N)}{N} = \mathcal{R}\overline{L}(N). \tag{3.8}$$

The maximum achievable rate $\mathcal{R}_{\text{sup}}^{\Pi}(N)$ at which nodes can transmit while keeping the queues stable and the number of packets in the network bounded, is obtained by applying the stability condition in the nodes, that is:

$$\lambda_{\text{sup}} = \mu_{\text{sup}} - \epsilon, \ \ \forall\, 0 < \epsilon < \mu_{\text{sup}}, \tag{3.9}$$

where λ_{sup} is the maximum average arrival rate per node and μ_{sup} is the maximum average number of packets that can be transmitted per unit of time. That is, $\mu_{\text{sup}} = 4$ for full-duplex communication channels and $\mu_{\text{sup}} = 2$ for half-duplex. Substituting (3.8) into (3.9), the maximum rate $\mathcal{R}_{\text{sup}}^{\Pi}(N)$ achieved by any space invariant routing algorithm Π is:

$$\mathcal{R}_{\text{sup}}^{\Pi}(N) = \frac{\mu_{\text{sup}}}{\overline{L}(N)} = \frac{2\beta}{\overline{L}(N)}. \tag{3.10}$$

Combining (3.7) and (3.10), $\mathcal{R}_{\text{sup}}^{\Pi}(N)$ is equal to the network capacity given in (3.5). □

As a consequence of Theorem 3.1, we have the following achievability result:

Corollary 3.2 *The network capacity $C_t^u(N)$ of a torus lattice network is equal to the upper bound given by (3.5).*

Theorem 3.1 says that, given the structural periodicity of the torus, the use of space invariant routing algorithms induces a uniform traffic distribution in the network that guarantees the maximum rate per node.

This uniform traffic distribution is not possible in the case of a square lattice network. Given the topology of a square lattice, as a node is located closer to the geographic center of the lattice, it belongs to the SPR of an increasing number of source-destination pairs. In the case of shortest path routing algorithms, this implies a higher traffic load.

Intuitively, the optimal routing algorithm has to avoid routing packets through the lattice center and promote as much as possible the distribution of traffic towards the borders of the lattice. In this way, we compensate the higher number of paths passing through the center of the lattice by enforcing a lower average traffic for these paths. This is the principle of *row-first* (*column-first*) routing [40]: nodes always route packets along the row (or column) in which they are located towards the destination node until they reach the destination's column (or row). Then, packets are sent along the destination's column (row) until they reach the destination node. This algorithm is illustrated in Figure 3.5. In the following, we prove that this simple algorithm is indeed optimal among all shortest path space invariant routing algorithms.

For the sake of simplicity, we restrict our analysis to the case of odd N. The analysis for even N is similar but more cumbersome, while essentially the same results hold. Notice also that since we are interested in large networks, this is not a limiting restriction.

Theorem 3.3 *For the square lattice network and the uniform communication model, the total average traffic $\lambda_{d_c}^{\Pi}$ that flows through the center node d_c for any space invariant routing algorithm Π, is lower bounded by:*

$$\lambda_{d_c}^{\Pi} \geq \mathcal{R}\sqrt{N}. \tag{3.11}$$

Proof: The prove is constructive: we show that this lower bound is actually tight and design a routing algorithm Π that achieves this lower bound. In this proof we make use of the concept of least displacement and the property of space invariant routing algorithms.

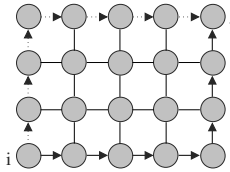


Figure 3.5: Row-first (solid lines) and column-first (dashed lines) routing algorithm. Nodes route packets using only the most external paths.

Let $\Gamma^{\Pi}(\delta_x, \delta_y, d_c)$ denote the traffic generated by all pair of nodes with a least displacement given by $[\delta_x, \delta_y]$ that flows through node d_c, that is,

$$\Gamma^{\Pi}(\delta_x, \delta_y, d_c) = \sum_{d_i \in V} \sum_{\substack{d_j \in V: \\ \delta(d_i, d_j) = [\delta_x, \delta_y]}} T(d_i, d_j) F^{\Pi}(d_i, d_j, d_c).$$

Given the symmetry of d_c, $\Gamma^{\Pi}(\delta_x, \delta_y, d_c)$ has the following properties:

$$\Gamma^{\Pi}(\delta_x, \delta_y, d_c) = \Gamma^{\Pi}(\delta_y, \delta_x, d_c). \tag{3.12}$$

$$\Gamma^{\Pi}(\delta_x, \delta_y, d_c) = \Gamma^{\Pi}(|\delta_y|, |\delta_x|, d_c). \tag{3.13}$$

The average traffic arrival rate to d_c can be obtained by summing over all possible least displacements in the network:

$$\lambda^{\Pi}_{d_c} = \sum_{\delta_x = -(N_1-1)}^{N_1-1} \sum_{\delta_y = -(N_1-1)}^{N_1-1} \Gamma^{\Pi}(\delta_x, \delta_y, d_c). \tag{3.14}$$

Using property (3.13), we can reduce the analysis of $\Gamma^{\Pi}(\delta_x, \delta_y, d_c)$ to only positive values of δ_x and δ_y. Using property (3.12), we can further reduce the analysis to the case $|\delta_x| \geq |\delta_y|$. That is,

$$\begin{aligned} \lambda^{\Pi}_{d_c} = 4 \sum_{\delta_x = 0}^{N_1-1} \Gamma^{\Pi}(\delta_x, \delta_x, d_c) \\ + 4 \sum_{\delta_x = 0}^{N_1-1} \Gamma^{\Pi}(\delta_x, 0, d_c) + 8 \sum_{\delta_x = 2}^{N_1-1} \sum_{\delta_y = 1}^{\delta_x - 1} \Gamma^{\Pi}(\delta_x, \delta_y, d_c). \end{aligned} \tag{3.15}$$

To derive now a lower bound for $\lambda^{\Pi}_{d_c}$, we can equivalently compute a lower bound for $\Gamma^{\Pi}(\delta_x, \delta_y, d_c)$ and apply (3.15). To compute $\Gamma^{\Pi}(\delta_x, \delta_y, d_c)$ we add the traffic contribution $F^{\Pi}(d_i, d_j, d_c)$ of all source-destination pairs $\{d_i, d_j\}$ such that $\delta(d_i, d_j) = [\delta_x, \delta_y]$. Instead of keeping d_c fixed and computing the traffic that goes through d_c for all $\{d_i, d_j\}$ such that $\delta(d_i, d_j) = [\delta_x, \delta_y]$, we can equivalently consider a fixed rectangle $R_c(\delta_x, \delta_y)$ of size $[\delta_x, \delta_y]$ and locate d_c in several positions. In other words, we determine the set $\mathcal{S}$ of relative positions

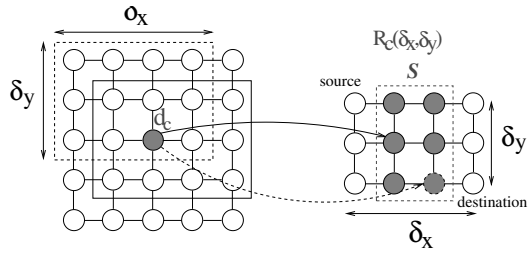


Figure 3.6: For all source-destination pairs $\{d_i, d_j\}$ such that $\delta(d_i, d_j) = [3, 2]$, we obtain the set $\mathcal{S}$ of relative positions of d_c in $R_c(\delta_x, \delta_y)$.

of d_c in $R_c(\delta_x, \delta_y)$ with respect to all source destination pairs $\{d_i, d_j\}$ such that $\delta(d_i, d_j) = (\delta_x, \delta_y)$. Then, the traffic that flows through d_c for any shortest path routing algorithm Π can be computed as the total traffic generated by Π in $\mathcal{S}$. Figure 3.6 shows an example for $[\delta_x, \delta_y] = [3, 2]$.

Once we obtain $\mathcal{S}$, we construct the routing policy Π that minimizes the total average traffic flowing through the set $\mathcal{S}$ or, equivalently, that minimizes $\Gamma^{\Pi}(\delta_x, \delta_x, d_c)$.

First note that if $\delta_y > \frac{N_1-1}{2}$, the set $\mathcal{S}$ has a vertical size smaller than δ_y and consequently, $\mathcal{S}$ does not fill completely any column of $R_c(\delta_x, \delta_y)$. We can therefore design a routing policy that uses only nodes in the set $R_c(\delta_x, \delta_y) \setminus \mathcal{S}$ and that generates no traffic in $\mathcal{S}$. The only routing policy that fulfills this condition for all $\delta_y > \frac{N_1-1}{2}$ consists of using only the two most external paths of SPR(d_i, d_j). Figure 3.7(a) illustrates this case. Therefore, for any Π, $\Gamma^{\Pi}(\delta_x, \delta_y, d_c) \geq 0$ for all δ_x.

If $\delta_y > \frac{N_1-1}{2}$ we distinguish between two cases. If $\delta_x > (N_1 - 1)/2$, $\mathcal{S}$ fills completely $N_1 - \delta_x$ columns of $R_c(\delta_x, \delta_y)$. Therefore, all routes between the source and the destination go through at least one node belonging to each of these $N_1 - \delta_x$ columns. Given that $T(d_i, d_j) = 1/(N-1)$ for all $d_i, d_j \in V,\ d_j \neq d_i$, the total traffic that goes through $\mathcal{S}$ is lower bounded by $\Gamma^{\Pi}(\delta_x, \delta_y, d_c) \geq \frac{\mathcal{R}}{N-1}(N_1 - \delta_x)$. Figure 3.7(b) illustrates this case. Note that there are many routing policies that achieve this lower bound, as for instance, the routing algorithm that uses only the two most external paths.

If $\delta_y \leq \frac{N_1-1}{2}$, $\mathcal{S}$ fills all the δ_x columns of $R_c(\delta_x, \delta_y)$ and any route between the source and the destination crosses at least $\delta_x + \delta_y$ nodes belonging to $\mathcal{S}$. That is, $\Gamma^{\Pi}(\delta_x, \delta_y, d_c) = \frac{\mathcal{R}}{N-1}(\delta_x + \delta_y))$. Note that we only consider the locations of d_c as a source of a transmission and not as a destination. Obviously, the packets that reach d_c and have d_c as final destination do not interfere with the traffic going through d_c, while the traffic generated at d_c itself does. Figure 3.7(c) illustrates this case.

Putting all three cases together, we have that $\Gamma^{\Pi}(\delta_x, \delta_y, d_c)$ is lower bounded as follows:

$$\Gamma^{\Pi}(\delta_x, \delta_y, d_c) \geq \begin{cases} \frac{\mathcal{R}}{N-1}(\delta_x + \delta_y) & \delta_x \leq \frac{N_1-1}{2}, \delta_y \leq \frac{N_1-1}{2}, \\ \frac{\mathcal{R}}{N-1}(N_1 - \delta_x) & \delta_x > \frac{N_1-1}{2}, \delta_y \leq \frac{N_1-1}{2}, \\ 0 & \text{otherwise}, \end{cases} \tag{3.16}$$

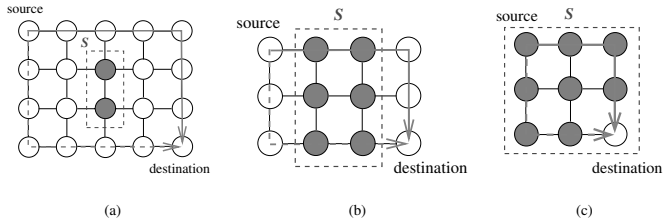


Figure 3.7: $R_c(\delta_x, \delta_x)$ for three possible cases in a 5×5 lattice network: (a) $\delta(i,j) = (4,3)$; since $\delta_y > \frac{N_1-1}{2}$, $\mathcal{S}$ does not fill completely any column of SPR(d_i, d_j). (b) $\delta(i,j) = (3,2)$; since $\delta_y \leq \frac{N_1-1}{2}$, $\mathcal{S}$ fills $N_1 - \delta_x$ columns. (c) $\delta(i,j) = (2,2)$; $\mathcal{S}$ fills all the δ_x columns. The arrows indicates two of the possible routing policies that generates the least possible traffic in $\mathcal{S}$.

and a routing algorithm Π that achieves minimization in the three cases consists in flowing data only through the most external paths.

Using (3.16) into (3.15), we bound the total traffic that flows through d_c as:

$$\lambda_{d_c}^{\Pi} \geq \left(\frac{\mathcal{R}}{N-1}\right)\Bigg\{4\sum_{\delta_x=0}^{(N_1-1)/2} 2\delta_x + 4\sum_{\delta_x=0}^{(N_1-1)/2}\delta_x + 4\sum_{\delta_x=(N_1+1)/2}^{N_1-1}(N_1-\delta_x) + 8\sum_{\delta_x=2}^{(N_1-1)/2}\sum_{\delta_y=1}^{\delta_x-1}(\delta_x+\delta_y) + 8\sum_{\delta_x=(N_1+1)/2}^{N_1-1}\sum_{\delta_y=1}^{(N_1-1)/2}(N_1-\delta_x)\Bigg\}.$$

and evaluating summations yields to $\lambda_{d_c}^{\Pi} \geq \mathcal{R}\sqrt{N}$. □

As a consequence of Theorem 3.3, we have the following three corollaries:

Corollary 3.4 *A shortest path space invariant routing algorithm achieves capacity in the uniform communication model only if the total average traffic $\lambda_{d_c}^{\Pi}$ that flows through the center node d_c is greater or equal to the total average traffic flowing through any other node d_x, that is:*

$$\lambda_{d_c}^{\Pi} \geq \lambda_{d_x}^{\Pi}, \quad \textit{for all } d_x \in V \setminus d_c.$$

Proof: The proof follows by contradiction. Suppose that a network capacity-achieving routing algorithm Π generates a traffic distribution where there exists a node d_x such that $\lambda_{d_x}^{\Pi} > \lambda_{d_c}^{\Pi}$. Then, from Theorem 3.3,

$$\lambda_{d_x}^{\Pi} > \mathcal{R}\sqrt{N}. \tag{3.17}$$

The maximum rate per node $\mathcal{R}^{\Pi}_{\text{sup}}(N)$ achieved by Π is obtained by imposing stability conditions for d_x, that is:

$$(\lambda^{\Pi}_{d_x})_{\text{sup}} = \mu_{\text{sup}} - \epsilon, \ \ \forall\, 0 < \epsilon < \mu_{\text{sup}}.$$

Particularizing (3.17) to the maximum rate, we have that $(\lambda^{\Pi}_{d_x})_{\text{sup}} > \mathcal{R}^{\Pi}_{\text{sup}}(N)\sqrt{N}$, and imposing the above stability condition:

$$\mathcal{R}^{\Pi}_{\text{sup}}(N) < \frac{(\lambda^{\Pi}_{d_c})_{\text{sup}}}{\sqrt{N}} = \frac{\mu_{\text{sup}} - \epsilon}{\sqrt{N}},$$

Recalling now the capacity formula of a square lattice network (3.4), we obtain:

$$\mathcal{R}^{\Pi}_{\text{sup}}(N) < C^u_s(N),$$

and therefore, Π does not reach capacity. □

Corollary 3.4 says that the factor that really limits the maximum achievable rate in the network is the amount of traffic routed through the center node d_c.

In the proof of Theorem 3.3, we have already characterized the set of routing policies that generates the minimum traffic in d_c. One of these policies consists in flowing data only along the two most external paths of the source and the destination SPR, that is, row-first (column-first) routing (Figure 3.5).

Theorem 3.5 *Row-first routing algorithm achieves capacity in a square lattice network.*

Proof: From Theorem 3.3, we know that $\lambda^{\text{r-f}}_{d_c} = \mathcal{R}\sqrt{N}$. It is easy to verify that the network traffic distribution induced by row-first routing is such that $\lambda^{\text{r-f}}_{d_c} > \lambda^{\text{r-f}}_{d_x}$ for all $d_x \in V \setminus d_c$.

In the case of infinite buffers, the maximum achievable rate $\mathcal{R}^{\text{r-f}}_{\text{sup}}(N)$ at which nodes can transmit information while keeping the number of packets in the network bounded, is obtained by applying the stability condition to the most loaded node d_c:

$$(\lambda^{\text{r-f}}_{d_c})_{\text{sup}} = \mu_{\text{sup}} - \epsilon, \ \ \forall\, 0 < \epsilon < \mu_{\text{sup}}. \tag{3.18}$$

Using now Theorem 3.3, and recalling the capacity formula of a square lattice network (3.4), we obtain:

$$\mathcal{R}^{\text{r-f}}_{\text{sup}}(N) = \frac{\mu_{\text{sup}}}{N} = C^u_s(N). \tag{3.19}$$

□

As a consequence, we have the following achievability result:

Corollary 3.6 *The network capacity $C^u_s(N)$ of a square lattice network is equal to the upper bound given by (3.4).*

The optimal routing strategy to achieve the maximum rate per node consists in making nodes use only the two most external shortest paths to route packets to any destination. Note that this contradicts the point-to-point routing algorithms derived in Chapter 2, where all shortest paths are used with equal probability. This suggests that there exists a trade-off between the maximum achieved rate and the robustness of the routing algorithms. This will be the topic of Chapter 5.

3.4 Central data gathering model

We now turn our attention to the central data gathering model, in which every node transmits information to a particular and previously designated node d_{BS} denoted *base station* that can be located anywhere in the network. This model corresponds to the case where one node (base station) collects the information generated by all the nodes in the network [34]. The communication matrix T is given by:

$$T(d_m, d_j) = \begin{cases} 1, & \text{if } d_j = d_{BS}, \\ 0, & \text{otherwise.} \end{cases}$$

Under the central data gathering model, routing in a torus is a particular case of routing in a square lattice. The reason is that for any base station location in a torus, the shortest path graph consists of a square lattice with the base station located in the center node. Therefore, in this section we need only to consider the square lattice. We apply the same type of analysis as in the uniform communication model. First, we bound the network capacity $C_s^{cg}(N)$ and then, we characterize the class of routing algorithms that achieve this capacity.

3.4.1 Network capacity

The following Lemma establishes an upper bound for the network capacity under the central data gathering model:

Lemma 3.2 *The network capacity $C_s^{cg}(N)$ for the central data gathering communication model in a square lattice is upper bounded as follows:*

$$C_s^{cg}(N) \leq \frac{|\varphi^l(d_{BS})|}{N-1}. \tag{3.20}$$

Proof: Since all the traffic from the network must reach d_{BS} using one of the links in the set $\varphi^l(d_{BS})$, the bottleneck of the network is clearly located in these links. Applying a bisection argument [40] to these links yields the result. □

Note that through the links that limit the capacity, the information flows only in one direction: from the inner nodes to d_{BS}. Therefore, the network capacity is equivalent for half-duplex and full-duplex links.

3.4.2 Optimal routing algorithms

Capacity achieving routing algorithms are characterized by the following Lemma:

Lemma 3.3 *A shortest path routing algorithm Π achieves capacity for a location of the base station d_{BS} if and only if the total arrival traffic to d_{BS} is uniformly distributed among the links in $\varphi^l(d_{BS})$. That is:*

$$\lambda_l^{\Pi} = \frac{\mathcal{R}(N-1)}{|\varphi^l(d_{BS})|}, \quad \textit{for all } l \in \varphi^l(d_{BS}). \tag{3.21}$$

Proof: Since all the arriving traffic to d_{BS} has to use one of the links in the set $\varphi^l(d_{BS})$,

$$\sum_{l \in \varphi^l(d_{BS})} \lambda_l^{\Pi} = \mathcal{R}(N-1). \tag{3.22}$$

In central data gathering, the most loaded link in the network obviously belongs to $\varphi^l(d_{BS})$. Applying the stability condition for these unitary capacity links:

$$\max_{l \in \varphi^l(d_{BS})} \lambda_l^{\Pi} = 1 - \epsilon, \ \forall \ 0 < \epsilon < 1. \tag{3.23}$$

Combining (3.22) and (3.23), the result follows. □

As a consequence of Lemma 3.3, we have the following achievability result:

Corollary 3.7 *The network capacity $C_s^{cg}(N)$ of a square lattice network for the central data gathering model is equal to the upper bound given by (3.20).*

Lemma 3.3 establishes that the only necessary and sufficient condition for a routing algorithm to achieve capacity is to uniformly distribute the traffic among the four input links to d_{BS}. Although there is a wide class of routing algorithms that satisfy this condition, we will show in next chapter that their performance is quite different when the buffers are constrained to be finite.

3.5 Border data gathering model

In this section, we apply similar tools to analyze routing in a substantially different communication model, namely, border data gathering. In border data gathering, all nodes located in the four edges of the square lattice act as base stations and inner nodes act as sources generating information that needs to be transmitted to any of these base stations without any specific mapping between source nodes and base stations (Figure 3.8). This configuration models a situation that arises frequently in integrated devices: Nodes located on the borders can be easily connected to high-capacity transmission lines, while nodes inside the device are difficult to connect and can only communicate to neighbor nodes. Note that several communication matrices are allowed. Obviously, this model can be considered only for the square lattice. We proceed as in previous sections: first, we compute the network capacity based on flow arguments. Then, we present the set of routing algorithms that achieve capacity.

3.5.1 Network capacity

The network capacity can be easily derived applying bisections augments:

Lemma 3.4 *The network capacity $C_s^{bg}(N)$ for the border data gathering communication model in a square lattice is upper bounded as follows:*

$$C_s^{bg}(N) \leq \frac{4}{\sqrt{N}-2}. \tag{3.24}$$

Proof: The bottleneck of the network is determined by the links connecting inner nodes to base stations. Therefore, the bisection that determines the network capacity is the one that separates the edge nodes from inner nodes (Figure 3.8). Noticing that there are $(N_1-2)^2$ nodes and $4(N_1-2)$ links through the cut, the result follows. □

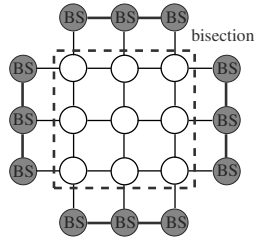


Figure 3.8: Border data gathering model and bisection that determines the network capacity.

Note also that on the links that limit the capacity, the information flows only in one direction: from the inner nodes to the edges. Therefore, the network capacity is equivalent for half-duplex and full-duplex links.

3.5.2 Optimal routing algorithms

Let S_{BS} denote the set of base stations in the network and $\varphi^l(S_{BS})$ the set of links that connect any base station with an inner node. Lemma 3.4 establishes that the maximum rate that nodes can transmit to S_{BS} is determined by the number of links in $\varphi^l(S_{BS})$. As a consequence, capacity achieving routing algorithms are characterized by the following Lemma:

Lemma 3.5 *A routing algorithm* Π *achieves capacity only if the total arrival traffic to* S_{BS} *is uniformly distributed among the links in* $\varphi^l(S_{BS})$*. That is:*

$$\lambda_l^{\Pi} = \frac{\mathcal{R}(\sqrt{N}-2)}{4}, \quad \textit{for all } l \in \varphi^l(S_{BS}). \tag{3.25}$$

The proof of this lemma is similar to the proof of Lemma 3.3.

First notice that no shortest path routing algorithm satisfies exactly this optimality condition. The shortest path routing algorithm that achieves the maximum rate $\mathcal{R}_{\text{sup}}^{\text{s-p}}(N)$ consists of distributing the traffic as uniformly as possible among the links in $\varphi^l(S_{BS})$. That is, when a node has more than one possible shortest path toward a base station, it distributes the load uniformly among these paths. This optimal shortest path routing is shown in Figure 3.9(a). The maximum rate $\mathcal{R}_{\text{sup}}^{\text{s-p}}(N)$ is limited by the most loaded links, that is, the links located in the middle of the four edges (Figure 3.9(a)):

$$\mathcal{R}_{\text{sup}}^{\text{s-p}}(N) = \frac{4}{2\sqrt{N}-5} < C_s^{bg}(N),$$

which is roughly one half of the network capacity $C_s^{bg}(N)$.

On the other hand, it can be shown that there exist multiple non-shortest path routing algorithms that achieve capacity. For instance, a simple strategy that achieves capacity consists

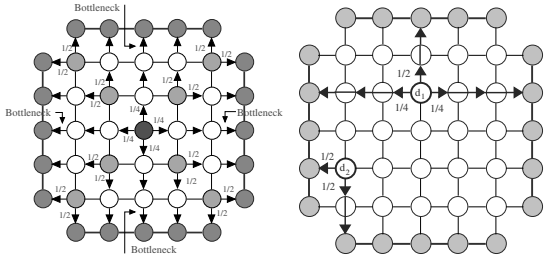


Figure 3.9: Routing algorithms for border data gathering. (a) Optimal shortest path routing. (b) Uniform border gathering: packets generated in a node are routed with equal probability to the two closest base stations located in the same row/column as the node (see d_2). If there are two base stations in the same column/row at the same distance (see d_1), we choose any of them with equal probability.

of the following: packets are always routed along the same row and column until they reach a base station, that is, packets do not turn. New packets generated in a node are routed with equal probability to the two closest base stations located in the same row or column as the node. If there are more than one base stations in the same row or column at the same distance, it chooses any of them with equal probability. We denote this routing algorithm as *uniform border gathering* and it is depicted in Figure 3.9(b). It is easy to verify that uniform border gathering satisfies the capacity condition (3.25) and therefore, $\mathcal{R}_{\text{sup}}^{\text{uniform}}(N) = C_s^{bg}(N)$.

Note that since in both routing algorithms, shortest path and uniform border gathering, packets flow only in one direction, the routing algorithms are equivalent for half-duplex and full-duplex links.

3.6 Summary

In this chapter, we considered the problem of multiple sources and/or destinations routing in lattice networks under three different communication models: uniform communication, central data gathering, and border data gathering. For each of these models, we studied capacity limits and derived constructive routing strategies that achieve this capacity.

In this chapter, we assumed that nodes have an infinite buffer for the temporary storage of packets. In the next chapter, we turn our attention to the case of finite buffers: we will show that the performance of these routing algorithms is quite different when the buffers are constrained to be finite. Indeed, we will show that the routing algorithms that achieve the maximum rate per node in the finite buffer case are different from the ones we presented in this chapter.

3.A Alternative capacity proof

The network capacity of an N nodes and M links network can be bounded as follows. Consider a data unit u, where $1 \leq u \leq \mathcal{R}NT$. Suppose it moves from the source d_i^u to the destination d_j^u following a shortest path of length $h(d_i^u, d_j^u)$. Then,

$$\sum_{u=1}^{\mathcal{R}NT} h(d_i^u, d_j^u) = \mathcal{R}NT\overline{L}. \tag{3.26}$$

Now we apply the condition that links are simplex, that is, can be used only by one node at the same time.

$$\sum_{u=1}^{\mathcal{R}NT} \sum_{s=1}^{h(d_i^u, d_j^u)} 1_{\{\text{The } s_{th} \text{ hop of } u \text{ is over link } i \text{ in slot } s\}} \leq C.$$

Summing over the links and the slots gives:

$$\sum_{u=1}^{\mathcal{R}NT} h(d_i^u, d_j^u) \leq MTC. \tag{3.27}$$

Combining equations (3.26) and (3.27):

$$\mathcal{R} \leq \frac{MC}{N\overline{L}}. \tag{3.28}$$

Substituting M, N and $\overline{L}$ in (3.28) yields the result.

3.B Average path length in a torus

In the uniform communication model, the traffic matrix T is defined as follows:

$$T(d_i, d_j) = \begin{cases} \frac{1}{N-1} & d_i \neq d_j \\ T(d_i, d_j) = 0 & d_i = d_j. \end{cases} \tag{3.29}$$

Then,

$$\overline{L} = \frac{1}{N} \sum_{d_i \in V} \frac{1}{N} \sum_{d_j \in V} T(d_i, d_j) h(d_i, d_j). \tag{3.30}$$

Given the structural homogeneity of the torus:

$$\overline{L} = \frac{1}{N-1} \sum_{d_j \in V, d_j \neq d_i}^{N} h(d_i, d_j), \quad \text{for any } d_i \in V. \tag{3.31}$$

The diameter d of a $N_1 \times N_1$ torus network is given by:

$$d = \begin{cases} N_1 & \text{for even } N_1 \\ N_1 - 1 & \text{for odd } N_1. \end{cases} \tag{3.32}$$

If we denote by $b(s)$ the size of the reachable set of nodes of separation s from node d_i, then:

$$\sum_{d_j \in V \setminus d_i} h(d_i, d_j) = \sum_{s=1}^{d} s b(s). \tag{3.33}$$

The number of nodes in the reachable set of separation s from node d_i for a torus lattice of size $N_1 \geq 3$ is given by [52]:

$$\begin{array}{lll} N_1 \text{ even}: & b = 4s & ;\ s \leq \frac{N_1}{2} - 1 \\ & b = 4s - 2 & ;\ s = \frac{N_1}{2} \\ & b = 4(n-s) & ;\ \frac{N_1}{2} + 1 \leq s \leq N_1 - 1 \\ & b = 1 & ;\ s = N_1 \\ N_1 \text{ odd}: & b = 4s & ;\ s \leq \frac{N_1 - 1}{2} \\ & b = 4(n-s) & ;\ \frac{N_1 - 1}{2} \leq s \leq N_1 - 1 \end{array} \tag{3.34}$$

Combining these equations and substituting in (3.31) yields the result.

Chapter 4

Optimal Routing in Networks with Constrained Buffers

4.1 Introduction

One of the important limitations of sensor networks is the small memory capacity of sensor devices (e.g. Berkeley motes have 512 KB [66]). This memory constraint translates into a limited (and generally small) space for the temporary storage of packets [75], which causes packet loss due to buffer overflow. In this chapter, we focus on the analysis and design of routing algorithms that maximize the throughput per node for networks with constrained buffers at the nodes.

In the case of finite buffers, deriving the network capacity and maximum rate requires to compute the distribution on the queue size at the nodes. Assuming that all packets have the same size, we model each node as a first come first served (FIFO) single-server queue with a deterministic service time equal the time a packet takes to be transmitted from one node to the following. Then, the distribution on the queue size at the nodes is obtained by solving the associated $G/D/1$ queueing network problem. However, the analysis of queueing networks is complex and no analytical exact solutions are known even for the simplest cases [10]. The common technique to resolve queueing network problems consists in using approximations to reduce the network to a simpler model that allows a tractable analysis. Most of these approximations assume exponential service times (exponentially distributed packet sizes) and Poisson arrival processes in the queues, and apply Jackson's independence assumption [10]: each queue in the network is analyzed as a M/M/1 queue independently of other queues. This independence approximation has been also used for more general arrival processes and deterministic service times [49]. Although the independence approximation works well for low rates, it degrades rapidly as the rate increases.

In this chapter, we propose a new approximation model for $G/D/1$ queue networks that captures the dependence between the distributions on the queue size at different nodes. In the particular case of lattice networks, this approximation model allows to reduce a $G/D/1$ queue lattice network of arbitrary size to a four $G/D/1$ queues network, simplifying considerably the network problem and obtaining an accurate distribution on the queue size at the nodes even at high rates. This approximation also allows us to study the effect of rout-

ing on the queue distribution and derive the routing algorithms that achieve the maximum transmission rate.

Using this approximation model, we analyze the distribution on the queue size induced by different routing algorithms under the same three communication models described in the previous chapter. Under the uniform traffic communication model, we obtain the distribution on the queue size at the nodes induced by the optimal routing algorithm, that is, row-first. Under the central and border data gathering, we show that the performance of different routing algorithms that achieve capacity under the infinite buffer assumption differs considerably when we consider finite buffers at the nodes. For both communication models, we derive the routing algorithms that minimize buffer overflow and hence achieve the maximum rate per node.

The rest of the chapter is organized as follows. In Section 4.2, we introduce the network model and definitions. In Section 4.3, we briefly review the main approximation techniques used to solve the queueing network problem. In Section 4.4, we study the uniform communication model with finite buffers and propose new approximate models to analyze the corresponding queueing network. Using these models, we compute the packet distribution in the queues and study the performance of routing algorithms under finite buffers. In Section 4.5, we study the central data gathering scenario and derive the optimal routing algorithm that minimizes overflow losses. In Section 4.6, we analyze the border data gathering problem.

4.2 Model and definitions

The network model we consider in this chapter is identical to the model described in the previous chapter with the additional constraint of finite buffers. We assume that sources generate constant size packets following a Bernoulli distribution with constant average rate of $\mathcal{R}$ packets per time unit. Nevertheless, as we will explain later, most of the approximation algorithms we propose in this chapter are valid for a more general class of sources.

We assume that nodes are equipped with limited buffer capabilities for the temporary storage of Q packets. When packets arrive at a particular node or are generated by the node itself, they are placed into the buffer until the node has the opportunity to transmit them through the required link. If the buffer is full, the new arrived packet is dropped. Equivalently, we can consider that there are 4 queues per node, each queue associated to one of the 4 output links.

Since the number of packets in a network with finite buffers is always bounded ($Q \times |V|$), a different definition for achievability is needed. Notice also that, due to randomness of buffer overflow, there is a non-zero probability of packet loss at any rate. A rate $\mathcal{R}$ is said to be *feasible* if the average loss probability is smaller than a given (sufficiently small) threshold τ_0. Along this paper, we consider $\tau_0 = 5.10^{-3}$. Note that the results we present in this chapter do not depend qualitatively on the choice of this parameter. Even if the quantitative values we show in the simulation sections definitively depend on the value of τ_0, the relative performance of different routing algorithms remains unchanged.

We extend the notation developed in the previous chapter to evidence the dependency of the rate with the buffer size Q:

Definition 4.1 *A routing algorithm Π is said to have maximum achievable rate $\mathcal{R}^{\Pi}_{sup}(N, Q)$ if any rate $\mathcal{R} = \mathcal{R}^{\Pi}_{sup}(N, Q) - \epsilon, \ \forall\, \epsilon > 0$, is feasible in a network of size $N_1 \times N_1$ with buffer size Q using Π as routing algorithm. Moreover, any rate $\mathcal{R} \geq \mathcal{R}^{\Pi}_{sup}(N, Q)$ is not feasible.*

Note that for $Q = \infty$, $\mathcal{R}^{\Pi}_{\text{sup}}(N, \infty)$ is the supremum of the set of achievable rates. Obviously, $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q) \leq C(N) \; \forall \; Q$.

In the next section we briefly review the approximation techniques used in the literature to solve the queueing network problem. Then, we propose a new approximation model that gives an accurate distribution even at high loads. Finally, using this new approximation, we study the routing algorithms that achieve the maximum $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q)$ for the three communication models described in the previous chapter.

4.3 Queueing network approximations

To solve the associated queueing problem and obtain the distribution on the number of packets in the queue for each node, some approximations are needed. Most of these approximations are based on Kleinrock's independence assumption and Jackson's theorem [10]. Jackson's theorem says that the numbers of customers in a queueing network are distributed as if each queue is $M/M/1$ and is independent of the other queues. This independence approximation is used even for more general queue models, as for instance, if the service time is other than exponentially distributed. Jackson's theorem provides a good first approximation in many practical situations even if its use cannot be rigorously justified [10].

Previous works that consider finite buffers are mainly based on Jackson's theorem [10]. Harchol-Balter and Black [31] considered the problem of determining the distribution on the queue sizes induced by the greedy routing algorithm in torus and square lattice networks by reducing the problem into a product-form Jackson queue network and analyze it using standard queueing theory techniques. Mitzenmacher [49] approximated the queueing network by an equivalent Jackson network with constant service time queues. He provided bounds on the average delay and the average number of packets for square lattices. Unfortunately, the separation of the upper and lower bound, in the general case, grows as the square root of the number of nodes.

To reduce the complexity of the queueing network analysis, all the aforementioned papers make an independence approximation and consider the analysis of just one queue (node). However, even if the independence approximation works well under low rate, it degrades rapidly as the rate increases. In this chapter we propose a new approximation model that captures the dependence between the distribution on the queue size at different nodes and consequently, provides a better approximation under high rate.

This approximation model is based on some results by Neely, Rohrs and Modiano's [53; 54] on equivalent models for tree networks of deterministic service time queues. We briefly review now the main results in [53; 54].

Theorem 4.1 *([53]) The total number of packets in a two stage system of deterministic service time queues is the same as in a system where the first stage queues are replaced by simple delays equal to the service time.*

Theorem 4.2 *([54]) The analysis of the queue distribution in the head node of a multi-stage tree system of deterministic service time queues with stationary and independent exogenous traffic sources can be reduced to the analysis of a simpler two-stage equivalent model, which is formed by considering only nodes located one stage away from the head node and preserving the exogenous inputs.*

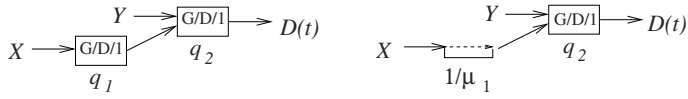


Figure 4.1: The total number of packets in this two stage system (a) with two deterministic service time queues q_1 and q_2 is the same as in this simplified system (b) where the first stage queue q_1 has been replaced by simple delays equal to its service time $1/\mu_1$.

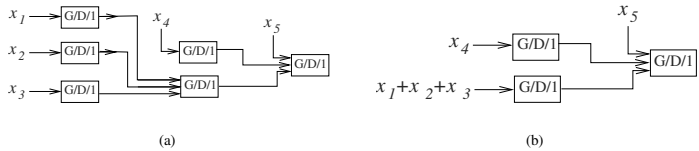


Figure 4.2: The number of packets in the head node of the tree network (a) is the same as in the two-stage equivalent model (b).

Figure 4.1 illustrates the equivalence provided by Theorem 4.1. Figure 4.2(a) shows a tree network and Figure 4.2(b) its two-stage equivalent model. Importantly, as long as the sources are stationary and independent, these equivalences do not require any assumption about the nature of the generated traffic.

4.4 Uniform communication with constrained buffers

We start by analyzing the problem of routing in lattice networks under the uniform communication model. We introduce some approximations that simplify the queueing network analysis and provide meaningful theoretical results that are close to the results obtained by simulation, as experimentally shown later.

We restrict our analysis to the routing algorithm that achieves capacity with unconstrained buffers in both torus and square lattices, namely, row-first routing. Moreover, as we show later in this section, row-first routing is also optimal for finite buffers.

When finite buffers are considered, the maximum rate per node is clearly reduced due to buffer overflow. Overflow losses will first appear in the most loaded node, which will determine the maximum achievable rate $\mathcal{R}^{\Pi}_{\text{sup}}(N,\infty)$. For both torus and square lattices, we denote the most loaded node as d_m. In a square lattice, d_m is clearly the node located in the center of the network(Lemma 3.4). In a torus, if the routing algorithm is space invariant, all

nodes support exactly the same average traffic (Theorem 3.1) and, furthermore, all nodes in the network are indistinguishable due to the torus symmetry. Therefore, we can consider that d_m is any node in the network.

First, we decompose d_m into four identically distributed and independent FIFO queues each one associated to one of its four output links. The arriving packets to d_m whose final destination is not d_m, are sent through one of the four output links depending on their destinations. In view of the symmetry of d_m for both torus and square lattices with respect to the central node, the arrival distributions to these four links are clearly identical. Moreover, due to the independence of packet generation, we assume that these arrival distributions are also independent. Therefore, we approximate the distribution on the queue size at d_m as the addition of these four iid distributions and compute it as the convolution of each individual queue. This way, we reduce the problem to computing the distribution on the size of only one of these iid queues, q_m, associated with the output link l_m in d_m.

Next, we propose different approximations for full-duplex and half-duplex links (whether l_m is half-duplex or full-duplex) and compare them with experimental results.

4.4.1 Approximation models for full-duplex communication channels

If l_m is a full-duplex channel, q_m has a dedicated link and, since all packets have the same size, it can be modeled as a deterministic service time queue. We can therefore use the results of Section 4.3 to obtain the distribution on the size of q_m. First, we identify q_m as the head node of a tree network composed of all the nodes sending traffic through l_m (Figure 4.3(a)). Applying Theorem 4.2, the distribution on the size of q_m can be *approximated* by the distribution at the head node of the two-stage model (Figure 4.3(b)), where we only consider the three neighbors located one hop away from d_m preserving the traffic generated by the entire network that flows through l_m.

Note that even if the two-stage model of Theorem 4.2 is an *equivalent* model for tree networks, the reduced two-stage model of Figure 4.3(b) is an *approximation* model. The reason is that the network associated to q_m (Figure 4.3(a)) is not exactly a tree and, therefore, it does not correspond exactly to the network of Theorem 4.2 (Figure 4.2(a)). In addition to the exogenous inputs generated at each node, there is also traffic leaving the network at each node that corresponds to the traffic that has reached its destination or the traffic that does not travel through q_m. Under the uniform communication model, the average traffic that reaches its destination and leave the network is equal to the average rate per node $\mathcal{R}$ for all nodes. However, as the network size increases, $\mathcal{R}$ needs to decrease as $\mathcal{O}(1/\sqrt{N})$ (network capacity is $\mathcal{O}(1/\sqrt{N})$), and consequently, the departing traffic at each node becomes negligible compared to the traffic that flows through the same node. Hence, the network of queues associated to q_m is *approximately* a tree network, and the two-stage model provides an approximated network which becomes more precise as the network size increases.

According to Theorem 4.2, the arrivals to the nodes of the first stage in the two-stage model correspond to the addition of all exogenous inputs that are routed through l_m. Since packets are generated in sources following independent Bernoulli distributions, this arrival process converges, as the number of nodes increases, to a Poisson distribution.

Note that under the uniform communication model, packets travel $\mathcal{O}(\sqrt{N})$ hops on average before reaching their destination. Using row-first routing, packets travel most of the time along the same row or column, turning only once. Consequently, the traffic entering a node by a row or a column link continues, with high probability, along the same row or column.

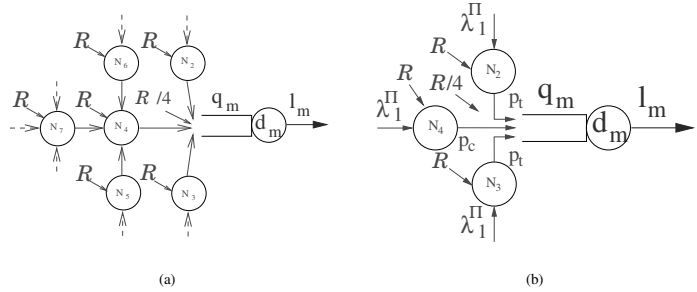


Figure 4.3: Tree network and equivalent two-stage model. (a) Tree network associated to one output link l_m of d_m. (b) Two-stage equivalent.

Let p_c denote the probability of a packet to continue along the same row or column, and p_t the probability of turning. These probabilities are easy to calculate at d_m:

$$p_c = \begin{cases} \frac{\sqrt{N}-1}{\sqrt{N}+1}, & \text{for the square lattice,} \\ \frac{\sqrt{N}/2-1}{\sqrt{N}/2+1}, & \text{for the torus,} \end{cases} \tag{4.1}$$

$$p_t = \begin{cases} \frac{\sqrt{N}-1}{2\sqrt{N}(\sqrt{N}+1)}, & \text{for the square lattice,} \\ \frac{\sqrt{N}/2-1}{\sqrt{N}(\sqrt{N}/2+1)}, & \text{for the torus.} \end{cases} \tag{4.2}$$

Note that p_c+p_t is equal to one minus the probability that the packet final destination is d_m. Note also that p_c goes asymptotically to one as the number of nodes increases, while p_t goes to zero. It follows that q_m receives most of the traffic from the node located in the same row or column as l_m.

Apart from the traffic that arrives from its neighbors, d_m generates also new traffic that is injected to the network at a rate $\mathcal{R}$. Considering again the symmetry of d_m, the fraction of this traffic that goes through l_m is $\mathcal{R}/4$. The average arrival rate $\lambda^{\Pi}_{q_m}$ to q_m can be computed as the addition of the traffic generated in d_m and the traffic arriving from its neighbors:

$$\lambda^{\Pi}_{q_m} = \mathcal{R}/4 + \lambda^{\Pi}_1 (p_c + 2p_t), \tag{4.3}$$

where λ^{Π}_1 is the total arrival rate to the neighbors of d_m (Figure 4.3(b)). For row-first routing and the uniform communication model, λ^{Π}_1 is equal to:

$$\lambda^{r-f}_1 = \begin{cases} \frac{\mathcal{R}\sqrt{N}}{4}, & \text{for the square lattice,} \\ \frac{\mathcal{R}\sqrt{N}}{8}, & \text{for the torus.} \end{cases} \tag{4.4}$$

We can express $\mathcal{R}$ as a fraction of the network capacity $C(N)$, that is, $\mathcal{R} = \alpha C(N)$, and denote α as the relative capacity. Then, recalling the capacity formula (3.4), $\lambda^{r-f}_1 = \alpha$

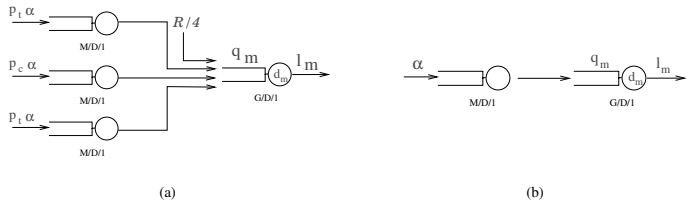


Figure 4.4: Approximation models for full duplex communication channels. (a) Two-stage model used to analyze the distribution on the size of q_m. (b) Two-queue model: reduced two-stage model without crossing packets and neglecting exogenous traffic in d_m.

for both torus and square lattices. Putting everything together, the resulting approximation model is shown in Figure 4.4(a).

Regardless of the number of nodes in the network, we reduce the analysis of the distribution on the size of q_m to a four queue network. This approximation holds for any exogenous traffic distribution as long as it is stationary and independent among the different sources (Theorem 4.2).

Using now these reduced approximation models, we can derive some interesting results about the buffer requirements to achieve a certain relative capacity.

Theorem 4.3 *The buffer size Q required to achieve a certain relative capacity α decreases with the network size N.*

Proof: By Theorem 4.1, the total number of packets in the approximated model (Figure 4.4(a)) is the same as a system where the queues of the first stage have been replaced by pure delays of T time slots (Figure 4.5(a)), and this is equivalent to injecting all the arrivals into a single pure delay (Figure 4.5(b)). The total average arrival rate λ_{s1}^{r-f} to the first stage queues is:

$$\lambda_{s1}^{r-f} = \alpha(p_c + 2p_t) = \alpha(\sqrt{N} - 1)/\sqrt{N}. \tag{4.5}$$

Therefore, note that for a fixed α and large N, the total number of packets in this two queue model is almost constant with N.

We can decompose the total number of packets $S(t)$ in the approximated model (Figure 4.4(a)) as the number of packets in the first stage $S_1(t)$ plus the number of packets in the head node $S_h(t)$, that is, $S(t) = S_1(t) + S_h(t)$. As N increases, p_c goes asymptotically to one and most of the traffic is served by the same first stage queue. Consequently, for a fixed α, $S_1(t)$ increases with N. Equivalently, $S_h(t)$ decreases. In the limit, we can approximate the model by just two constant service time queues as shown in Figure 4.4(b), where no buffer is needed in the head node. □

We can simplify our model even further while still keeping the important properties that determine the queue size distribution. Note that, since p_t is $\mathcal{O}(1/\sqrt{N})$, we can simplify the

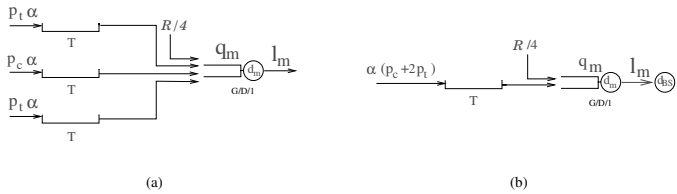


Figure 4.5: Two-stage equivalent networks. (a) If we replace the queues of the first stage by pure delays of T time slots, the total number of packets in the approximated model remains constant. (b) In terms of number of packets, this is equivalent to injecting all the arrivals to a single pure delay.

model for large networks by assuming that the number of packets turning at d_m is negligible, i.e., packets arrive at q_m only from the neighbor located in the same row or column as l_m. Similarly, the exogenous traffic generated at d_m, goes also asymptotically to zero ($\mathcal{O}(1/\sqrt{N})$) as compared with the incoming traffic α, and can also be neglected. Consequently, we approximate the queueing network by a two-queue model where q_m is a deterministic service time queue that receives traffic from another deterministic service time queue with the same service time and average arrival rate equal to α (Figure 4.4(b)). It follows that the number of packets in q_m is (at most) one with probability α and zero with probability $1 - \alpha$.

Finally, the distribution $P_m(k)$ on the total queue size k at d_m in the two-queue model, is given by the addition of four independent and identically distributed queues associate to the four outgoing links from d_m:

$$P_m(k) = \begin{cases} \binom{4}{k}(1-\alpha)^{(4-k)}\alpha^k, & \text{for } 0 \leq k \leq 4, \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

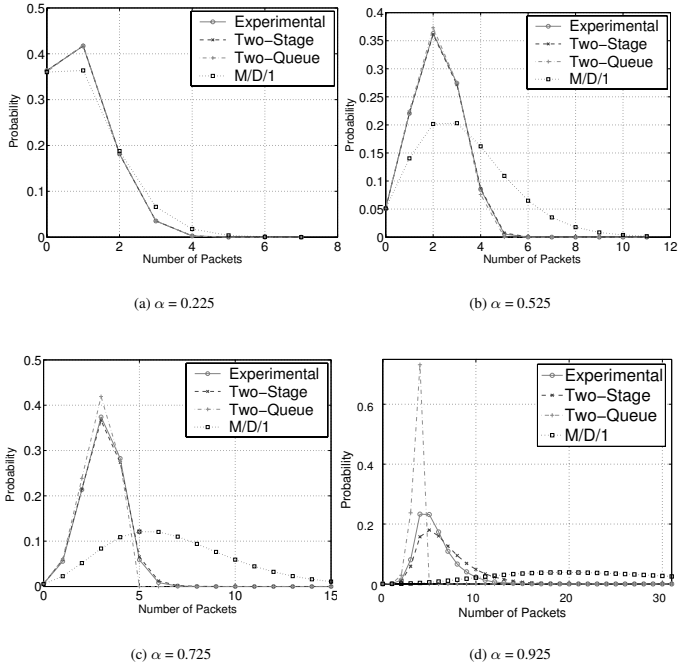
Note that both approximation models proposed, namely the two-stage model (Figure 4.4(a) and the two-queue model Figure 4.4(b)), are asymptotically exact.

Simulation results

Figure 4.6 shows the distribution on the size of q_m obtained by simulating the whole queueing network, the two-stage model (Figure 4.4(a)), the two-queue model (Figure 4.4(b)) and the usual M/D/1 approximation for different values of the relative capacity α in a 121×121 square lattice network.

For the M/D/1 approximation, we simply apply Jackson's Theorem and consider that each queue in the network is M/D/1 independent of other queues [49]. Therefore, we approximate q_m by a M/D/1 queue with a Poisson arrival with rate α.

Both the two-stage and two-queue models allow very good analysis in low and medium load. Experimentally, we have found that a good approximation is obtained for $\alpha < 0.8$. Beyond this traffic intensity, some of the assumptions we make are not totally valid and the



(a) $\alpha = 0.225$ (b) $\alpha = 0.525$

(c) $\alpha = 0.725$ (d) $\alpha = 0.925$

Figure 4.6: Distribution on the queue size at d_m using both approximation models we propose and the independence approximation for different values of α in a 121×121 square lattice network with full-duplex links: a) $\alpha = 0.225$, b) $\alpha = 0.525$, c) $\alpha = 0.725$, and d) $\alpha = 0.925$. Both approximation models we propose clearly outperform the independence approximation model.

approximation quality degrades. For instance, to approximate the network by a tree, we neglected the traffic leaving the network at each (destination) node, which increases as $\mathcal{R}$ increases.

Both approximation models that we propose clearly outperform, at all rates, the M/D/1 model based on Jackson's theorem. The reason is that the M/D/1 approximation neglects the existing correlation between the traffic and queue occupancy of neighbor nodes, while a simple two-stage model captures this correlation. Therefore, the M/D/1 model only approximates the distribution closely under low load conditions, that is, while the independence

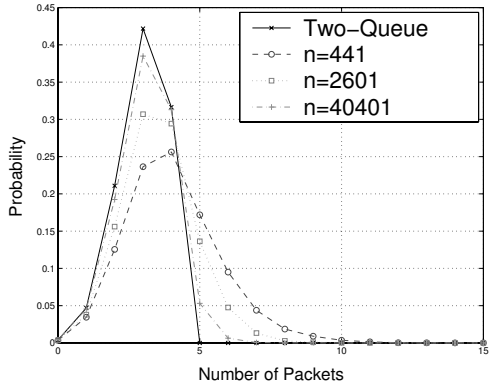


Figure 4.7: Distribution on the queue size at d_m for $\alpha = 0.75$ and different network sizes with full-duplex links. The full line shows the distribution given by the two-queue approximation which, for a fixed relative capacity α, is independent of the network size N. Dashed lines show the distribution obtained experimentally for different network sizes N.

approximation is valid. Under medium and high loads, the independence assumption does not hold, and the approximation quality degrades rapidly. For $\alpha = 0.525$, we already observe that this approximation is far from the distribution obtained experimentally.

Figure 4.7 shows the distribution on the queue size at d_m obtained experimentally for a constant relative capacity $\alpha = 0.75$, as a function of the network size N. We compare the experimental distributions with the distribution given by the two-queue approximation model, which is independent of the network size. As expected, as the size of the network N increases, the packet distribution converges asymptotically to the two-queue model (Figure 4.4(b)). Consequently, as stated in Theorem 4.3, the probability of having more than four packets in d_m (one for each output link l_m) goes asymptotically to zero as we increase N.

Note that in all the approximation and simulation results shown in this chapter, we compute the distribution of queue sizes by assuming infinite buffer queues (G/D/1) at the nodes. This distribution gives complete information about the buffer requirement to achieve a certain capacity or the capacity achieved with a given buffer size. For instance, if we need to transmit information at a rate per node equal to 0.725 of the network capacity in a 121×121 network, we compute the distribution of the queue size in the most loaded node (Figure 4.6(c)) and obtain that the required buffer is equal to 7 packets.

4.4.2 Approximation models for half-duplex communication channels

For half-duplex, we cannot apply the same techniques as in the full-duplex case since the arrival and service times in d_m are no longer independent. If d_m receives k packets from its

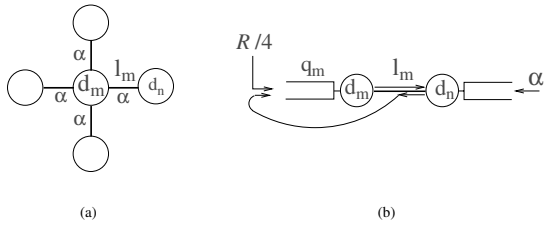


Figure 4.8: Half-duplex approximation model. (a) Network with half-duplex links. (b) Approximated model for the queue associated to l_m.

neighbors, not only is its queue increased by k packets, but also it can transmit, at most, $4-k$ packets using the remaining links.

We assume as in the full-duplex case that d_m is composed of four independent and identically distributed queues associated to the four output links and we analyze the distribution on the size of one of these queues q_m associated to the output link l_m.

To capture the dependence between arrivals and departures, we propose the following approximation. Every time d_m wants to send a packet through l_m, it has to compete for l_m with one of its neighbors, d_n (Figure 4.8(a)). If d_m takes l_m first, it can transmit a packet and the size of q_m is reduced by one. However, if d_n takes the link first and sends a packet, not only is d_m unable to transmit, but also the size of q_m is increased by one if the final destination of the packet is not d_m. Note that, in practice, packets sent by d_n never go through l_m (packets do not go backwards) although they stay in d_m. However, by putting these packets into l_m we simulate packets arriving from the other neighbors of d_m and prevent packet transmissions. This approximation is represented in Figure 4.8(b).

We denote by ρ_m the utilization factor of q_m. That is, $\rho_m = \frac{\lambda^{\Pi}_{q_m}}{\mu_{q_m}} = \frac{\alpha}{\mu_{q_m}}$, where $\lambda^{\Pi}_{q_m}$ is the arrival rate to q_m, and μ_{q_m} is the service rate. Note that $\lambda^{\Pi}_{q_m}$ is identical in both half-duplex and full-duplex models. Similarly, we denote by ρ_n the utilization factor of the queue q_n in d_n associated to l_m. We assume that the probability that d_m captures the link before d_n is equal to $1/2$. Therefore, if q_m has a packet waiting to be transmitted, the probability p_s of sending it this time slot is simply equal to the probability of d_m being the first to capture the link plus the probability of d_n having nothing to transmit through l_m:

$$p_s = \frac{1}{2} + \frac{1}{2}(1-\rho_n) = 1 - \rho_n/2; \tag{4.7}$$

We model q_m service time as a geometric distribution with parameter p_s. That is, if d_m does not capture l_m in this time slot, we assume that it tries to capture it in the next time slot with the same probability.

As in the full-duplex case, we approximate arrivals to d_n as a Poisson distribution with parameter α. Accordingly, interarrival times are independent and exponentially distributed with the same parameter. In addition to arrivals from d_n, new packets are also produced at

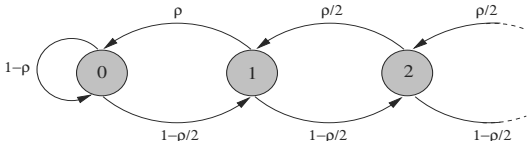


Figure 4.9: Markov chain model for the half-duplex links model.

d_m following a Bernoulli distribution with rate $\mathcal{R}$. Considering again the symmetry of d_m, the fraction of this traffic that goes through l_m is $\mathcal{R}/4$.

Note that both distributions, arrivals and service time, are memoryless. This memoryless condition allows to use a Markov chain analysis, that is, if we denote by $X_m(t)$ the number of packets in the queue q_m at time t, $\{X_m(t) \mid t > 0\}$ can be approximated using a Markov chain. As the network size increases, the difference between both utilization factors ρ_m and ρ_n becomes negligible, and we can assume that $\rho_m = \rho_n = \rho$. Moreover, the new traffic generated at d_m becomes negligible ($\mathcal{O}(1/\sqrt{N})$) compared to the traffic that arrives from d_n.

Applying these simplifications, the transition probability matrix $P_m(j,k)$ associated to $\{X_m(t) \mid t > 0\}$ can be approximated by:

$$P_m(0,k) = \begin{cases} 1-\rho, & k = 0, \\ \rho, & k = 1, \end{cases}$$

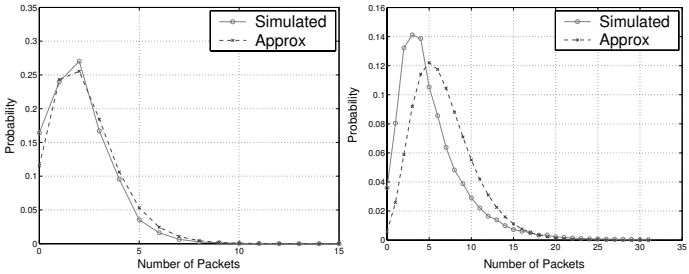
$$P_m(j,k) = \begin{cases} 1-\rho/2, & k = j-1, \\ \rho/2, & k = j+1, \end{cases}$$

whose transition graph is shown in Figure 4.9.

Simulation results

Figure 4.10 shows the distribution on the queue size at d_m for different values of α in a 121×121 square lattice network with half-duplex links. This figure compares the packet distribution obtained by simulation with the Markov chain approximation. This model closely approximates the experimental distribution for low to moderate rate per node ($\alpha < 0.8$). As in the full-duplex case, beyond this traffic intensity, the independence assumptions are no more valid and the approximation quality degrades.

Figure 4.11 shows the distribution on the queue size at d_m obtained experimentally for a constant relative capacity $\alpha = 0.75$, as a function of the network size N. We compare the experimental distributions with the distribution given by the Markov model approximation, which is independent of the network size. Note that the Markov model closely approximates the experimental distributions. A key difference with the case of full-duplex links is that, as the network size increases, the buffer requirements do not go asymptotically to zero. The intuitive reason is that, in the case of half-duplex links, l_m is shared between d_m and d_n and, even if the input rate $\lambda_{l_m}^{\Pi}$ is less than the link capacity, there is a non-zero probability that d_m competes for the link with d_n, in which case one of them has to store the packet for a further transmission. In other words, the stationary probability distribution $\nu_m(k)$ of the Markov chain that approximates the number of packets k in q_m, has positive values for $k > 1$ for any value of N.



(a) $\alpha = 0.725$ (b) $\alpha = 0.925$

Figure 4.10: Distribution on the queue size at d_m for different values of α in a 121×121 square lattice network with half-duplex links.

4.5 Central data gathering with constrained buffers

We established in the previous chapter (Lemma 3.3) that the only necessary and sufficient condition for a routing algorithm to achieve capacity under the central data gathering model and unconstrained buffers, is to uniformly distribute the traffic among the four arrival links to d_{BS}. Although there is a wide class of routing algorithms that satisfy this condition, we show in this section that their performance is quite different when the buffers are constrained to be finite.

4.5.1 Approximation models

For the sake of simplicity, we restrict our analysis to a particular location of d_{BS}: the square lattice center. Note that the analysis of this location is equivalent to solving the problem for any location in the torus network. Nevertheless, a similar analysis can be carried out for any location. Note that, for the central node, row-first routing does not satisfy the optimality condition: by always forwarding packets along the same row until they reach the column of d_{BS}, most of the traffic reach d_{BS} through the upper and lower links while the rest of the links are underused. However, there are many routing algorithms that achieve capacity for infinite buffers. For instance, a simple routing algorithm that satisfies the capacity condition is the *random greedy algorithm* [40], where nodes use row-first or column-first as routing algorithm with equal probability.

To analyze the network capacity for a given routing algorithm under finite buffers, we proceed as in the uniform communication model. First, we identify the most loaded node d_m and associate the network to a tree. Then, we reduce this tree to its two-stage equivalent model and obtain the packet distribution in d_m by analyzing the packet distribution in the head node of the two-stage model. We perform this analysis for any shortest path routing algorithm Π that achieves capacity in the infinite buffer case.

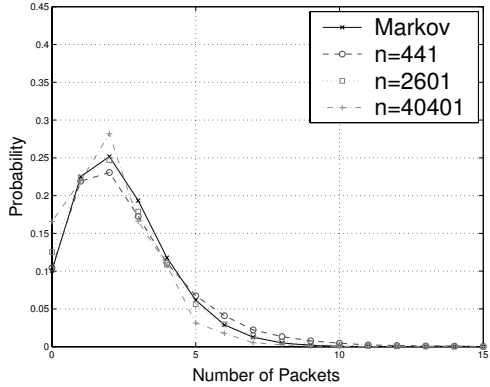


Figure 4.11: Distribution on the queue size at d_m for $\alpha = 0.75$ and different network sizes with half-duplex links. The full line shows the distribution given by the Markov model approximation which, for a fixed relative capacity α, is independent of the network size N. Dashed lines show the distribution obtained experimentally for different network sizes N.

The bottleneck of the network is clearly located in the four neighbors of d_{BS}. Moreover, if Π achieves capacity when buffers are infinite, the total arrival traffic to d_{BS} is uniformly distributed among the four links in $\varphi^l(d_{BS})$ (Lemma 3.3). Due to the independence of packet generation, we can assume that the distributions on the queue size in these four nodes are independent and identically distributed. Consequently, we reduce the problem to computing the queue distribution for one of these neighbors, say d_m. We denote by l_m the link between d_m and d_{BS} and by q_m the queue in d_m associated to l_m (Figure 4.12(a)).

We consider now only those nodes in the network that generate traffic through l_m. These nodes form a tree with q_m as head, with exogenous inputs at each node, and with no traffic leaving the network. Applying Theorem 4.2, the packet distribution in q_m is the same as in its two-stage model (Figure 4.12(b)). Note that in this case there is not traffic leaving the network at each node as in the uniform communication case. Therefore, the two-stage model is not an approximation but an exact model for all rates.

The arrivals to the three nodes of the first stage are the addition of all the traffic generated by the network that goes through l_m. If Π achieves capacity for infinite buffers, by Lemma 3.3, the total average traffic that flows through l_m is equal to:

$$\lambda_{l_m}^{\Pi} = \frac{\mathcal{R}(N-1)}{4}. \tag{4.8}$$

We denote by λ_1^{Π}, λ_2^{Π} and λ_3^{Π} the average arrival rates to the three first stage nodes of the two-stage model (Figure 4.12(b)). These three nodes have to route all the traffic that goes

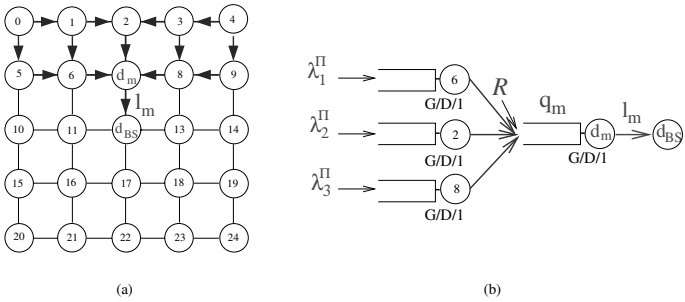


Figure 4.12: Two-stage equivalent model. (a) The possible nodes in the network that generate traffic through l_m constitute a tree network where l_m is the head node and (b) its two-stage model.

through l_m except the traffic generated by d_m itself. That is,

$$\lambda_1^{\Pi} + \lambda_2^{\Pi} + \lambda_3^{\Pi} = \frac{\mathcal{R}(N-1)}{4} - \mathcal{R}. \tag{4.9}$$

We obtain the distribution on the size of q_m by analyzing the distribution at the head node of the two-stage model, which obviously depends on the values of λ_1^{Π}, λ_2^{Π} and λ_3^{Π}.

4.5.2 Optimal routing algorithms

Particularly, we are interested in finding the routing algorithm Π that, for a given Q, achieves the maximum $\mathcal{R}_{\text{sup}}^{\Pi}(N, Q)$. This is equivalent to minimizing the number of packets in q_m for any given $\mathcal{R}$.

Different routing algorithms generate different values for λ_1^{Π}, λ_2^{Π} and λ_3^{Π}, and consequently, different distributions on the size of q_m. First, we analyze the values of λ_1^{Π}, λ_2^{Π}, and λ_3^{Π} that generate the minimum number of packets in q_m and then, we analyze the routing algorithm that induces such values.

Theorem 4.4 *In a two-stage network where the total average arrival rate is fixed, i.e.,* $\lambda_1^{\Pi} + \lambda_2^{\Pi} + \lambda_3^{\Pi} = \lambda_t$*, the values of* $\lambda_i^{\Pi},\ i \in 1, 2, 3$ *that minimize the number of packets in the head node for any arrival distribution are such that all traffic arrives only through one node of the first stage. That is:*

$$\lambda_i^{\Pi} = \begin{cases} \lambda_t, & \textit{for } i\textit{=1,2 or 3}, \\ 0, & \textit{otherwise}. \end{cases} \tag{4.10}$$

Proof: By Theorem 4.1, the total number of packets in the two-stage model (Figure 4.12(b)) is the same as a system where the first stage queues has been replaced by pure

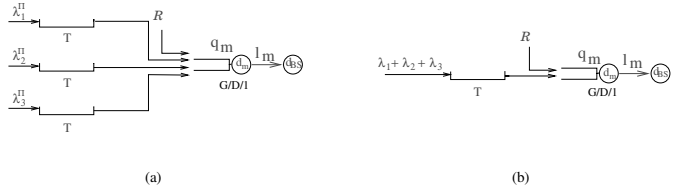


Figure 4.13: Two-stage equivalent model. (a) Two-stage model where the first stage queues has been replaced by pure delays of T time slots, (b) equivalent to injecting all the arrivals to a single pure delay.

delays of T time units (Figure 4.13(a)). In terms of number of packets in the system, this is equivalent to injecting all the arrivals to a single pure delay (Figure 4.13(b)). Consequently, the total number of packets in the system is equivalent for any combination of λ_i^{Π} values.

We can decompose the number of packets in the two-stage model as the packets in the first stage plus packets in the head node. Minimizing the number of packets in the head node is therefore equivalent to maximizing the packets in the first stage. Since the first stage is composed of three $G/D/1$ queues with equal service time, the number of packets in the first stage is maximized when all the traffic goes through only one queue. □

Using the two-stage equivalent model and Theorem 4.4, we can easily design routing algorithms that minimize packet overflow. Consider a routing algorithm Π that achieves capacity for the infinite buffer case. That is, Π generates the optimal traffic distribution to the neighbor nodes of d_{BS} according to Lemma 3.3.

As we have previously pointed out, maximizing $\mathcal{R}_{\text{sup}}^{\Pi}(N, Q)$ is equivalent to minimizing the number of packets in the most loaded nodes. Therefore, if we want Π to be also optimal for finite buffers, it needs to generate the optimal arrival distribution (as given in Theorem 4.4) in all the nodes.

Intuitively, if Π achieves the maximum $\mathcal{R}_{\text{sup}}^{\Pi}(N, Q)$, the input traffic to the most loaded node d_m has to arrive from only one of its neighbors, say d_n. This way, we minimize the overflow losses in d_m. However, by routing all the traffic that arrives at d_m through d_n, the congestion problem is translated to d_n. Consequently, d_n is now the most loaded node, and we can apply the same argument as before. Furthermore, as the network size increases, the difference between the traffic that flows through d_m and d_n goes asymptotically to zero. It is clear now that to obtain the routing algorithm that minimizes the number of packets in all nodes, and equivalently, minimizes the overflow losses, Theorem 4.4 has to be applied to all nodes:

Theorem 4.5 *The routing algorithm that minimizes overflow losses consists in making nodes receive all traffic exclusively from one neighbor.*

We apply now Theorem 4.5 to design algorithms that minimize overflow losses in the case where we constraint packets to take only shortest paths towards the destination and more generally, when non shortest paths are also allowed.

The shortest path routing algorithm that implements this principle is shown in Figure 4.14(a) and consists in the following. In the $N_1 \times N_1$ square lattice, there are $2(N_1 - 1)$ nodes that have only one possible shortest path toward d_{BS}. We denote this set of nodes by $SD(d_{BS})$. For any other node, the optimal routing algorithm consists in forwarding packets to the closest node in $SD(d_{BS})$. Note that there is only one closest node in $SD(d_{BS})$ for all the nodes except for those nodes located in the two diagonals of the lattice. Diagonal nodes forward packets towards only one of the two closest nodes in $SD(d_{BS})$ in such a way that each of the four diagonal nodes at the same distance from d_{BS} chooses a different forwarding node. We denote this routing algorithm as *cross routing*.

Cross routing generates the optimal node arrival distribution among all shortest path routing algorithms. Although the most loaded nodes, that is, nodes located close to d_{BS}, receive packets from more than one neighbor, most of the traffic arrives mainly from one. The average arrival rates generated in d_m by cross routing are: $\lambda_1^{c-r} = \mathcal{R}(N-9)/4$, $\lambda_2^{c-r} = \mathcal{R}$, and $\lambda_3^{c-r} = 0$. It follows that cross routing is asymptotically optimal.

However, according to Theorem 4.5, the optimal routing consists in making nodes receive all traffic exclusively from one neighbor. This condition can only be fully satisfied by a *non-shortest path* routing. Applying this condition recursively, the set of optimal routing algorithms is such that it divides the network into four disjoint subsets of $(N-1)/4$ nodes and joins them with a single path that does not pass twice through the same node and ends in d_{BS}. We denote the optimal routing algorithms set as traveling salesman (TS) routing.

Figure 4.14(b) shows an example of a TS routing algorithm where the traffic flows toward d_{BS} following a spiral. We denote this routing algorithm as *snail routing*. Clearly, Snail routing belongs to TS routing and, according to Theorem 4.5, generates the optimal arrival distribution in all nodes.

Applying theorem 4.2, for all TS routing algorithms, the packet distribution in any node d_i of the network is equivalent to the distribution in the head node of a two queues tandem network, where the exogenous inputs are in first queue the addition of all the traffic that d_i relays, and in the second queue the local traffic generated at d_i. As the network sizes increases, the local traffic generated at d_i becomes negligible with respect to the relayed traffic. In the limit, we can approximate the model by just two constant service time queues where no buffer is needed in the head node. Consequently, the buffer size Q required to achieve a certain relative capacity α goes asymptotically to zero with the network size N.

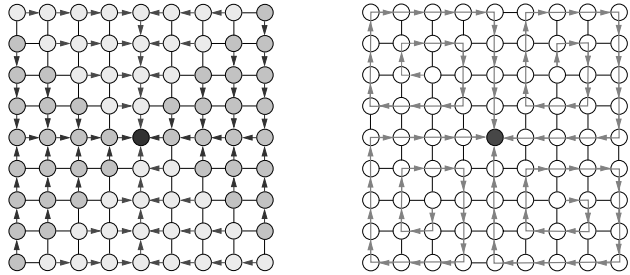
Note that Theorem 4.4 is valid for any arrival distribution, and consequently, it holds for any exogenous traffic generation process at the nodes. Thus, even if the maximum achievable rate $\mathcal{R}_{\text{sup}}^{\Pi}(N,Q)$ clearly depends on the traffic generation process, cross routing and TS routing still achieve the highest maximum rate among the shortest path and non-shortest path routing algorithms, respectively.

We can easily apply the analysis shown in this section to the analogous broadcast communication model, where the central node is the only source in the network and generates different information packets for each node of the network.

4.5.3 Simulation results

We compare the performance of random greedy routing, cross routing and TS routing. Figure 4.15 shows the maximum relative capacity $\mathcal{R}_{\text{sup}}^{\Pi}(N,Q)/C_{cg}(N)$ achieved by different routing algorithms in a 21×21 square lattice network as a function of the buffer size Q.

Notice that although all routing algorithms asymptotically achieve capacity as the buffer



(a) (b)

Figure 4.14: Routing algorithms for finite buffers. (a) Cross routing and (b) Snail routing.

size increases, the maximum achievable rate $\mathcal{R}_{\text{sup}}^{\Pi}(N, Q)$ under small buffers differs strongly among different routing algorithms. As expected, the maximum $\mathcal{R}_{\text{sup}}^{\Pi}(N, Q)$ corresponds to snail routing (TS routing), while cross routing performs best among shortest path routing algorithms.

Figure 4.16 shows the maximum rate achieved by different routing algorithms relative to the maximum rate achieved by the snail routing for a fixed buffer size $Q = 5$ as a function of the network size N. Since all routing algorithms analyzed are asymptotically optimal with the network size, the performance gap between snail routing and these algorithms decreases as the network size increases for a fixed value of Q. The reason is that, as the network size increases, the most loaded nodes receive most of the traffic mainly from only one neighbor. Moreover, note that for a fixed buffer size $Q = 5$, the relative rate achieved goes asymptotically to 1 with the network size N. As we explained previously, the required buffer to reach capacity decreases with the network size, and therefore, even a small buffer of $Q = 5$ is enough to reach capacity for values of N large enough.

Although TS routing achieves the maximum $\mathcal{R}_{\text{sup}}^{\Pi}(N, Q)$, the delay incurred by the packets may be unacceptable. Notice that a packet generated by the furthest node must travel across $(N-1)/4$ nodes before reaching d_{BS}, while for a shortest path routing, the furthest node is $\sqrt{N}-1$ hops away. Moreover, the average path length $\overline{L}_{TS}$ for any TS routing algorithm is $\mathcal{O}(N)$, while for any shortest path routing, $\overline{L}_{s-p}$ is $\mathcal{O}(\sqrt{N})$. TS routing represents an extreme case of the existing trade-off between $\mathcal{R}_{\text{sup}}^{\Pi}(N, Q)$ and delay, achieving the optimal rate per node drastically increases the delay. Equivalently, since most of the energy is commonly consumed in the transmission process, to increase the average path length is equivalent to increase the average power consumption in the network, which might not be a good idea in a wireless scenario for example.

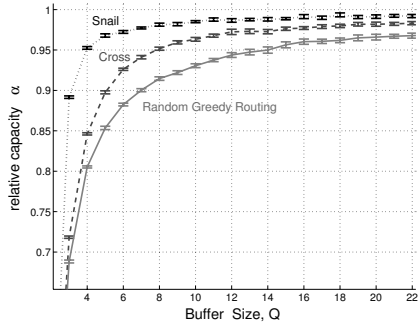


Figure 4.15: Routing for central data gathering: maximum relative capacity α achieved by different routing algorithms in a 21×21 square lattice for different buffer sizes Q.

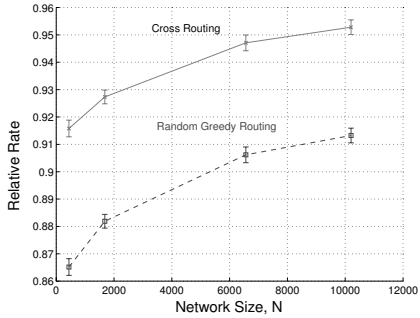


Figure 4.16: Routing for central data gathering: maximum rate achieved by cross routing and greedy routing relative to snail routing for a fixed buffer size $Q = 5$ for different network sizes N.

4.6 Border data gathering with constrained buffers

Finally, we turn our attention to the border data gathering problem. We use the approximation model derived in the previous sections to obtain the routing algorithm that achieves the maximum $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q)$ for finite buffers.

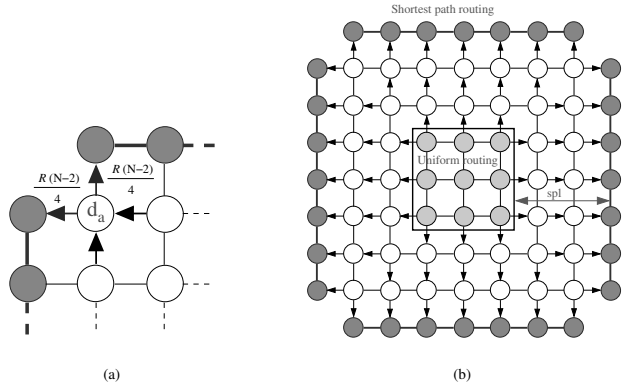


Figure 4.17: Maximum rate trade-off in border data gathering: (a) the nodes located close to the edges carry more traffic. (b) Adaptive routing.

4.6.1 Optimal routing algorithms

We proceed as in the previous section: we apply the routing condition to minimize overflow losses (Theorem 4.5), that is, we make nodes receive all traffic exclusively from one neighbor. Note that the optimal shortest path routing algorithm we derived for infinite buffers (Section 3.5) is also optimal for finite buffers since all nodes receive traffic from only one neighbor. However, we showed that this routing policy does not achieve capacity with infinite buffers. For infinite buffers, we showed that the algorithm that reached capacity was a non-shortest path algorithm that we denoted by uniform border gathering. Under the uniform border gathering algorithm, most nodes receive traffic from many of its neighbors, and therefore, it does not satisfy the optimality condition for finite buffers. Actually, as we show in the next Lemma, in border data gathering no routing algorithm reaches capacity and behaves optimally for finite buffers.

Lemma 4.1 *Under the border data gathering communication model, the routing condition to minimize overflow losses (4.10) and the capacity condition (3.25) cannot be both satisfied simultaneously.*

Proof: Let Π be a routing algorithm that achieves capacity under the infinite buffer assumption. Consider a node d_a located in the diagonal close to the edges, as illustrated in Figure 4.17(a). Notice that it is enough to focus on the traffic that goes through a certain node. By Lemma 3.5, d_a has to carry the traffic of at least two links in $\varphi^l(d_{BS})$, that is, $\lambda^{\Pi}_{d_a} \geq 2\frac{\mathcal{R}(\sqrt{N}-2)}{4}$. That is, d_a receives traffic from more than one neighbor. Otherwise, there is a link l such that $\lambda^{\Pi}_{l} > \frac{\mathcal{R}(\sqrt{N}-2)}{4}$, and Π does not achieve capacity. On the other hand, if

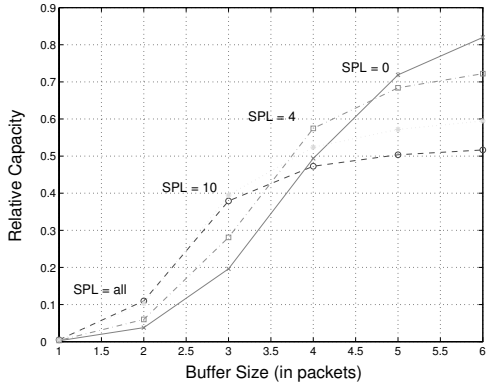


Figure 4.18: Adaptive routing: maximum relative capacity achieved by adaptive routing with different shortest-path limit values as a function of the buffer size in a 41×41 square lattice.

d_a receives traffic from only one neighbor, $\lambda_{d_a}^{\Pi}$ is upper bounded as $\lambda_{d_a}^{\Pi} \leq \frac{\mathcal{R}(\sqrt{N}-2)}{4}$, and (3.25) is not satisfied. □

Lemma 4.1 implies that the design of the optimal routing algorithm for a given queue size Q, should trade-off both the conditions related to approaching capacity and the conditions related to operate optimally in the case of finite buffers. Therefore, we propose an adaptive routing algorithm that depends on the buffer size Q. As in previous sections, the most critical nodes are those located close to the base stations, that is, the most loaded nodes. Therefore, it is in those nodes where it is more important to apply the optimal queue condition.

Consequently, we define the following routing algorithm: nodes located at a distance less than a fixed value *spl* (shortest path limit) from any base station, route packets according to shortest path routing. That is, in nodes close to any base station, we reduce buffer overflow by making nodes receive from one single neighbor. On the other hand, nodes further than spl route packets according to the uniform border gathering. That is, we distribute load as uniformly as possible among the base stations so that the maximum capacity can be achieved. We denote this routing algorithms as *adaptive routing* and it is depicted in Figure 4.17(b). Note that when spl is equal to zero, adaptive routing is equivalent to uniform border gathering, that is, the more loaded node (nodes close to the border) receives traffic from more than one neighbor. As we increase the value of spl, these nodes start receiving packets from only one neighbor (shortest path routing). Finally, when spl is equal to $(\sqrt{N}-1)/2$, all nodes receive packets from only one neighbor and adaptive routing is equivalent to shortest path routing.

4.6.2 Simulation results

Figure 4.18 shows the values of $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q)$ achieved by adaptive routing algorithm with different values of spl in a 41×41 square lattice network, as a function of the buffer size Q.

First, note the trade-off between the rate $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q)$ achieved for big and small buffer sizes: no routing strategy can achieve high rates for both extremes. For high buffer values, the optimal routing strategy consists in choosing spl=0, which results in uniform routing. As the buffer size decreases, the optimal values for spl decrease and, when the buffer goes to zero, the optimal value for spl is the maximum, which results in shortest path routing.

4.7 Summary

In this chapter, we focused on the analysis and design of routing algorithms in networks where nodes have a limited buffer space. We proposed alternative approximation models to the usual Jackson's Theorem to obtain a more accurate distribution on the queue size at the nodes. Using these approximations, we derived the routing condition to minimize overflow losses: nodes receive all traffic exclusively from one neighbor. According to this condition, we designed the routing algorithms that maximize the throughput per node for networks with finite buffers at the nodes under three different communication models: uniform communication, central data gathering, and border data gathering.

Note that the routing condition to minimize packet overflow implies the use of single path routing between any source destination pair. As we showed in Chapter 2, single path routing is very sensitive to network unreliability. This suggests a trade-off between capacity and robustness that will be explored in Chapter 5.

In this chapter, as well as in the previous chapter, we analyzed the problem of routing restricted to lattice networks, where a theoretical analysis is still feasible. A natural extension is to consider more general random networks, which is the object of Chapter 6.

Chapter 5

Routing in Unreliable Networks

5.1 Introduction

Sensor networks consist of simple devices with very limited power and processing capabilities which present a high degree of unreliability. Temporary node failures are commonly present with a certain "recovery" period after which, nodes start working again. Besides nodes going up and down, links are also unreliable and temporary link failures are also frequent in wireless networks.

Routing in large and unreliable networks becomes prohibitively complex in terms of both computation and communication: the set of available routes between any two nodes changes randomly due to random node failures. To overcome this complexity, we already proposed in Chapter 2 the use of fully decentralized routing algorithms where routing decisions at each node are only based on local information. To avoid explicit route discovery or repair computations, and to avoid maintaining explicit state information, we proposed there to use a decentralized routing through a constrained random walk.

In Chapter 2, we studied the routing problem from a single source-destination perspective. We provided a fully distributed routing algorithm with a load balancing property: nodes located at the same distance from the source must carry the same traffic. This implies that all the available routes between the source and the destination are used with a certain probability. When the network becomes unreliable, we showed that this uniform traffic distribution performed better than other routing schemes.

Then, we analyzed the routing problem for multiple sources and/or destinations and characterized the routing algorithms that maximize the rate per node. For any source-destination pair, these routing algorithms required using only a few of the available paths between the source and the destination. For instance, in the case of uniform communications, a simple routing algorithm that maximizes the rate per node is *row-first*, which uses only the two most external paths between the source and the destination. For the data gathering case, the optimal routing uses only one path. Equivalently, we showed in Chapter 4 that when the buffers at the nodes are constrained to be finite, the optimal routing strategy to minimize packet overflow consisted in making packets flow along a unique path.

This suggests that, in the case of unreliable networks, there exists a trade-off between the maximum achievable rate per node and robustness, being more remarkable in the case of networks with small buffers at the nodes. On the one hand, to overcome packet losses

due to the network unreliability, packets should be distributed among the available paths. On the other hand, to increase the rate per node and reduce packet losses due to overflow losses, packets are constrained to use only a few of these paths. The subject of this chapter is to explore this trade-off in the context unreliable random networks with multiple sources multiple destinations.

In particular, we propose to use a family of randomized routing algorithms which are obtained as a convex linear combination of two routing algorithms, the first one being optimal when there are no node failures and the second one being appropriate when the probability of node failure is high. For a given node failure rate, we show that a specific combination achieves the best maximum rate per node.

The rest of the chapter is structured as follows: In Section 5.2, we start by introducing the two failure models we consider to model sensor networks unreliability. Then, in Section 5.3, we analyze the routing algorithms that achieve the maximum rate per node depending on the degree of unreliability of the network. Finally, in Section 5.4, we present simulation results.

5.2 Model and definitions

The network model we consider in this chapter is identical to the model described in Chapter 3 with the additional constrained that nodes are subject to failures. We consider two different models for the temporary node failures. In the first model, we assume that nodes switch between on and off states over time, independently for each node, following a Markov rule with transition probabilities p_0 and p_1. That is, the stationary probability of a node being off, and thus its associated links, is given by $P_{OFF} = \frac{p_0}{p_1+p_0}$. This model tries to capture node failures due to any malfunction in the node.

The second model assumes that a node failure depends on how frequently it is used for routing. For this purpose, we initially assign to each node a given energy and assume that energy consumption in a node is proportional to the number of packets transmitted and received by the node. Assuming that nodes are powered by a renewable source of energy, once the energy of a node is exhausted, it is refueled after a "recovery" period. We model this recovery period as a random variable with a geometric distribution. This model is relevant, for example, for sensor networks with solar panels.

Recall that, for a given routing algorithm Π, we denote by $F^{\Pi}(d_i, d_j, d_k)$ the traffic generated at node d_i with destination node d_j that flows through node d_k according to a particular routing algorithm Π.

Definition 5.1 *Given a $N_1 \times N_1$ grid network, let $d_s = [0, 0]$ and $d_d = [N_1 - 1, N_1 - 1]$ be the two most distant nodes. We define the fairness $f(\Pi)$ of a routing algorithm Π as:*

$$f(\Pi) = \frac{1}{1 + \frac{1}{N_1} \sum_{d_i \in V} \left(F^{\Pi}(d_s, d_d, d_i) - F^{uni}(d_s, d_d, d_i)\right)^2}, \tag{5.1}$$

where $F^{uni}(d_s, d_d, d_i)$ is the uniform node-to-node traffic distribution, that is, according to the notation introduced in Chapter 2, $F^{uni}(d_s, d_d, d_i) = \frac{1}{|Diag(d_i)|}$.

The fairness of a routing algorithm is a measure of how uniform the node-to-node distribution is.

We assume a uniform communication model where all nodes transmit information with a constant average rate and the probability of any node communicating to any other node in the

network is the same for all pairs of nodes. In the next section, we study routing algorithms that maximize this rate per node under certain decentralization conditions.

5.3 Routing algorithms for unreliable networks

In dynamic networks the maximum achievable rate per node depends on two factors: the limited connectivity, which determines the static network capacity, and the packet losses due to node failures and overflow losses.

In Chapter 2 we proposed a distributed randomized routing algorithm for achieving robustness against failures and maximum path diversity with very low computational complexity and minimal state information. This routing algorithm is based on the idea that when nodes in the network can fail at any moment and sources have no state information about the network, the best one can do is to distribute the traffic as uniformly as possible among all the nodes in the shortest path region. The load distribution that this algorithm induces for any pair of source-destination nodes is the most uniform possible: two nodes in the shortest path region which are located at the same distance from the source carry the same load. We denote this routing algorithm by *spreading*. Notice that $f(spreading) = 1$. The spreading algorithm is proposed based on the intuition that under failures, distributed communications are more robust. Moreover, we showed through simulations that when the network becomes unreliable, spreading performed better than other routing schemes.

However, according to Chapter 3, in the multiple sources and/or destinations scenario, spreading is not a good strategy if we want to maximize the rate per node when there are no failures in the network, in which case, row-first is optimal. On the other hand, row-first routing is obviously not a good strategy when failures are present in the network: it routes packets using only the two most external paths of the entire shortest path region, thus not taking advantage of the full diversity in the network, and consequently, being very vulnerable to failures.

In other words, while row-first achieves the most uniform load distribution in the network among the shortest path routing algorithms, the source-to-destination traffic is distributed over very few nodes (only over the edge nodes). On the contrary, the spreading algorithm achieves the most uniform source-to-destination load distribution while the load distribution in the network is not optimal. This is illustrated in Figure 5.1, which shows the network and source-to-destination distributions induced by row-first and spreading routing algorithms. This illustrates also the existing trade-off between robustness and maximum rate per node.

In the case of unreliable networks, we propose to use a routing algorithm that adapts to the failure rate of the network by controlling the shape of the source-to-destination load distributions. Clearly, if the failures in the network are nonexistent or very rare, the best solution is to use a very confined source-to-destination distribution (low fairness) to maximize the rate per node. As the failure rate increases and the network becomes more unreliable, the source-to-destination distribution should be wider (high fairness) to avoid inoperative nodes and overcome the network unpredictability. This will cause a reduction in packets losses due to node failures, which will also contribute to increase the rate per node. However, at the same time, this will generate a suboptimal overall traffic distribution that reduces the rate per node. Therefore, for a given failure rate, there is an optimal source-to-distribution shape that achieves the maximum rate per node.

Both routing algorithms, spreading and row-first, can be defined in terms of the forward-

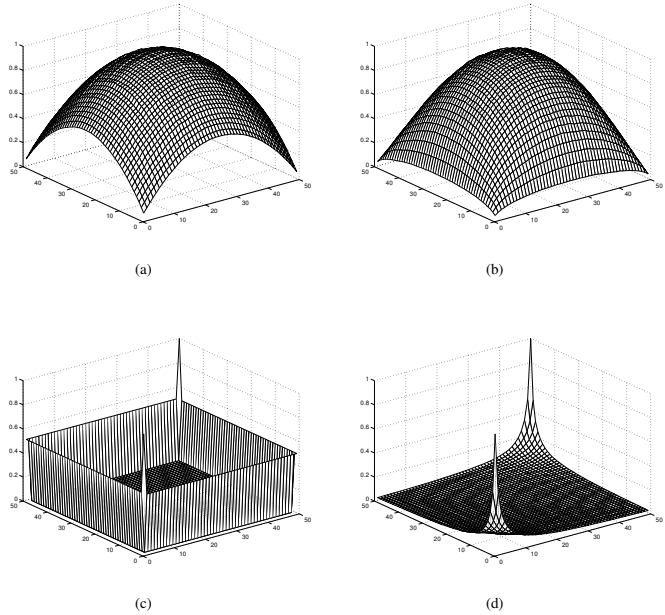


Figure 5.1: Load distributions in a 50×50 square grid network: Top: load distribution in the network (uniform communication model). Bottom: source-to-destination load distribution for nodes $[0, 0]$ and $[49, 49]$. Left: row-first routing. Right: spreading routing. The x and y axis represent the lattice coordinates of a node. The z-axis represents the number of packets carried by node $[x, y]$ normalized such that the maximum load is 1. The load distribution generated in the network by row-first (a) is more uniformly distributed than the distribution generated by spreading (b) while the source-to-destination load distribution induced by spreading (d) is as uniform as possible and the distribution generated by row-first (c) is concentrated only in very few nodes.

ing probabilities of a constrained random walk at any given node. Note that in this particular topology, and under the shortest path condition, a random walk is defined by a single number p, the probability of choosing one of the two links in the shortest path towards the destination. By convention, we define $p(d_k)$ to be the probability that node d_k forwards a packet to its neighbor that is closer to the boundary of the lattice ($1 - p(d_k)$ being the probability of

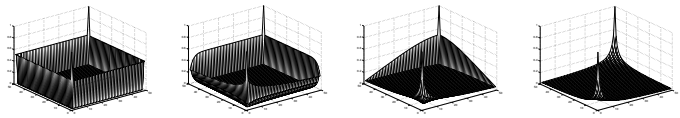


Figure 5.2: Source-to-destination distribution generated by constrained spreading with values of $\phi = [0.0.25, 0.75, 1]$ (from left to right).

forwarding to the other). We showed in Chapter 2 that the local parameters of the spreading algorithms for any node $d_k = [i_k, j_k] \in V$ were given by:

$$p(d_k) = \begin{cases} \frac{|Diag(d_k)|-h_e(d_k)}{|Diag(d_k)|+1}, & \text{if} \quad i_k + j_k < N_1 - 1 \quad \textit{(expansion stage)}, \\ \frac{h_e(d_k)}{|Diag(d_k)|-1}, & \text{if} \quad i_k + j_k \geq N_1 - 1 \quad \textit{(compression stage)}. \end{cases} \tag{5.2}$$

In the same way, we can easily define the local parameters of row-first as follows:

$$p(d_k) = \begin{cases} 1, & \text{if} \quad i_k + j_k > 0, \\ 1/2, & \text{if} \quad i_k + j_k = 0. \end{cases} \tag{5.3}$$

To control the shape of the source-to-destination distribution we use a convex linear combination of the two limit cases, i.e., spreading and row-first, which in terms of forwarding probabilities is defined as:

$$p(d_k) = \begin{cases} (1-\phi)\frac{|Diag(d_k)|-h_e(d_k)}{|Diag(d_k)|+1} + \phi, & \text{if} \quad i_k + j_k < N_1 - 1 \quad \textit{(expansion stage)}, \\ (1-\phi)\frac{h_e(d_k)}{|Diag(d_k)|-1} + \phi, & \text{if} \quad i_k + j_k \geq N_1 - 1 \quad \textit{(compression stage)}. \end{cases} \tag{5.4}$$

where ϕ is the parameter that determines the fairness of the routing algorithm, and consequently, the shape of the source-to-destination and the network load distributions. We denote this routing algorithm as *constrained spreading*. If $\phi = 0$, constrained spreading generates a very constrained source-to-destination traffic distribution and the most uniform traffic distribution in the network. As we increase ϕ, the source-to-destination traffic distribution becomes wider and the network distribution less uniform. In the limit, when $\phi = 1$, the source-to-destination traffic distribution is uniform. Figure 5.2 shows the source-to-destination distribution generated by constrained spreading with values of $\phi = [0, 0.25, 0.75, 1]$.

Note that in the case of the torus grid network, since spreading routing is also space invariant, it also achieves capacity when there are no node failures in the network, i.e., it generates a uniform overall load distribution. Consequently, the optimal routing algorithm in dynamic networks is just to use spreading routing.

5.4 Simulation results

In this section we analyze through simulations the performance of the routing algorithms family proposed in previous section. For completeness, we compare also the performance

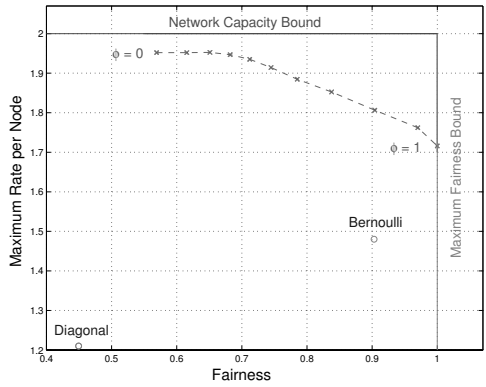
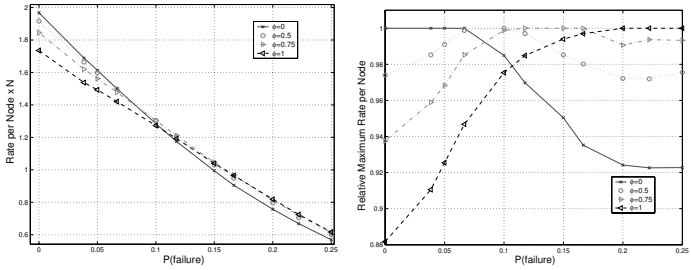


Figure 5.3: Maximum rate per node - fairness trade-off for a 34×34 nodes square grid network. The x axis denote fairness and y axis the maximum rate per node achieved. We represent both quantities for different values of ϕ, from 0 to 1 with 0.1 interval size.

of some other shortest path space invariant routing policies used in square grid networks: *Diagonal* and *Bernoulli* routing algorithms. Diagonal routing [6] is a probabilistic routing strategy which states that propagation of messages toward the diagonal should be given preference where possible, where diagonal denotes the set of nodes that are an equal number of rows and columns away from the destination node. Bernoulli routing consists of flipping a fair coin to decide which of the two feasible neighbors on a next hop to pick at each node.

Figure 5.3 shows the trade-off between maximum achievable average rate per node and fairness (5.1) for Bernoulli, Diagonal, and constrained-spreading routing algorithms for different values of ϕ in a 34×34 nodes square grid network with a buffer size per node $Q = 100$, where ϕ takes values in the interval $[0, 1]$ with a step size of 0.1. Notice that constrained-spreading routing algorithms clearly outperform both Bernoulli and Diagonal, in fairness and maximum rate per node achieved. Observe also that when $\phi = 0$, that is, we have a pure row-first routing, we achieve the maximum rate per node, however the fairness is low. On the other hand, for $\phi = 1$, we have a spreading routing algorithm and consequently the fairness is maximized while the rate per node decreases importantly. Between these two extremes, we obtain intermediate results for different values of ϕ.

We analyze now the two dynamic network models we described in Section 5.2. First we consider the case of dynamic networks based on the Markov failure model. We fix the transition probability $p_{ON \to OFF} = 0.01$ and make the transition probability $p_{OFF \to ON}$ take values in the interval $[1, 0.01]$ such that p_{OFF} varies linearly in the interval $[0, 0.25]$. Figure 5.4 shows the rate per node achieved by different constrained-spreading routing algorithms characterized by different values of ϕ in a 34×34 square grid network with buffers of size $Q = 100$. The simulation time was 600,000 time slots. Figure 5.4(b) shows the rate per



(a) (b)

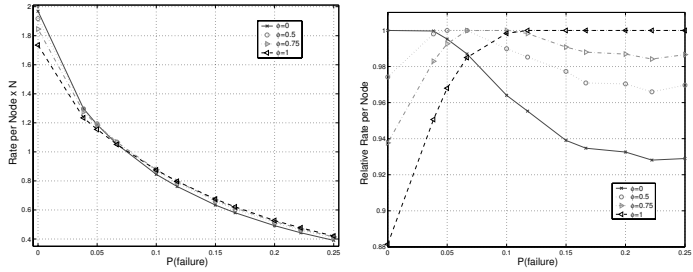
Figure 5.4: Average performance of constrained-spreading for different values of ϕ under the Markov failure model in a 32×32 square grid network. (a) Maximum rate per node. (b) Maximum rate per node relative to the best rate achieved by any of the routing algorithms.

node relative to the best rate achieved by any of the considered routing algorithms for each given value of p_{OFF}.

Notice that, if the network is static ($p_{OFF} = 0$), the best strategy is to choose $\phi = 0$ (row-first routing), that achieves a rate per node 12% higher than the $\phi = 1$ strategy. However, as the unreliability of the network increases, the performance for $\phi = 0$ degrades rapidly. For very unreliable networks (values of p_{OFF} close to 0.15 or above), the best routing algorithm consists in choosing $\phi = 1$ (spreading), which achieves a rate per node 8% higher than the $\phi = 0$ strategy. We observe also, that depending on the values of p_{OFF}, the optimal values for ϕ are different.

We repeated the same experiment considering the energy based failure model. We consider that the energy of a node is depleted after sending 500 information packets. Then, after a "refueling" period, nodes become active again. The results are presented in Figure 5.5. In this case, the $p_{OFF \to ON}$ is going to determine the "refueling" period. As the "refueling" period increases, the network becomes more unreliable, given that more nodes would be inoperative waiting for more energy supply. Figure 5.5(a) shows the rate per node achieved by different routing algorithms under different network conditions and Figure 5.5(b) shows the relative rate per node.

Observe that the behavior of the routing algorithms under both failure models is very similar: for very short "refueling" periods, low values for ϕ achieve better results, while large "refueling" periods calls for large values for ϕ. However, in the energy based model, the rate per node decreases more rapidly. This is due to the fact that in the square grid network, nodes situated in the center of the network have to deal with a higher traffic load (see Figure 5.1). Therefore, their energy is depleted more rapidly than any other node, resulting in more frequent temporary failures. However, the relative performance of constrained-spreading routing algorithms remains quite similar.



(a) (b)

Figure 5.5: Average performance of constrained-spreading for different values of ϕ under the energy failure model in a 32×32 square grid network. (a) Maximum rate per node. (b) Maximum rate per node relative to the best rate achieved by any of the routing algorithms.

5.5 Summary

In this chapter, we showed that achieving robust communications and maximizing the achievable rate-per-node are incompatible goals: while robust communications require the use of as many paths as possible between the source and the destination, maximizing the rate per node requires using only a few of the available paths. To illustrate this trade-off, we studied routing algorithms that maximize the rate per node for dynamic networks under two different failure modes: a Markovian model that reproduces failures due to malfunction in the nodes, and an energy limited model which is related to depletion of communication resources. We proposed to use a particular combination of two routing algorithms, namely row-first and spreading, row-first being the routing algorithm that achieves capacity in the static case, and spreading being the most robust algorithm against failures. The combination of these two routing algorithms defines a family of randomized routing algorithms, each of them being suitable for a given probability of node failure. All the routing algorithms that we proposed are fully decentralized and can be easily distributed.

Chapter 6

Data Gathering in Random Networks

6.1 Introduction

Most of the routing algorithms we proposed in previous chapters rely in one way or another on the regular topology of the network model we considered. For instance, the optimal routing algorithms we derived in previous chapters to reduce overflow losses were specially designed for the square grid. Motivated by the insight gained in grid networks, we tackle now the problem of routing in random networks.

Throughout this chapter, we assume that either the network is wired, or that if it is wireless, there exists either a transmission schedule or a frequency division multiplexing that avoids conflicts. Therefore, we abstract the random network as a random graph with point-to-point links and transform the problem into a graph with nearest neighbor connectivity. Under this hypothesis, nodes can simultaneously receive multiple packets from different neighbors and, at the same time, transmit one packet to the next hop. If a node receives or generates locally a new packet while transmitting another packet to the next hop, the newly arrived packet is temporarily stored in the buffer until there is an opportunity to transmit it.

As we showed in grid networks, when nodes have a limited buffer for the temporary storage of packets, the maximum achievable rate is considerably reduced with respect to the infinite buffer case due to packet overflow. As explained before, the analysis of this maximum achievable rate requires computing the buffer occupancy distribution in the nodes, which is not feasible without some approximations. Although random networks are very different in nature from grid networks, we show that similar principles can be applied to analyze this buffer occupancy.

We first analyze the case where nodes have unconstrained memory: we study the network capacity and optimal routing algorithms for infinite queues. We show that the problem of finding the optimal routing algorithm that achieves network capacity is actually an NP-hard problem and propose a distributed approximation algorithm that behaves close to optimal. Then, using similar techniques as in Chapter 3, we analyze the effect of finite buffers on the network and show that finding the optimal routing algorithm is also an NP-hard problem. We propose an approximation algorithm to minimize overflow losses that achieves a maximum

rate at least three times higher on average than the rate achieved with the usual Shortest Path Routing algorithm. We show that this routing algorithm requires combining adequately shortest path routing and traveling salesman routing.

Finally, we discuss also the *wireless* case, where packet transmissions among different nodes can interfere and result in packet collisions. We show that to achieve network capacity, it is necessary to construct a transmission schedule which is an NP-complete problem. We propose a simple alternative method that consists in modifying the media access control to deal with finite buffer and reduce packet overflow.

The rest of the chapter is structured as follows: In Section 6.2, we introduce the random network model. In Section 6.3, we analyze capacity limits of random networks. In Section 6.4, we investigate the infinite buffer case and provide routing algorithms that maximize the achievable rate per node. In Section 6.5, we analyze optimal routing algorithms that minimize packet overflow for the finite buffer case. Finally, in Section 6.6, we analyze wireless random networks for both finite and infinite buffers.

6.2 Model and definitions

We randomly deploy N nodes on a square area following a uniform distribution. We assume that each node is a source of data as well as a relay for some other sources to reach the destination. We will use nodes, sources, and sensors interchangeably in subsequent discussions. We assume that nodes use a fixed transmission power and achieve a fixed transmission range R_{tx}. We denote the euclidean distance between any two nodes d_i and d_j as $s(d_i, d_j)$.

We assume that there exists a transmission schedule or frequency division multiplexing that avoids conflicts: if the euclidean distance between two nodes is less than the communication radio R_{tx}, we assume that there exists a link between both nodes with capacity C_l. Hence, we transform the problem into a graph with nearest neighbor connectivity.

We denote by $\varphi^n(d_i)$ the set of nodes that are located inside the circle of radius R_{tx} and center d_i, i.e., $\varphi^n(d_i) = \{d_j : s(d_i, d_j) \leq R_{tx}\}$. Equivalently, $\varphi^n(d_i)$ is the set of neighbors of d_i in the connectivity graph. We assume that nodes are aware of their neighbor nodes, but not of their absolute positions in the network. We denote as $n(d_i)$ the number of neighbors of d_i, that is: $n(d_i) = |\varphi^n(d_i)|$. For a fixed network size N, we choose the surface dimensions such that the average number of neighbors per node is equal to n_{avg}.

We consider the *central data gathering* communication model, in which all nodes send their data to one common fixed node d_{BS}, denoted as base station, and located at the center of the square area. This scenario corresponds to the case where one node (base station) collects the information generated by all the nodes in the network [34].

We assume that nodes generate packets uniformly with a rate of $\mathcal{R}$ packets per second. More specifically, nodes generate one packet in every interval of duration $(1/\mathcal{R})$ seconds. Within an interval, the exact packet origination time is uniformly distributed from 0 to $(1/\mathcal{R})$, independently among nodes. This models a sensor network that regularly measures a certain phenomenon without node synchronization.

A routing algorithm Π defines how traffic flows from any source to the base station d_{BS}. We consider the class of single path routing algorithms, that is, all the packets generated at any node follow one single route to d_{BS}. We assume that routing algorithms are time invariant, that is, Π does not change over time. We also assume that nodes forward all packets to one single neighbor or *next hop* regardless the source of the packets. This allow us to completely

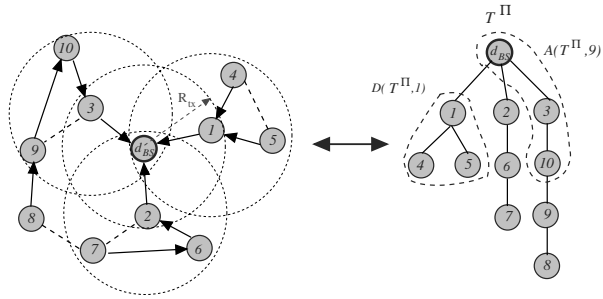


Figure 6.1: We represent the random network with a connectivity graph $G_c = (V, E)$, where the edges represent that the distance between two nodes is less than the transmission range (dashed lines). Nodes forward all packets to one single neighbor determined by the routing algorithm Π (arrows). Any routing algorithm Π is completely characterized by its data gathering tree (T^{Π}).

characterize any routing algorithm Π by its Data Gathering Tree (DGT) T^{Π} with d_{BS} as root node. For illustration purposes, Figure 6.1 shows a routing algorithm and its corresponding data gathering tree. Shortest path routing algorithms are those where packets transmitted between any node d_i and d_{BS} always follow a path with the minimum number of hops, in which case, T^{Π} is a *shortest path tree* (SPT).

Given that nodes have a single parent node towards d_{BS}, the traffic that flows through a given node d_i can also be determined by the number of children of d_i in the data gathering tree. We denote by $\mathcal{D}(d_i, T^{\Pi})$ the subtree that has d_i as root node which consists of all the nodes that send their traffic through d_i according to the data gathering tree T^{Π}, or in other words, the set of descendants of d_i in T^{Π}. Similarly, we denote by $\mathcal{A}(d_i, T^{\Pi})$ the set of nodes that relay the traffic sent by d_i towards d_{BS} according to T^{Π}, or in other words, the ancestors of d_i in T^{Π}. An example of both subtrees is depicted in Figure 6.1.

The length of a path is defined as the number of hops in that path. Moreover, we denote by $h(d_i, d_j)$ the length of the shortest path between nodes d_i and d_j. We denote by $\overline{L}^{\Pi}(N)$ the average number of hops that packets take to reach the base station when nodes route packets according to Π and by $\lambda^{\Pi}_{d_k}$ the traffic arrival rate to node d_k according to Π.

We assume that nodes are equipped with buffer capabilities for the temporary storage of Q packets. When packets arrive at a particular node or are generated by the node itself, they are placed into a queue until the node has the opportunity to transmit them to the next hop. If the buffer is full, the newly arrived packet is dropped.

Note that the definitions of network capacity and maximum achievable rate introduced before are also valid for random networks:

Definition 6.1 *An random network of N nodes with infinite buffers is said to have capacity $C_{\{w,c\}}(N)$ if any rate $\mathcal{R} = C_{\{w,c\}}(N) - \epsilon,\ \forall\ \epsilon > 0$, is achievable. Moreover, any rate*

$\mathcal{R} \geq C_{\{w,c\}}(N)$ *is not achievable.*

The subscripts "w" and "c" indicate the network model, the wired or collision-free case and the wireless or collision model respectively.

Definition 6.2 *Given a random network of N nodes with finite buffers of size Q, a routing algorithm Π is said to have maximum achievable rate $\mathcal{R}^{\Pi}_{sup}(N,Q)$ if any rate $\mathcal{R} = \mathcal{R}^{\Pi}_{sup}(N,Q) - \epsilon, \quad \forall\ \epsilon > 0$, is feasible using Π as routing algorithm. Moreover, any rate $\mathcal{R} \geq \mathcal{R}^{\Pi}_{sup}(N,Q)$ is not feasible.*

The goal is to design scalable routing algorithms, with all the decentralization properties discussed in Chapter 2, that for a given buffer size Q, minimize packet losses due to buffer overflow. We start by studying the network capacity of random networks.

6.3 Network capacity

In the case of infinite buffers, the network capacity is limited by the nodes that support most traffic. For the central data gathering communication model, the most loaded nodes are clearly the neighbors of the base station, which have to relay the traffic generated by the whole network towards d_{BS}. Applying bisection arguments [40; 50] to these nodes, the network capacity $C_w(N)$ is upper bounded as:

$$C_w(N) \leq \left\lceil \frac{n(d_{BS})C_l}{N-1} \right\rceil . \tag{6.1}$$

As in the regular lattice case, the network capacity decreases linearly with the total number of nodes N. This linear decrease is due to the bottleneck at the base station inherent in the data gathering communication model. However, as opposed to regular topologies, this upper bound is not always achievable in random networks, and in general, the network capacity depends on the specific network topology.

6.4 Routing algorithms for infinite buffers

The maximum achievable rate $\mathcal{R}^{\Pi}_{\text{sup}}(N,\infty)$ under the assumption of infinite buffers can be easily derived by applying the stability condition in the most loaded node d_m. Given the centralized structure of the traffic matrix, and the fact that we consider single path routing algorithms, it is clear that $d_m \in \varphi^n(d_{BS})$. Noting also that all the traffic transmitted by d_m has to reach d_{BS} by a single link of capacity C_l, we have:

$$\mathcal{R}^{\Pi}_{\text{sup}}(N,\infty) = \min_{d_i \in \varphi^n(d_{BS})} \frac{C_l}{|\mathcal{D}(d_i, T^{\Pi})|}. \tag{6.2}$$

6.4.1 Optimal routing algorithms

The optimal routing algorithm Π_{opt} that maximizes $\mathcal{R}^{\Pi}_{\text{sup}}(N,\infty)$ is such that the total arrival traffic to d_{BS} is distributed as uniformly as possible among the nodes in $\varphi^n(d_{BS})$. Equivalently, Π_{opt} minimizes the most loaded node in $\varphi^n(d_{BS})$:

$$\Pi_{\text{opt}} = \arg\max_{\Pi} \left(\min_{d_i \in \varphi^n(d_{BS})} \frac{C_l}{|\mathcal{D}(d_i, T^{\Pi})|} \right) = \arg\min_{\Pi} \left(\max_{d_i \in \varphi^n(d_{BS})} |\mathcal{D}(d_i, T^{\Pi})| \right). \tag{6.3}$$

The maximum network capacity is achieved if the load is uniformly distributed in $\varphi^n(d_{BS})$, that is:

$$\max_{d_i \in \varphi^n(d_{BS})} |\mathcal{D}(d_i, T^{\Pi_{\text{opt}}})| = \left\lceil \frac{N-1}{n(d_{BS})} \right\rceil .$$

However, depending on the network topology, this upper bound is not always achievable.

Theorem 6.1 *The algorithm to find the optimal routing algorithm that maximizes the achievable rate per node in a random network with infinite buffers is NP-complete.*

Proof: The optimal routing Π_{opt} is such that $T^{\Pi_{\text{opt}}}$ consists of $n(d_{BS})$ disjoint trees of maximum size $\left\lceil \frac{N-1}{n(d_{BS})} \right\rceil$ whose roots are each of the neighbors of d_{BS}. Note that the problem of identifying these disjoint trees is a particular case of the Bounded Component Spanning Forest (BCSF) [23]. Given a graph $G = (V, E)$, a weight $w(v) \in Z_0^+$ for each $v \in V$ and two positive integers $K \leq |V|$, and B, the BCSF problem consist in finding a partition of V into $k \leq K$ disjoint sets $V_1, V_2, \ldots, V_k$ such that, for $1 \leq i \leq k$, the subgraphs of G induced by V_i are connected and the sum of the weights of the vertices in V_i does not exceed B. The BCSF problem is NP-complete even if all the weights equal 1 and B is any fixed integer larger than 2. Finding the optimal routing algorithm Π_{opt} is a BCSF problem for $w(v) = 1$, $B = \frac{N-1}{n(d_{BS})}$, and $k = n(d_{BS})$. □

6.4.2 Approximation algorithms

The usual distributed algorithm to compute the DGT in a sensor network is the distributed Bellman-Ford (DBF) algorithm [10]. The DBF algorithm computes the shortest path from any node to the base station d_{BS} by making nodes select as their next hop the neighbor that is located closer to d_{BS} in number of hops. As we will see later, the DBF algorithm generally leads to a highly unequal distribution of the traffic in $\varphi^n(d_{BS})$. In the following, we propose a simple distributed approximation routing algorithm whose DGT equalizes the load as much as possible among the nodes in $\varphi^n(d_{BS})$.

Given any routing algorithm Π, we can decompose T^{Π} into $n(d_{BS})$ disjoint subtrees, $ST_{d_i} = \mathcal{D}(d_i, T^{\Pi}),\ d_i \in \varphi^n(d_{BS})$, each subtree associated to a different node in $\varphi^n(d_{BS})$. We define the weight $w_{\Pi}(d_j)$ of a node d_j as the number of nodes that d_j relays towards d_{BS} (including itself) according to the routing algorithm Π, that is, $w_{\Pi}(d_j) = |\mathcal{D}(d_j, T^{\Pi})|$. With a slight abuse of notation, we define the weight of a subtree as the weight of its root node, that is, $w_{\Pi}(ST_{d_i}) = w_{\Pi}(d_i) = |\mathcal{D}(d_i, T^{\Pi})|$. Given a node d_j, we define as $ST^{-1}(d_j)$ the subtree to which d_j belongs. That is:

$$ST^{-1}(d_j) = ST_{d_i} : d_j \in ST_{d_i}.$$

The goal is to define an algorithm whose associated DGT is such that the weights of all subtrees are as similar as possible. We propose a distributed algorithm based on the DBF algorithm with a different cost matrix than the hop distance to d_{BS}.

Each node d_j maintains two costs, $c_1(d_j)$ and $c_2(d_j)$: $c_1(d_j)$ is its own weight, that is, $c_1(d_j) = w_{\Pi}(d_j)$; $c_2(d_j)$ is the weight of the subtree the node is connected to, that is, $c_2(d_j) = w_{\Pi}(ST^{-1}(d_j))$. Initially, all subtrees ST_{d_i} consist of only their root nodes, that is, $ST_{d_i} = \{d_i\},\ d_i \in \varphi^n(d_{BS})$. Consequently, all the nodes in $\varphi^n(d_{BS})$ are initialized with an individual weight of one and a subtree weight of one, i.e., $c_1(d_i) = 1, c_2(d_i) = 1$ for all

$d_i \in \varphi^n(d_{BS})$. The rest of the nodes are not connected, and therefore, they are initialized with an individual weight $c_1(d_j) = 1$ (just their own weight) and a subtree weight equal to $c_2(d_j) = \infty$ (to indicate that nodes are not connected). Nodes try to locally equalize the weight of the subtrees as much as possible by choosing as next hop the neighbor with the minimum subtree weight. More precisely, a node d_j chooses its next hop $\nu(d_j)$ as follows:

$$\nu(d_j) = \underset{d_k \in \varphi^n(d_j)}{\arg\min} \; (c_1(d_j) + c_2(d_k)) \,. \tag{6.4}$$

In other words, d_j joins the same subtree as $\nu(d_j)$. Note that when d_j joins a new subtree ST_{d_N}, it also implies that all the nodes that d_j is relaying, also join the same subtree. The weights of all the nodes belonging to the subtree $ST_{d_N} = ST^{-1}(\nu(d_j))$ have to be updated with the weight of the new nodes. Particularly, we update the individual weights c_1 of $\nu(d_j)$ and all its ancestors and the subtree weight c_2 of all the nodes belonging to ST_{d_N} as follows:

$$c_1(d_k) = c_1(d_k) + c_1(d_j) \text{ for all } d_k \in \mathcal{A}(d_j, ST_{d_N}), \tag{6.5}$$

$$c_2(d_k) = c_2(d_k) + c_1(d_j) \text{ for all } d_k \in ST_{d_N}. \tag{6.6}$$

Similarly, if d_j is leaving a subtree ST_{d_O} it was connected to, to join a new subtree with a lower weight, it is also necessary to update the weights of the nodes belonging to ST_{d_O} by subtracting the weight of the leaving node d_j. That is:

$$c_1(d_k) = c_1(d_k) - c_1(d_j) \text{ for all } d_k \in \mathcal{A}(d_j, ST_{d_O}), \tag{6.7}$$

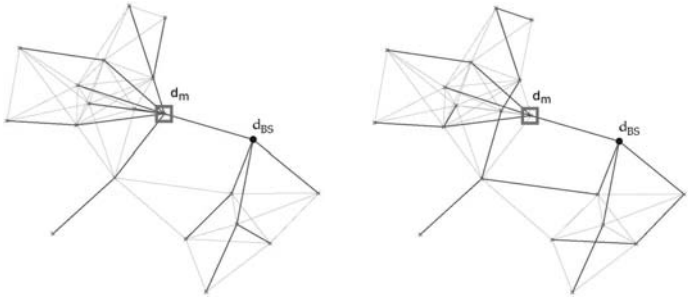
$$c_2(d_k) = c_2(d_k) - c_1(d_j) \text{ for all } d_k \in ST_{d_O}. \tag{6.8}$$

It can be verified that this algorithm converges in a finite time to a solution. The proof goes along the same line as the proof of the convergence of the DBF algorithm [10]. We denote this routing algorithm as Uniform data gathering Traffic Distribution (UTD) algorithm. Note that UTD does not take into account the distance in hops to the base station to build the DGT, but only the weight of each subtree. Therefore, it does not necessarily lead to shortest path routing.

We can easily consider a shortest-path version of the UTD algorithm by forcing nodes to choose the next hop only among the neighbors which are closest (in the hops metric) to the base station. We denote this algorithm as UTD-SP routing. Out of the multiple possible shortest path trees of a random network, DBF gives randomly one, while UTD-SP tries to compute the one that distributes the traffic as uniformly as possible. By constraining the neighbors that can be chosen as next hop, the traffic distribution achieved by UTD-SP is obviously less uniform than in the unconstrained case, i.e., UTD.

Note that both routing algorithms can also be applied in the case of multiple data collectors and for different base station locations. While considering multiple base stations has the effect of increasing the achievable rate per node, placing the sink near the border of network reduces it. In both cases, UTD and UTD-SP equalize the load as much as possible among the neighbor nodes of the base stations.

For illustration purposes, we show the DGT generated by the DBF and UTD algorithms for a single realization of a 20 nodes random network in Figure 6.2. Note that the UTD algorithm distributes the load more uniformly among the nodes in $\varphi^n(d_{BS})$ even if non shortest paths are needed. In this example, using the DBF algorithm, the capacity will be limited by the node d_m which relays the traffic of 12 nodes, while using UTD, this load has been reduced to 6 nodes. On the other hand, the length of the longest path using the DBF algorithm is of two hops, while for the UTD algorithm is 6 hops.



(a) (b)

Figure 6.2: Data gathering in a 20 nodes random network: DGT generated by (a) DBF and (b) UTD algorithms.

6.4.3 Simulation results

To analyze the performance of the different routing algorithms in random networks with finite buffers, we compare the traffic distribution generated in $\varphi^n(d_{BS})$. In particular, we are only interested in the most loaded node d_m, that is, the node that limits the maximum achievable rate. We define the distribution factor $DF(\Pi)$ of a routing algorithm Π as the ratio between the number of nodes that relay traffic in the most loaded node d_m according to Π, that is, $|\mathcal{D}(d_m, T^{\Pi})|$, and the number of nodes that relay traffic in the most loaded node in an optimal uniform traffic distribution. That is, $DF(\Pi) = \frac{|\mathcal{D}(d_m,T^{\Pi})|n(d_{BS})}{(N-1)}$. Clearly, $DF(\Pi)$ depends on the specific network topology, and $DF(\Pi) \geq 1$. Note that this uniform distribution may be even not feasible for some random network topologies. The maximum rate achieved by Π can be written in terms of $DF(\Pi)$ as:

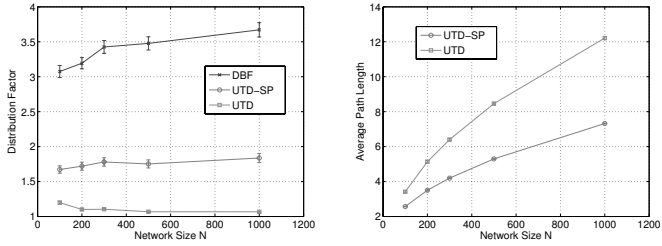
$$\mathcal{R}^{\Pi}_{\text{sup}}(N, \infty) = \frac{C_w(N)}{DF(\Pi)},$$

where $C_w(N)$ is the upper bound given in (6.1).

We compare now the performance of the DBF, UTD, and UTD-SP algorithms in terms of distribution factor and average path length. Figure 6.3 shows the performance of these three routing algorithms as a function of the network size and Figure 6.4 as a function of the average number of neighbors per node.

Figure 6.3(a) and Figure 6.4(a) show the average distribution factor $\overline{DF}$ achieved by the DBF, UTD, and UTD-SP algorithms. Figure 6.3(b) and Figure 6.4(b) show the average path length $\overline{L}$ for the same three algorithms.

First, notice the highly unbalanced traffic distribution achieved by the DBF algorithm and how it degrades with the network size (Figure 6.3(a)) and with the connectivity of the



(a) (b)

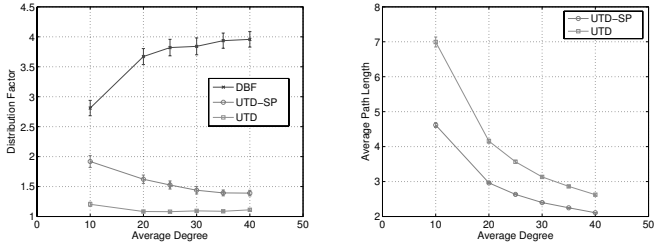
Figure 6.3: Data gathering in a random network with infinite buffers and average number of neighbors per node equal to 15 as a function of the network size N. (a) Average distribution factor $\overline{DF}(\Pi)$, i.e., the ratio between the number of nodes that relay traffic in the most loaded node according to Π and the number of nodes that relay traffic in the most loaded node in an optimal uniform traffic distribution. (b) Average path length $\overline{L}$ in hops.

network (Figure 6.4(a)). The reason is that the nodes that discover first a path towards d_{BS} have a higher probability to be chosen as relay by its neighbors, which will only change their next hop if a node with a shorter path toward d_{BS} is discovered. Therefore, the traffic distribution generated by the DBF algorithm is highly unbalanced. Moreover, this effect is more remarkable in networks with higher average number of neighbors per node, where a node that discovers a shortest path can be selected by a higher number of neighbors.

Note that the smaller $DF(\Pi)$ is achieved, in all cases, by UTD. Moreover, the traffic distribution in $\varphi^n(d_{BS})$ generated by the UTD and UTD-SP algorithms becomes more uniform as the network size increases (Figure 6.3(a)). A similar behavior can be observed with the network connectivity (Figure 6.4(a)): the traffic distribution becomes more uniform as the average number of neighbors per node increases. Clearly, in large and dense networks, it is easier to find a data gathering tree that uniformly balances the load among the neighbors of d_{BS}.

Note that in all cases, the distribution factor achieved by the UTD algorithm is at least three times lower than the distribution factor achieved by the DBF algorithm. Consequently, the maximum rate achieved by UTD is at least three times higher than the maximum rate achieved by using the DBF algorithm.

As expected, the UTD algorithm presents in both comparisons a higher average path length (Figures 6.3(b) and 6.4(b)). However, this difference with the shortest path routing algorithm is bounded in all cases within a constant factor of approximately 1.7. Note that DBF and UTD-SP algorithms are both shortest path routings and, therefore, have the same average path length.



(a) (b)

Figure 6.4: Data gathering in a 200 nodes random network with infinite buffers: (a) average distribution factor $\overline{DF}$, and (b) average path length $\overline{L}$ in hops, as a function of the average number of neighbors per node n_{avg} for 250 random networks.

6.5 Routing algorithms for finite buffers

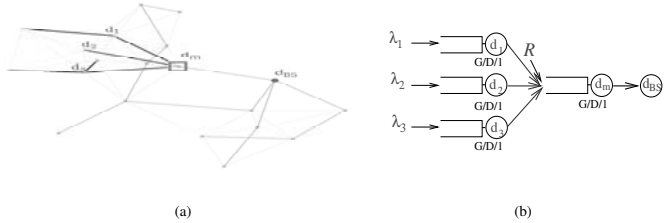
If nodes have a limited space for the temporary storage of packets, the maximum achievable rate is clearly reduced due to the presence of buffer overflows. Moreover, there is no reason to believe that optimal routing algorithms which achieve the maximum rate for infinite buffers will perform well under finite buffers. We already showed in Chapter 4 that, in the case of lattice networks, the necessary conditions for a routing algorithm to be optimal under finite buffers are more restrictive than for infinite buffers.

In this section, we show that the problem of finding the routing algorithm that achieves the maximum rate in a random network with finite buffers is also NP-complete. Then, we propose a simple approximation algorithm that achieves a maximum rate of, at least, three times the maximum rate achieved by the usual DBF algorithm, for any given loss probability.

6.5.1 Optimal routing algorithms

A necessary condition for the optimal routing algorithms Π_{opt} to achieve the maximum rate $\mathcal{R}_{\text{sup}}^{\Pi}(N,Q)$ is to distribute the load uniformly among the nodes in $\varphi^n(d_{BS})$. However, this condition is *not sufficient* under finite buffers. Overflow losses will appear now in the most loaded nodes, reducing the maximum achievable rate.

Computing the maximum achievable rate $\mathcal{R}_{\text{sup}}^{\Pi}(N,Q)$ for a finite value of Q requires analyzing the packet distribution in the queue of the most loaded node d_m, that is, where overflow losses will appear first. To compute the packet distribution in the queue of d_m, it is necessary to analyze the entire queue subtree $\mathcal{D}(d_m, T^{\Pi})$ with d_m as a head node, depicted in Figure 6.5(a). Using the same approximation models for tree networks [53; 8] we used in Chapter 4, this analysis can be reduced to the analysis of a much simpler two stage network formed by considering only nodes located one stage away from d_m and preserving



(a) (b)

Figure 6.5: The packet distribution in the queue of the most loaded node d_m in a random network (a) is the same as in its equivalent two-stage model (b).

the exogenous inputs. The resulting equivalent model is shown in Figure 6.5(b). The packet distribution in the queue of d_m is identical if we consider the entire random network (Figure 6.5(a)) or just the two-stage model (Figure 6.5(b)). But more importantly, using this approximation model, we can also derive the DGT that minimizes overflow losses.

Nodes close to d_{BS} carry a higher load and will suffer first from packet overflow losses, determining the maximum achievable rate $\mathcal{R}_{\text{sup}}^{\Pi}(N, \infty)$. Reducing overflow losses in the most loaded nodes is equivalent to reducing the number of packets in the queue of the nodes, or similarly, to make the packet distribution in their queues confined to small values. As we already showed in Chapter 4, the routing strategy that minimizes the number of packets in the queues of all nodes, and equivalently, minimizes the overflow losses, consists in making nodes receive traffic exclusively from one of their neighbors.

Combining both conditions, the uniform traffic distribution and the overflow loss minimization, Π_{opt} is such that the data gathering tree $G = T^{\Pi_{\text{opt}}}$ it induces consists of $n(d_{BS})$ disjoint subtrees $ST_{d_i} = \mathcal{D}(d_i, G),\ d_i \in \varphi^n(d_{BS})$, where each ST_{d_i} is a path of at most $\left\lceil \frac{N-1}{n(d_{BS})} \right\rceil$ nodes where each node is visited exactly once. In other words, each ST_{d_i} is a Hamiltonian path.

Theorem 6.2 *The algorithm to find the optimal routing algorithm that maximizes the achievable rate per node for a random network with finite buffers is NP-complete.*

Proof: Note that this problem is a particular case of Theorem 6.1 where all subtrees are constrained to be Hamiltonian. An equivalent way to show that this problem is NP-complete is to note that a particular case of this problem (when $n(d_{BS}) = 1$) is equivalent to finding a path starting from d_{BS} that visits each nodes of the network exactly once. If we create a dummy node d_D that is connected to all nodes in the graph, this problem is equivalent to finding a Hamiltonian cycle in this new graph. A Hamiltonian cycle is a closed loop through a graph that visits each node exactly once. The problem of deciding whether a graph has a Hamiltonian cycles is a special case of the traveling salesman problem (TSP), where each pair of nodes with an edge between them has cost 0 and for each missing edge, the cost is 1 [39]. The problem of finding a Hamiltonian path is NP-complete, and the only known way

to determine whether a graph has a Hamiltonian circuit is to undertake an exhaustive search. □

Note that the optimal routing algorithm Π_{opt} for random networks is similar to the optimal algorithm for regular topologies: we showed that the algorithm that minimizes overflow losses in a square grid consisted in distributing the load uniformly among the four arrival links to the base station using a Hamiltonian path within the set of nodes associated to each of these links. The regular topology of the square grid allowed to easily construct Hamiltonian paths. However, in the more general case of random networks, it is an NP-hard problem.

6.5.2 Approximation algorithms

In the following, we introduce a distributed routing algorithm that reduces overflow losses while allocating the load as uniformly as possible among the nodes in $\varphi^n(d_{BS})$.

In general, the heuristics to solve the traveling salesman problem are quite difficult to distribute. A simple and popular heuristic is based on the Minimum Spanning Tree (MST). We start by computing the MST of the set of nodes relaying traffic through a particular neighbor of d_{BS}. In our case, given that the only cost metric is the number of hops, the MST is just the shortest path tree. Then, we do a depth-first search in the MST and generate an ordered list of nodes according to the order in which they were discovered. We generate the routing graph by relaying these vertices in the ordered list by a shortest path. Simple approximation methods exists also for the euclidean TSP where shortest paths are always preferable than indirect paths [39]. However, for the particular communication model we consider, we can design specific algorithms which are less complex and exhibit better performance.

It is important to note that in the data gathering communication model, nodes close to d_{BS} carry more traffic than those nodes located far away, and therefore, they are subject to higher packet overflow losses. As we will show later, most of the packet losses are indeed concentrated in very few nodes close to d_{BS}. From Theorem 4.5, we know that the routing condition to minimize overflow losses consists in making nodes receive all traffic exclusively from one neighbor. Therefore, in order to minimize buffer overflow, this condition has to be specially enforced in the nodes close to d_{BS}. To ensure now a uniform load distribution in $\varphi^n(d_{BS})$, we make nodes located further from d_{BS} route their traffic such that the weight of the subtrees is as equalized as possible. This idea is depicted in Fig 6.6.

Based on this idea, we propose the following distributed routing algorithm. First, using an approximation algorithm, we establish $n(d_{BS})$ disjoint Hamiltonian paths $ST(d_i)$, $d_i \in \varphi^n(d_{BS})$, of length $L_Q < \left\lceil \frac{(N-1)}{n(d_{BS})} \right\rceil$ departing from each neighbor of d_{BS}. To generate these paths, we use a simple incremental method that consists in inserting new points into a partial path, starting from a single vertex, until the path is complete. One simple version of this heuristic that seems to work best is furthest point insertion [39]: among the neighbors of the end point that do not belong to any path (if any), we insert the one that is furthest from d_{BS}. Note that we can not assure that all Hamiltonian paths end with the same number of nodes, and therefore, we can not guarantee an optimal traffic distribution.

Once we have generated the Hamiltonian paths, the nodes that do not belong to any path, forward their traffic preferentially to the end points of the previously established paths using the UTD routing algorithm with a small modification: the nodes that belong to any of the Hamiltonian paths set their subtree cost c_2 according to their position in the route, such that c_2 is inversely proportional to the number of hops towards d_{BS}. In this way, there is a big penalty to relay traffic on those nodes close to d_{BS}, belonging to any Hamiltonian path.

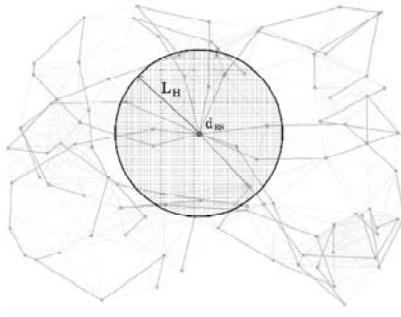


Figure 6.6: Routing algorithm for random networks with finite buffers: nodes inside the circle route packets using a Hamiltonian path while the outside nodes try to distribute load uniformly among the neighbors of the base station d_{BS}.

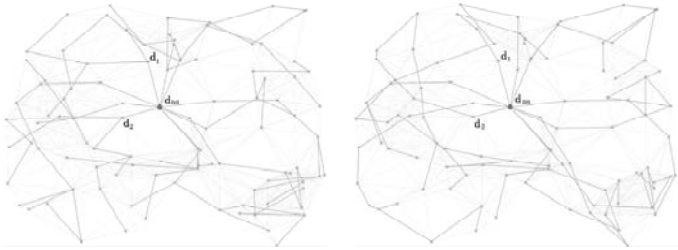
Using this penalty gradient, we also ensure that all nodes remain connected. That is, we allow nodes to connect to other nodes than the end-points if these are the minimum subtree weight nodes. We denote this routing algorithms as UTD-Q routing. Note that the UTD-Q routing algorithm can be executed in a totally distributed way and that it can also be applied in the case of multiple data collectors and for different base station locations.

To reduce overflow losses, it is convenient to choose a large value for L_Q, so that many nodes receive traffic from only one neighbor. However, if L_Q is too large, the traffic would be distributed unevenly among $\varphi^n(d_{BS})$, and overflow loses would consequently increase. This suggests that there exists an optimal L_Q value for which overflow losses are minimized. Note that the average path length increases linearly with L_Q, indicating that there exists a trade-off between overflow losses (capacity) and the average number of transmissions (delay).

For illustration purposes, we show the DGT generated by the UTD and UTD-Q algorithms with $L_Q = 4$ for a single realization of a 200 nodes random network in Figure 6.7. Note that UTD-Q avoids having nodes (such as d_1 and d_2) located close to the base station and receiving traffic from several neighbors. However, it is also important to note that UTD induces also in some cases Hamiltonian paths in those nodes close to the base station. The reason is that the number of nodes located at a distance of d hops from d_{BS} decreases linearly with d. Thus, in order to generate a uniform traffic distribution, it is necessary that nodes close to d_{BS} receive traffic from only a few of their neighbors.

6.5.3 Simulation results

In this section, we compare through simulations the performance of different algorithms for packet routing in random networks with finite buffers. To drive the simulations we use NAB (Network in A Box) [33], a network simulator described in the introduction chapter.



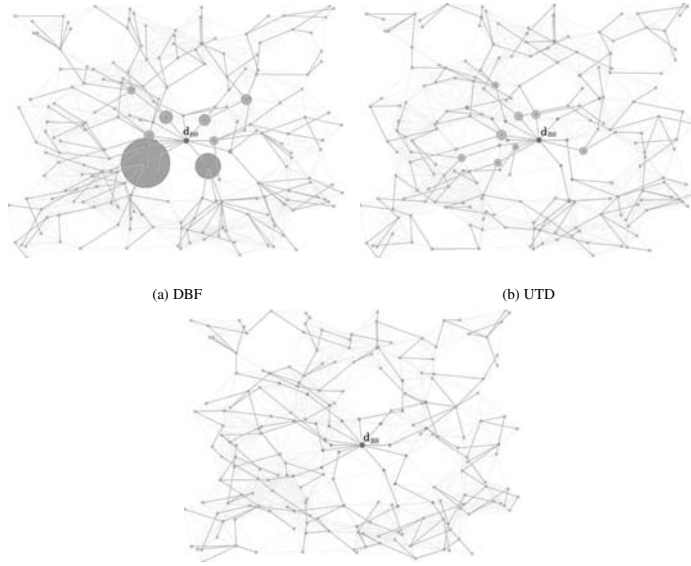
(a) (b)

Figure 6.7: Data gathering routing in a random network: DGT generated by (a) UTD and (b) UTD-Q routing algorithms.

We first analyze through simulation the overflow loss distribution depending on the routing algorithm used. Figure 6.8 shows the average overflow losses per node using the DBF, UTD and UTD-Q algorithms for a transmission rate $\mathcal{R} = 0.75C_w$, in a random network of $N = 200$ nodes with an average number of neighbors per node $n_{\text{avg}} = 15$ and a buffer size $Q = 3$. Overflow losses are represented by a circle centered in each node whose radius is proportional to the average losses in each node. As expected, DBF routing (Figure 6.8(a)) induces high losses in some of the neighbors of d_{BS}, mainly due to a highly unbalanced traffic distribution. These hot-spots are suppressed when UTD routing (Figure 6.8(b)) is used instead. However, nodes close to d_{BS} that receive traffic from multiple neighbors still present significant overflow losses. Note from Figure 6.8(b) that all the nodes that present significant losses receive traffic from several neighbors. The UTD-Q algorithm reduces considerably overflow losses in these nodes by making them receive all the traffic from only one neighbor.

We compare now the performance of different routing algorithms for packet transmissions over random networks. As performance metric, we compute the maximum achievable rate $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q)$ considering that a rate is feasible if the average loss probability in the network is smaller than 1%. Particularly, we compute $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q)$ as a function of the buffer size Q, the network size N, and the network connectivity n_{avg} for two shortest path routing algorithms, DBF and UTD-SP, and two non shortest-path algorithms, UTD and UTD-Q. The values given for UTD-G correspond to the maximum rate achieved for the optimal L_Q value.

Figure 6.9 compares the average performance of DBF, UTD, UTD-SP, and UTD-Q as a function of the buffer size Q in random networks of $N = 300$ nodes with average number of neighbors per node $n_{\text{avg}} = 15$. Figure 6.9(a) shows the gain in the maximum achievable rate gain over the DBF algorithm, that is, $\mathcal{R}^{\Pi}_{\text{sup}}(N, Q)/\mathcal{R}^{\text{DBF}}_{\text{sup}}(N, Q)$. As expected, UTD-Q achieves always the highest rate per node and, on average, can be up to 4.5 times higher than the rate achieved with DBF routing. This gain is clearly more noticeable for small buffer sizes, where overflow losses are more critical. Note also that the performance improvement of UTD-Q over UTD is only significant for small buffers. The reason is that, as we show in



(a) DBF

(b) UTD

(c) UTD-Q

Figure 6.8: Overflow losses per node in a 200 nodes network with $n_{\text{avg}} = 15$ and $Q = 3$ using the (a) DBF, (b) UTD, and (c) UTD-Q routing with $L_Q = 5$. Buffer overflow losses are represented by a circle centered in each node whose radius is proportional to the average losses in each node.

Figure 6.7, UTD tends to generate also Hamiltonian paths in those nodes close to the base station, so that the benefits of UTD-Q are diminished.

Figure 6.9(b) shows the average path length relative to the average shortest path length, i.e., the average path length if all the nodes use a shortest path to the base station. The reason why UTD-Q presents a higher path length for small buffers is explained in Figure 6.9(c). As the buffer size increases, both non-shortest path algorithms, UTD and UTD-Q, present an average path length of approximately 1.5 times the average shortest path length. This illustrates the trade-off between overflow losses and the average number of transmissions.

Figure 6.9(c) shows the average optimal L_Q length of the UTD-Q routing that achieves the maximum rate. As expected, the optimal L_Q length is higher for small buffers and decreases monotonically with the buffer size. For small buffer sizes, to reduce overflow losses, it is necessary to increase the length of the Hamiltonian paths, even if this implies a small load

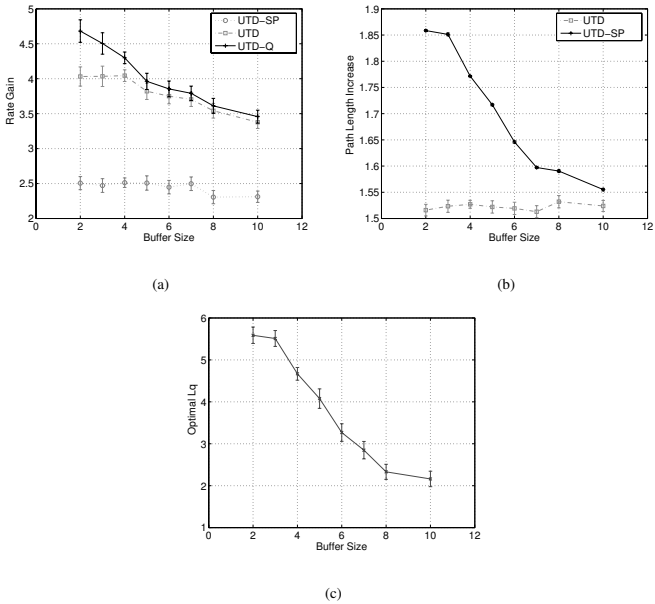


Figure 6.9: Average performance of DBF, UTD, UTD-SP, and UTD-Q as a function of the buffer size Q in random networks of $N = 300$ nodes with average number of neighbors per node $n_{\text{avg}} = 15$. (a) Average maximum achievable rate gain over the DBF routing, (b) average path length relative to the average shortest path length, and (c) average optimal L_Q length.

unbalance in $\varphi^n(d_{BS})$. Consequently, the average path length is also increased. For large buffer sizes, the uniform load distribution becomes more crucial in order to reduce losses and consequently, the length of the Hamiltonian paths should be reduced.

Figure 6.10 compares the average performance of the same four algorithms as a function of the average node connectivity n_{avg} in random networks of $N = 300$ nodes and a buffer size $Q = 3$. Note that, for all algorithms, the rate gain over DBF (Figure 6.10(a)) increases linearly with n_{avg}. The reason is that, as we explained in the infinite buffer case, the traffic distribution generated by DBF in $\varphi^n(d_{BS})$ becomes more unequally distributed as the average number of neighbors per node increases. Note also that the performance gain of UTD-Q over UTD increases slightly with n_{avg}. The reason is that in dense networks, nodes located

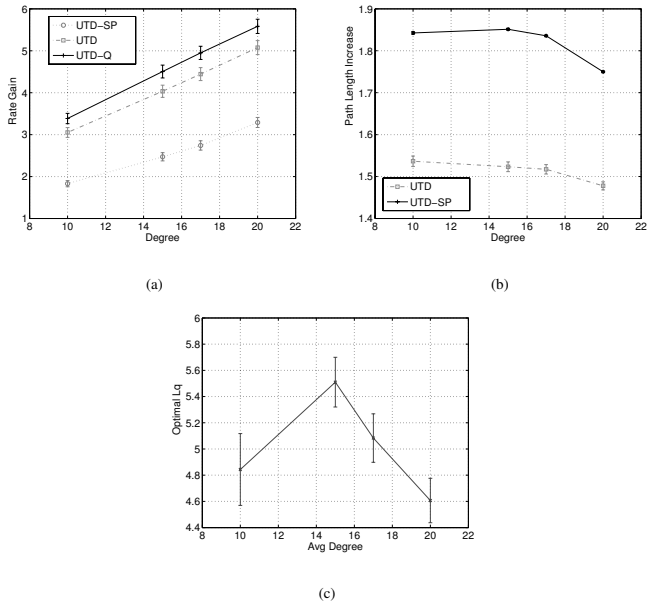


Figure 6.10: Average performance of DBF, UTD, UTD-SP, and UTD-Q as a function of the average node connectivity n_{avg} in random networks of $N = 300$ nodes and a buffer size $Q = 3$. (a) Average maximum achievable rate gain over DBF routing, (b) average path length relative to the average shortest path length, and (c) average optimal L_Q length.

close to the base station are in the communication range of an increasing number of nodes. Consequently, they can receive traffic from multiple nodes, increasing the overflow losses in non-Hamiltonian paths. From Figure 6.10(b) and Figure 6.10(c), we can conclude that the average path length and the optimal L_Q length do not change significantly with n_{avg}.

Figure 6.11 compares the average performance of the same four algorithms as a function of the network size N in random networks with an average number of neighbors per node $n_{\text{avg}} = 15$ and a buffer size $Q = 3$. The performance gain of UTD-Q and UTD over DBF (Figure 6.11(a)) slightly increases with the network size and remains almost steady for networks bigger than 300 nodes. Note also that the performance improvement of UTD-Q over UTD is more significant for bigger networks. On the other hand, the average path length (Fig-

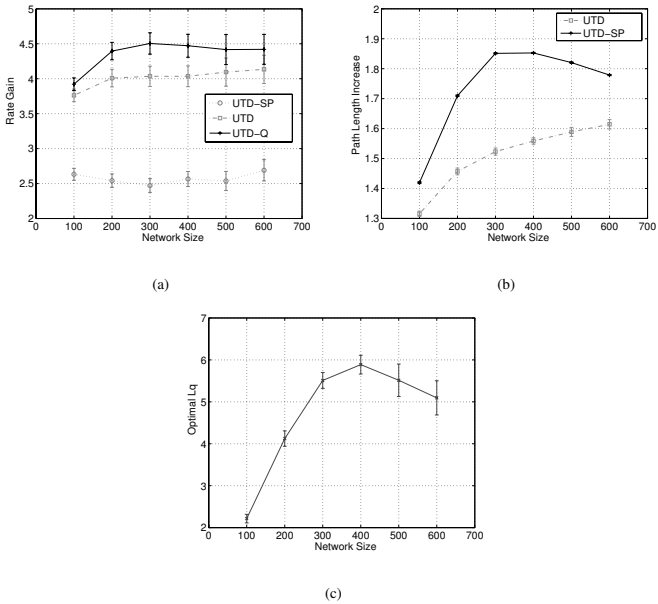


Figure 6.11: Average performance of DBF, UTD, UTD-SP, and UTD-Q as a function of the network size N in random networks with an average number of neighbors per node $n_{\text{avg}} = 15$ and a buffer size $Q = 3$. (a) Average maximum achievable rate gain over DBF routing, (b) average path length relative to the average shortest path length, and (c) average optimal L_Q length.

ure 6.11(b)) and the average optimal L_Q (Figure 6.11(c)) increases significantly with N. The reason is that, for a constant average number of neighbors per node, bigger networks allow for longer Hamiltonian paths while achieving a uniform traffic distribution in $\varphi^n(d_{BS})$.

In view of these results, we can conclude that the main reason for the better performance of the UTD routing algorithms family over the DBF algorithm is the more uniform distribution of the load in $\varphi^n(d_{BS})$. However, the performance improvement of UTD-Q over UTD is significant for small buffers and big networks. It is also important to note that, even if the average overflow losses using the UTD and UTD-Q are similar for many network conditions, the losses distribution differ considerably: while UTD concentrates the losses in very few nodes close to d_{BS}, UTD-Q distributes these losses over a higher number of nodes. This can

also be observed in Figure 6.8.

6.6 Wireless random networks

In previous sections, we assumed that either the network is wired, or that there exists a transmission schedule or frequency division multiplexing that avoids conflicts. Under this assumption, nodes can simultaneously receive multiple packets and, at the same time, transmit one packet to the next hop. This allowed us to model random networks as random graphs with point-to-point connectivity. We turn our attention now to the problem of routing in wireless networks.

We analyze first network capacity and show that the optimal transmission strategy requires generating a transmission schedule which is also an NP-complete problem. Then, we give the wireless interpretation of the optimal routing strategy we derived for wired random networks. Based on this interpretation, we propose a simple alternative method to the time scheduling that consists in modifying the media access control to deal with finite buffer and reduce packet overflow.

6.6.1 Network model

The network model we consider is similar to the one of the previous section but with wireless links. We assume that nodes use a fixed transmission power and achieve a fixed transmission range R_{tx}. We adopt the following commonly used interference model [29]: let d_i and d_j be two sources with distance $s(d_i, d_j)$ between them. Then, the transmission from d_i to d_j will be successful if and only if:

$$s(d_i, d_j) \leq R_{tx} \;\; and \;\; s(d_k, d_j) > R_{tx}, \tag{6.9}$$

for any source d_k that is simultaneously transmitting. This interference model implies that nodes can neither receive more than one transmission at a time nor transmit and receive at the same time.

To avoid packet collisions, we assume a listening mechanism such as CSMA: nodes listen to the channel before initiating a packet transmission, and transmit only if the channel is idle. To resolve the problem of packet collisions due to the well known hidden terminal problem, we assume an ideal RTS/CTS contention control scheme: when a node wants to transmit data to another node, it sends out a Request To Send (RTS) packet. The receiver node replies with a Cleared To Send (CTS) packet. After the transmitter node receives the CTS packet, it transmits the data packet. Any other node receiving the CTS packet should refrain from sending data for a given random time (solving the hidden node problem). To simplify the analysis, we assume that this RTS-CTS mechanism is error-free and neither packet losses nor collisions occur during this mechanism.

When packet transmission from a node is not possible (either the channel is busy or the transmission will result in a collision with an already transmitting node), the packet is stored in the buffer and re-scheduled after some random time t_{BO}. This backoff mechanism is widely used in media access control to reduce contention [75]. The idea of the backoff mechanism is to restrain a node from accessing the channel for a period of time and hopefully, the channel will become free after the backoff period. To avoid synchronization among periodic streams of traffic, t_{BS} is a uniformly distributed random variable that takes values in a backoff window $[0, T_{BO}]$.

As we see in the next section, the achieved rate per node in the wireless scenario highly depends on the media access control and transmission strategies [65]. We start by reviewing some results on network capacity and transmission strategies.

6.6.2 Network capacity

In the wireless scenario, d_{BS} can only receive packets from one of its neighbors simultaneously. Therefore, an obvious upper bound on the network capacity is given by the single shared channel between d_{BS} and its neighbors, which carries all the traffic generated by the entire network:

$$C_c(N) \leq \frac{C_l}{N-1}. \tag{6.10}$$

This upper bound can only be achieved if the base-station is receiving all the time. Unfortunately, it can be shown that this upper bound is in general not achievable with high probability as the number of sensors increases to infinity [17]. Duarte-Melo and Liu [17] gave a constructive lower bound that can be achieved with high probability in a randomly deployed network: they showed that the maximum achievable rate per node is arbitrarily close to $\frac{C_l}{(N-1)(2-\pi R_{tx}^2)}$.

However, this lower bound assumes that nodes are synchronized and that there exists a schedule that determines what subset of nodes can transmit simultaneously during which time slot. This schedule which is left unspecified in [17], is highly non trivial to generate, especially in a distributed context. Note also that both upper and lower bounds are $\mathcal{O}(1/N)$, indicating that network capacity decreases with the total number of nodes N.

6.6.3 Optimal transmission strategies

The optimal transmission strategy requires generating an optimal schedule that determines the subset of nodes that can transmit simultaneously. Unfortunately, the problem of deriving optimal channel access schedules for multihop networks is an NP-Complete problem [4; 19]:

Proposition 6.1 *The algorithm to find the optimal transmission schedule that maximizes the achievable rate per node in a random wireless network with is NP-complete.*

Proof: We represent the wireless random network with a connectivity graph $G_c = (V, E)$, where the edges represent that the distance between two nodes is less than the transmission range. That is, transmissions from one node are heard by all its neighbor nodes. If we assume that time is slotted and nodes are synchronized, the schedule can be seen as a function $f : V \rightarrow \mathbb{N}$ that determines in which time slot each node transmit. To derive the optimal scheduling function f is equivalent to solving the graph-coloring problem in a random graph.

The graph k-colorability problem [23] consist in, given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, finding a function $f : V \rightarrow \{1, 2, \ldots, k\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$. The minimum chromatic number of the graph is the minimal k that allows to solve the k-colorability problem.

Starting from the original network connectivity graph G_c, we construct the interference graph $G_i = (V, E_i)$ with edges connecting nodes that are withing each other's interference range. Computing the optimal schedule is equivalent to obtaining the chromatic number of G_i and solve the k-colorability problem, which is an NP-complete problem [23]. □

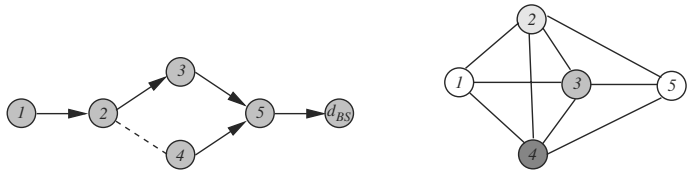


Figure 6.12: To obtain the optimal schedule requires solving the graph coloring problem in the interference graph. a) Original wireless network, where edges represent that the distance between two nodes in less than the transmission range and arrows indicate the routing algorithm. b) Equivalent interference graph, with edges connecting nodes that are withing each other's interference range. For instance, since nodes 1 and 3 cannot transmit simultaneously (collision at node 2), there exists an edge between both nodes.

We illustrate this optimal schedule with an example. Figure 6.12(a) shows the connectivity graph of a wireless network where arrows represent the routing algorithm and Figure 6.12(b) shows its correspondent interference graph. For this simple graph, we can easily obtain the chromatic number ($k = 4$) and solve the 4-colorability problem. Therefore, as shown in Figure 6.12(b), the optimal schedule consists in cycles of 4 time slots: in the first time slot nodes 1 and 5 transmit simultaneously, and the nodes 2, 3, and 4 transmit in the second, third and fourth time slots respectively. Given that all nodes transmit once during each period of 4 time slots, we need 5 of these periods to transmit one packet from each node to the base station. Therefore, the maximum generation rate per source is limited to one packet each 20 time slots. However, note that this time scheduling is clearly not efficient.

To achieve the same amount of data delivered to the base-station for each node, a non-uniform allocation of the bandwidth is necessary. While solving the graph coloring problem results in a time schedule that assigns the same bandwidth to all the sources, the optimal strategy requires assigning a higher bandwidth to the nodes that relay traffic for other nodes. To consider the relayed traffic, we use *virtual sources* [17]: for each node, we create one virtual source for every source whose traffic goes through this node. We reconstruct now the interference graph adding these virtual sources to the connectivity graph and solve the graph coloring problem to obtain the optimal traffic schedule.

Figure 6.13 shows the modified interference graph of the previous example with the virtual sources. It is easy to verify that the chromaticity of this new graph is 11. Therefore, there exists an optimal schedule of length 11 in which all nodes transmit one packet to the base station. Consequently, the maximum rate that nodes can generate is limited to one packet every 11 times slots, which is higher than the rate we obtained previously.

However, the k-colorability problem is NP-complete for all $k \geq 3$ [23]. This problem is indeed among the first ones proved to be intractable, and hence, it is very unlikely that an optimal polynomial-time algorithm could ever be devised for it.

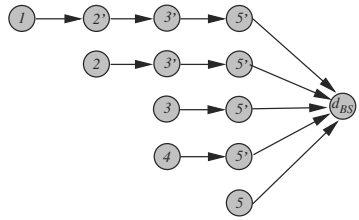


Figure 6.13: To achieve a fair bandwidth load, we need to consider also the traffic nodes relay. We generate a *virtual source* (marked with a prime) for each of the sources relayed by a node and solve the graph coloring problem in this new graph.

6.6.4 Approximative solutions for finite buffers

Applying graph theory results, the chromaticity of a random graph is upper bounded by the highest degree n_{max} plus one [12]. Therefore, there exists a schedule of length at most n_{max} that allows all nodes to transmit at least once. Some polynomial time approximation algorithms to compute schedules can be found in [56]. However, even to generate suboptimal schedules is non trivial.

An alternative solution with good performance can be found in the Media Access Control (MAC): some MAC schemes are designed to achieve fair bandwidth delivery to the base station [75] or to have a fair sharing of the channel among local competing neighbors [51]. Among these, the backoff is a widely used MAC mechanism to reduce contention. The idea is to restrain a node from accessing the channel for a period of time and hopefully, the channel will become free after the backoff period. Generally, the backoff time t_{BO} is randomly drawn from a fixed backoff window $[0, T_{BO}]$. In the following, we propose a simple backoff mechanism to reduce overflow losses in wireless networks.

To motivate the problem, we show first in Figure 6.14 the overflow loss distribution in a 500 nodes random wireless network with $n_{\text{avg}} = 15$ and $Q = 3$ using a shortest path routing algorithm and backoff window size equal for all nodes. Overflow losses are represented by a circle centered in each node whose radius is proportional to the average losses in each node. Note that this overflow loss distribution is similar to the distribution we obtained in the wired case: a) overflow losses occur in nodes close to the base station and hence carrying more traffic, and b) nodes that present high overflow losses receive traffic from multiple neighbors (Figure 6.15).

The reason why nodes receiving traffic from multiple neighbors present high overflow losses can be illustrated through an example. Let us consider a simple four nodes network as the one depicted in Figure 6.16(a), where all nodes generate packets locally at the same rate and node 4 relays traffic for nodes 1, 2, and 3. We consider the network in a high rate regime and assume that nodes have always at least one packet in the queue waiting to be transmitted. If we assume that all nodes have the same backoff window, the probability of capturing the channel for a packet transmission is identical for all four nodes, and equivalently, the average packet transmission time is also identical. However, the average arrival rate to node 4 is four times higher than to nodes 1, 2, and 3. Analyzing the distribution on the number of

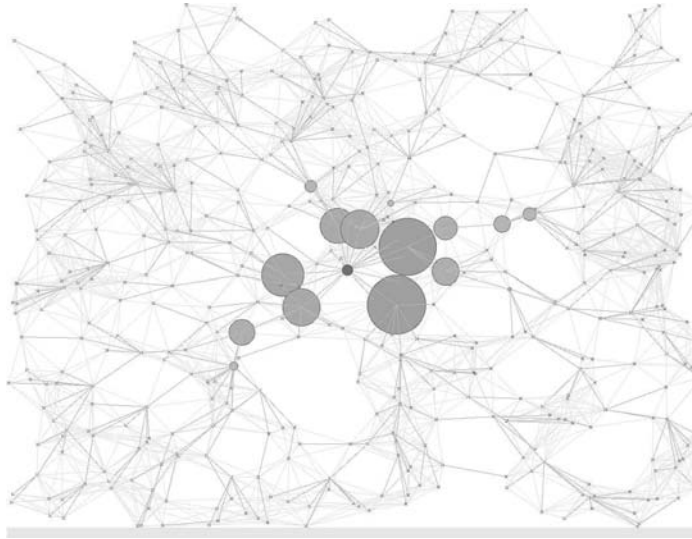


Figure 6.14: Overflow losses per node in a 500 nodes network with $n_{\text{avg}} = 15$ and $Q = 3$ using a shortest path routing algorithm and fixed backoff window identical in all nodes. Buffer overflow losses are represented by a circle centered in each node whose radius is proportional to the average losses in each node.

packets waiting in the queue of each node, it is clear that the number of packets in node 4 is significantly higher than in any other node, which in the finite buffer case, will translate into higher overflow losses. Thus even if the backoff mechanism regulates the transmission rate of the nodes by preventing transmissions after packet collisions, it induces unbalanced queue distributions.

To reduce overflow losses, we need a transmission strategy that minimizes the number of packets waiting in the queue of the most loaded nodes. For instance, in the previous example the optimal transmission schedule that minimizes the number of packets in the queues is depicted in Figure 6.16(b): when node 4 has one packet waiting to be transmitted, it has priority over its neighbors to capture the channel. In this case, the maximum buffer size required at any node is just of one packet.

Note that this optimal schedule is very similar to the routing strategy that minimizes overflow losses in the wired case, which consisted in making nodes receive traffic exclusively from one of their neighbors (Chapter 4). This routing strategy guaranteed that packets received in high loaded nodes can be immediately transmitted, so that the queue remains constrained. The wireless interpretation of this optimality condition is that when a node receives

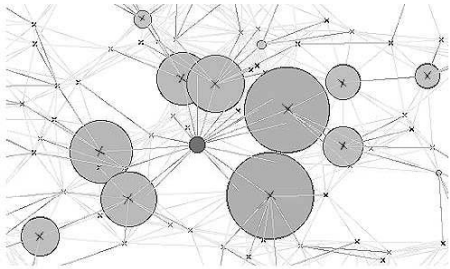


Figure 6.15: Detail of the overflow loss distribution in the center of the network: nodes that present high overflow losses receive traffic from multiple neighbors.

one packet from any of its neighbors, it must have the opportunity to transmit it without receiving a new packet from its neighbors. This condition is equivalent to enforcing a higher probability to capture the channel in nodes that carry more traffic.

A simple approach to induce this asymmetry in the channel access probability is to use the backoff mechanism. In the previous example, we considered that all nodes have the same backoff window size T_{BO} and consequently, identical average packet transmission times. Indeed, the average packet transmission time of a node d_i is inversely proportional to the size of its backoff window $T_{BO}(d_i)$. Thus, to minimize the number of packets waiting in the queue of the most loaded nodes, we require a different backoff window in each node which is inversely proportional to the total traffic arrival rate, i.e., inversely proportional to the traffic nodes relay. Under the central data gathering model, we can approximate the relayed traffic with the distance to the base station: the traffic arrival rate to a node is inversely proportional to its distance to the base station. Using this approximation, we propose a simple MAC mechanism where each node adapts the size of its backoff window according to its distance to the base station such that if $h(d_i, d_{BS}) > h(d_j, d_{BS})$ then $T_{BO}(d_i) > T_{BO}(d_j)$.

As the network size increases, the traffic generated locally in high loaded nodes becomes negligible with respect to the relayed traffic. Therefore, the total arrival traffic to a node d_i is in average $\mathcal{O}(n_{\text{avg}})$ times higher than the traffic that goes through its neighbors for which d_i relays traffic. Consequently, we require d_i to have an average packet transmission time $\mathcal{O}(n_{\text{avg}})$ times smaller than these neighbors, or equivalently, a backoff window $\mathcal{O}(n_{\text{avg}})$ times smaller. Note also that the neighbors of d_i that relay traffic through d_i are located one hop further from the base station that d_i. Consequently, we require a backoff window that increases linearly with the distance to the base station and with slope equal to the average neighbors per nodes, that is:

$$T_{BO}(d_j) = T^0_{BO} n_{\text{avg}} h(d_j, d_{BS}). \tag{6.11}$$

In the next section, we evaluate through simulations the performance of this simple MAC mechanism for wireless networks with small buffers at the nodes.

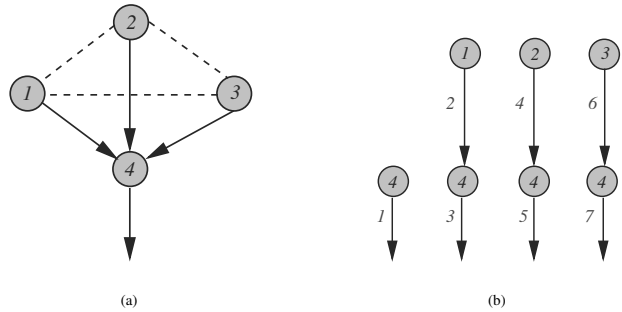


Figure 6.16: Wireless network example: a) if all nodes have the same probability to capture the channel for a transmission, the average number of packets in the queue of node 4 is significantly higher than in any other node, b) optimal transmission strategy with a maximum buffer size of one packet.

Note that the backoff mechanism is a self-regulated mechanism: it is only applied when collisions appear in the network, that is, when the network becomes congested. This control mechanism can be combined with other control mechanisms, such as the linear increase and multiplicative decrease approach [75].

Note also that the gain of the backoff approach we proposed to reduce overflow losses depends on the MAC protocol used. Certain MAC schemes can already incorporate congestion control mechanisms, that reduce the benefits of this simple algorithm.

6.6.5 Simulation results

In this section, we compare through simulations the performance of the linear increasing backoff window mechanism with a constant backoff window where all the nodes have identically backoff window sizes. To drive the simulations we use NAB (Network in A Box) [33], a network simulator described in the introduction chapter.

Figure 6.17 compares the average performance of both MAC mechanisms for different buffer sizes Q as a function of the packet generation rate $\mathcal{R}$ in random networks of $N = 100$ nodes with average number of neighbors per node $n_{\text{avg}} = 15$. Particularly, we show the throughput gain at the base station of the linear increasing backoff window mechanism over the constant backoff window mechanism. As expected, the throughput gain is more significant when buffers at the nodes are small. Note also that the gain increases linearly with the rate for all buffer sizes. Thus, the throughput gain is more noticeable in networks with small buffers and transmitting in a high load regime, that is, when overflow losses are more likely to happen.

Figure 6.18 compares the average performance of the same two MAC mechanisms for different network sizes N as a function of packet generation rate $\mathcal{R}$ in random networks with

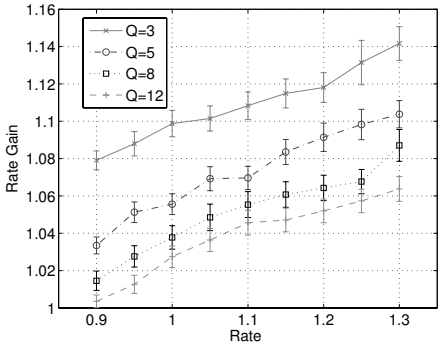


Figure 6.17: Average throughput increase of the MAC mechanism using a linear increasing backoff window with the distance to the base station with respect to a constant backoff window for different values of the buffer size Q in random networks of $N = 100$ nodes with average number of neighbors per node $n_{\text{avg}} = 15$.

buffer size $Q = 5$ and average number of neighbors per node $n_{\text{avg}} = 15$. Note that the throughput gain increases with the network size, especially at lower rates. As above, the gain increases linearly with the rate for all network sizes.

In general, the throughput gain we obtain by using the linear backoff mechanism in the wireless case are qualitatively similar to the gain reported by using the UTD-Q algorithm in the wired case. That is, this gain increases significantly with average degree n_{avg}, decreases with the buffer size Q and slightly increases with the number of nodes N. This indicates that, even if wireless networks are very different in nature from wired networks, the overflow loss mechanism behaves quite similarly in both.

6.7 Summary

In this chapter, we studied the problem of routing in random networks. We first analyzed a collision-free or wired random network model and considered both infinite and finite buffers at the nodes. In the infinite buffer case, we studied the network capacity and showed that the problem of finding the optimal routing algorithm that achieves this capacity is an NP-hard problem. Then, we proposed a distributed approximation algorithm that behaves close to optimal. Using similar techniques as in Chapter 3, we analyzed the effect of finite buffers on the routing strategy and showed that finding the optimal routing algorithm is also an NP-hard problem. We proposed an approximation algorithm to minimize overflow losses that achieves an average rate per node considerably higher than the rate achieved with the usual shortest path routing algorithm.

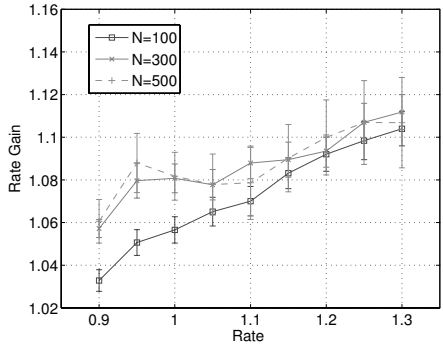


Figure 6.18: Average throughput increase of the MAC mechanism using a linear increasing backoff window with the distance to the base station with respect to a constant backoff window for different network sizes N with $Q = 5$ and average number of neighbors per node $n_{\text{avg}} = 15$.

Finally, we discussed the wireless case, where packet transmissions among different nodes can interfere and result in packet collisions. We showed that the overflow loss process is quite similar in both collision-free and wireless models. We proposed a simple method to reduce overflow losses that consists in adapting the backoff window size linearly with the distance to the base station.

Chapter 7

Joint Source Coding and Routing

7.1 Introduction

Multipath routing was shown to be a good strategy in unreliable networks: by exploiting the space diversity of the network and making the data flow along multiple routes, we were able to overcome node failures and reduce the unpredictability of random networks. In this regard, we described a routing algorithm that uses all the available paths between the source and the destination such that the load is uniformly distributed. In the context of large and dense unreliable networks where the number of available paths between any two nodes is large, this routing algorithm outperforms other routing schemes that only make use of one or few paths (Chapter 5). In this chapter, we show that we can exploit this multipath property even further by using an appropriate source coding mechanism.

Until now we have characterized any routing algorithm by the maximum rate that nodes can *reliably* transmit: That is, we considered only the network layer. However, there is much to be gained if the upper layers are also considered in the design of routing algorithms. Particularly, we investigate the interaction of the source coding and the routing mechanisms in sensor networks, and consider a new performance metric of routing algorithms, namely the distortion achieved at the destination. The idea is to combine the space diversity provided by the routing algorithm and the network with an appropriate source coding technique such as Multiple Description (MD) coding.

Packet transmissions over sensor networks are subject to many failures such as node breakdowns, link failures, overflow losses, packet corruption, etc. In scenarios where retransmissions are not possible due to time constraints or expensive feedback, source coding techniques that make all the received packets useful can be of great benefit. If, in addition to this, the network provides the necessary spatial diversity to transmit packets along different paths, multipath routing reduces the loss probability correlation among packets. Multiple Description (MD) codes are specially conceived for this scenario: as opposed to Single Description (SD) coding, an MD source encoder partitions information into *descriptions* that can be sent over the available paths to the destination. Depending on the subset of packets that is correctly received, the destination computes an estimate of the original source. The quality of the estimate depends on the number of descriptions received but, in contrast to the single-description case, the loss of packets does not lead to a catastrophic failure only to degradation. This is illustrated in Figure 7.1.

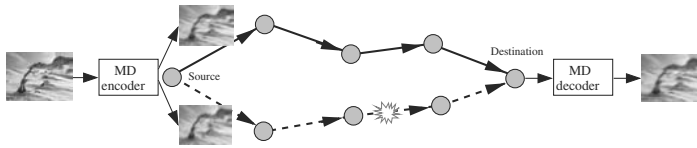


Figure 7.1: MD coding and multipath routing: source information is encoded in two descriptions (packets), which are complementary and at the same time independently good. Each description is sent along a different path. Even if we lose one of these descriptions, the destination is able to compute an estimate of the original signal.

The signal distortion at the destination depends on the number of packets received and on the coding method used to generate the descriptions. Therefore, if we want to maximize the quality of the reconstructed data, it is necessary to optimize jointly two elements: the information rate at the destination and the reconstruction of the original data from the received packets.

An interesting scenario where MD coding can be beneficial is in real time signal transmission, where packet retransmissions are not possible due to the delay constraint. In this chapter, we study the joint optimization of the source coding and the routing mechanisms under this scenario: we formulate and solve analytically a network problem that merges both coding and routing, in the context of real time data transmissions. We show that the most sophisticated scheme (using MD coding and multipath routing) performs significantly better and is more robust over a wide range of network parameters than the usual scheme, that is, SD coding and single path routing.

One of the main limitations of sensor networks is the reduced memory available at the nodes. This memory constraint translates into a limited (and generally small) space for the temporary storage of packets, which causes packet loss due to buffer overflow. This is also a scenario where MD coding can be used to reduce the consequences of overflow losses. We also consider the practical aspects of using MD coding in large sensor networks where the set of available paths between any source-destination pair is large. While most of the previous work studies MD coding from a rate-distortion point of view, we concentrate on the practical aspects of implementing an MD coding scheme in a sensor network.

The results obtained in both scenarios, real time data and finite buffer networks, indicate the benefit of combining MD coding techniques and multi path routing in large and dense unreliable networks.

The rest of the chapter is organized as follows. We start in Section 7.2 by a brief review of some of the existing results and techniques of MD coding. In Section 7.3, we introduce a simple network model where we study the optimal coding and routing strategies in terms of achieved distortion. In Section 7.4.1, we analyze some MD coding techniques capable of generating an arbitrary number of descriptions and in Section 7.4.2, we apply these MD coding techniques in a data gathering sensor network.

7.2 An overview of multiple description coding

We start by a brief review of some of the existing results and techniques of MD coding. The performance of a source coder is characterized by its rate-distortion curve. In the case of MD coding, closed formulas for the rate-distortion curves only exists for the case of Gaussian sources and the two descriptions case. We consider, therefore, an independent and identical distributed Gaussian source with zero mean and unit variance $\sigma^2 = 1$. The information generated by the source is encoded by an MD encoder that generates 2 descriptions using B bits in total. We assume that both descriptions are equally important and the encoder assigns $B/2$ bits per description. The achievable rate-distortion in this case is defined by the following equations [55]:

$$\begin{aligned} D_s^1 = D_s^2 &= 2^{-2\frac{B}{2}(1-\eta)}, \\ D_c &= 2^{-2B}\frac{1}{1-(1-2D_s)^2}, \end{aligned} \tag{7.1}$$

where $D_s^i,\ i \in \{1,2\}$, are the side distortions, that is, the distortion in the case that only the i-th description gets to the destination, and D_c is the central distortion, or the distortion achieved when both descriptions are received. The parameter η, $(0 \leq \eta \leq 1)$, represents the trade-off between the side and the central distortion, or equivalently, the amount of redundancy that the coder adds to the generated descriptions.

A possible practical implementation of an MD coder is Multiple Description Scalar Quantization (MDSQ), proposed by Vaishampayan [72]. An MD scalar quantizer can be seen as an ordinary quantizer plus an index assignment that generates two indices per quantized sample. The index assignment consist in searching the sample value in a matrix and returning the row and column indices of the sample. The trade off between central and side distortion is represented by the number of diagonals we fill into the matrix. Figure 7.2 shows some possible realizations of this matrix. For instance, if we fill only one diagonal of the matrix we have a pure repetition code, achieving the best side distortion. On the other hand, if we fill all possible diagonals we achieve the best central distortion possible. This scheme has been proved to be asymptotically optimal [72].

An alternative way to implement MD coding is Unequal Error Protection (UEP) codes with a progressive source coder [25]. The idea is to use a progressive source coder to produce a representation at rate $(2-\zeta)(B/2)$, $\zeta \in [0,1]$, and then partition this representation in three parts. The initial (most important) $\zeta(B/2)$ bits are repeated in each description; the second $(1-\zeta)(B/2)$ bits are put in description 1; and the final $(1-\zeta)(B/2)$ bits are put in description 2. This is summarized in the following scheme, where the square represents repeated bits:

Description 1:	ζ (B/2)	(1−ζ)(B/2)
Description 2:	ζ (B/2)	(1−ζ)(B/2)

In the particular case of a memoryless Gaussian source with squared error distortion and two descriptions, the achievable rate-distortion region is given by the following formulas:

$$\begin{aligned} D_c &= 2^{-2(2-\zeta)\frac{B}{2}}, \\ D_s^1 &= 2^{-2\frac{B}{2}}, \\ D_s^2 &= 2^{-2\frac{B}{2}\zeta}. \end{aligned} \tag{7.2}$$

The parameter ζ, $(0 \leq \zeta \leq 1)$, represents the trade-off between side and central distortions,

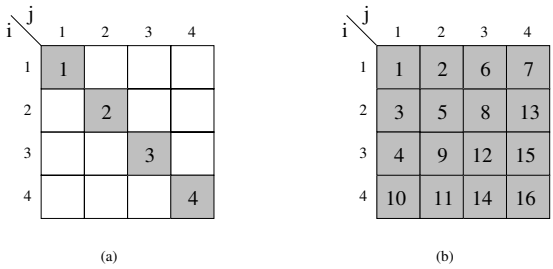


Figure 7.2: Index assignment method in Multiple Description Scalar Quantization (MDSQ): for each sample we generate two descriptions that correspond with the row and column index in the matrix. (a) If we only fill one diagonal of the matrix, it results in a simple repetition code that minimizes the side distortion. (b) All possible diagonals of the matrix have been filled, which corresponds to minimizing the central distortion.

or equivalently, the amount of redundancy that the coder adds to the generated descriptions. Goyal [25] proposed a very simple method to practically implement this coding strategy using a uniform scalar quantizer followed by a double description generator.

For illustration purposes, we show in Figure 7.3 the side and central distortion trade offs for both the MDSQ and UEP coding techniques. We compare also the theoretical distortion with the distortion achieved experimentally using a uniform scalar quantizer followed by a double description generation with $B/2 = 8$ bits per description. Note that MDSQ performs slightly better than UEP for almost all central-side distortion pairs. The difference between theoretical and experimental values is almost completely explained by two factors: first the performance gap due to the non optimal quantizer, and secondly, the fact of considering only one dimension in the quantizer.

7.3 Real time services over sensor networks: a toy example

An interesting scenario where MD coding can be of great value is in real time data transmission where packet retransmissions are not possible due to a delay constraint. Until now we have not considered the delay incurred by packets to travel from the source to the destination: all packets that reached the destination were considered as correctly received. However, in certain applications involving real time data, packets need to reach the destination within a certain delay, above which, the information they carry becomes obsolete.

The joint performance of a coding scheme and a routing algorithm can be characterized by the achieved distortion at the destination. To translate the rate that a routing algorithm provides into a distortion at the destination, we need to consider the source coding mechanism, characterized by its rate-distortion curve. However, as we have seen in Section 7.2,

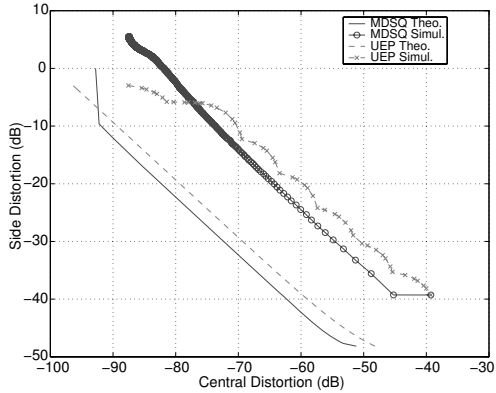


Figure 7.3: Central and side distortion trade off for MDSQ and UEP coding techniques with 8 bits per description. Solid lines represent MDSQ and dashed lines UEP. The side and central distortions are expressed in dB, i.e., $10\log(D_s)$.

closed formulas for the rate-distortion curve exists only for the case of Gaussian sources and two descriptions. Under a simple scenario, we derive in this section analytical expressions for the distortion at the destination and compare quantitatively different coding and routing techniques. Namely, we compute the distortion improvement achieved when two descriptions and multipath routing are used instead of the usual single path and single description coding.

7.3.1 Network model and assumptions

To obtain analytical expressions for the distortion, we simplify the network model as much as possible while still providing the main characteristics we want to explore: multiple source-destinations and space diversity. We consider a simple four nodes network (Figure 7.4): two devices act as sources, d_{s1} and d_{s2}, sending data to d_{d1} and d_{d2} respectively. Any device may serve as relay for ongoing communications.

We assume that sources generate real time data modeled as a zero mean and unit variance Gaussian random variable that needs to reach the destination within a maximum tolerable delay of Δ. Packets that have gone through a delay exceeding Δ are dropped at the destination. We assume that intermediate nodes do not have the capability of dropping packets that have already reached the maximum delay.

We model a sensor network that regularly measures a certain phenomenon without node synchronization. In this regard, we assume that sources generate one packet of constant size B bits each interval of T time units. The exact packet generation time inside the interval is given by a random variable uniformly distributed from 0 to T, independent among nodes.

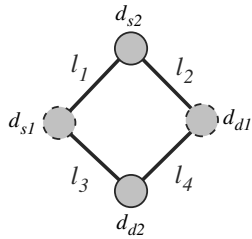


Figure 7.4: Network model: d_{s1} and d_{s2} transmit real time data to d_{d1} and d_{d2} respectively.

The arrival process A_i of such a source can be characterized by the following formula:

$$A_i = iT + a(i), \quad i = [0, 1, ...], \tag{7.3}$$

where a is a random variable uniformly distributed in the interval $[0, T]$.

Links represent simplex communication channels of capacity C_l. Note that for both source-destination pairs there exist two possible paths, each composed of two links. Both packet flows share the same network and hence compete for a certain capacity. We model each link l_i as a first come first served (FIFO) single-server queue q_i with a deterministic service time $1/\mu = B/C_l$, which is the time a packet takes to be transmitted from one node to the following.

We denote by $\tau(d_i, d_j)$ the random variable that indicates the total delay that a packet experiences to travel from d_i to d_j. Correspondingly, $f_{\tau(d_i,d_j)}$ represents its probability density distribution and $F_{\tau(d_i,d_j)}$ its cumulative distribution.

This simple scenario allow us to derive analytical expressions for the distortion achieved at the destinations d_{d1} and d_{d2} under different transport and coding mechanisms. First, we study the reduction in distortion achieved by using a multipath routing algorithm that distributes load between the two available paths instead of a single path routing. Secondly, we analyze the benefit of using an appropriate source coding mechanism combined with the multipath routing scheme.

7.3.2 Single description coding and single path routing

We start by computing the distortion for the usual single description coding and single path routing. Figure 7.5 illustrates the routing algorithm and the equivalent flow model. Since this flow model is equivalent for both source-destination pairs, we concentrate only on the pair (d_{s1}, d_{d1}). Assuming that a packet is dropped when the packet delay has exceeded a fixed value Δ, the distortion D at the destination in the single description case is given by:

$$D = 2^{(-2B)} F_{\tau(d_{s1},d_{d1})}(\Delta) + (1 - F_{\tau(d_{s1},d_{d1})}(\Delta)). \tag{7.4}$$

To obtain the delay cumulative distribution $F_{\tau(d_{s1},d_{d1})}$ incurred by packets to reach the destination, we model each link l_i as a $G/D/1$ queue q_i with a deterministic service time $1/\mu = B/C_l$ and analyze the resulting queueing network problem shown in Figure 7.5. The

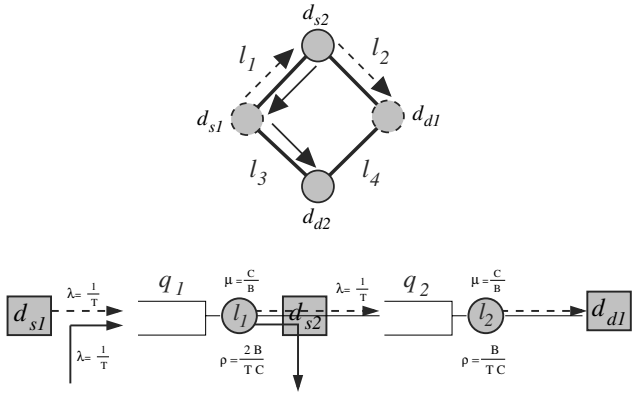


Figure 7.5: Top: single description and single path routing coding flow model. Bottom: queueing network model, where links are modeled by $G/D/1$ queues with a deterministic service time $\frac{1}{\mu} = \frac{B}{C_l}$.

analysis of this queueing network can be found in Appendix 7.A.1. After some computations, we obtain the following approximation for the cumulative distribution:

$$F_{\tau(d_{s1},d_{d1})}(\Delta) \approx \begin{cases} 0, & \text{if } \Delta \leq \frac{2B}{C_l}, \\ \beta + \frac{C_l}{B}(1-\beta)(\Delta - \frac{2B}{C_l}), & \text{if } \frac{2B}{C_l} \leq \Delta < \frac{3B}{C_l}, \\ 1, & \text{if } \frac{3B}{C_l} \leq \Delta, \end{cases} \tag{7.5}$$

where $\beta = \sqrt{(1-\rho_1)}$ and ρ_1 is the queue utilization factor of q_1. Combining now (7.4) and (7.5) we obtain the distortion at d_{d1}.

7.3.3 Single description coding and multipath routing

We now turn our attention to the case of SD coding and multipath routing. The optimal multipath routing strategy for both source-destination pairs simply consists in using the two available paths with equal probability. Figure 7.6 illustrates the routing algorithm and the equivalent flow model. Since this flow model is equivalent for both source-destination pairs, we concentrate only on the pair (d_{s1}, d_{d1}).

Given that we use again a SD coding scheme, we compute the distortion at d_{d1} using (7.4). To obtain the delay cumulative distribution $F_{\tau(d_{s1},d_{d1})}(\Delta)$ incurred by packets to reach the destination, we proceed as in the previous section: we model each link l_i as a $G/D/1$ queue q_i with a deterministic service time $1/\mu = B/C_l$ and analyze the resulting queueing network problem shown in Figure 7.6. The analysis of this queueing network can be found in

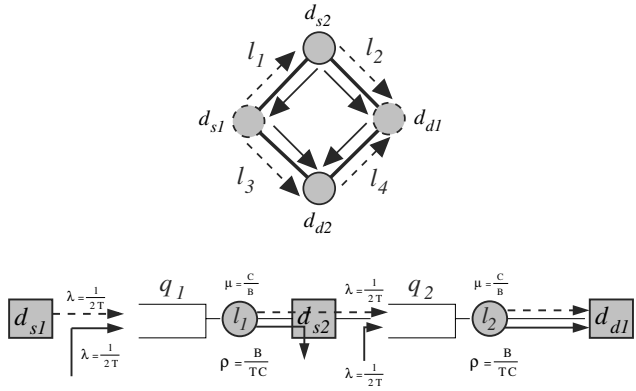


Figure 7.6: Top: SD coding and multipath routing flow model. Bottom: queueing network model, where links are modeled by $G/D/1$ queues of service time $\frac{1}{\mu} = \frac{B}{C_l}$.

Appendix 7.A.2. After some computations, we obtain the following approximation for the cumulative distribution:

$$F_{\tau(d_{s1},d_{d1})}(\Delta) \approx \begin{cases} 0, & \text{if } \Delta < \frac{2B}{C_l}, \\ \mathcal{F}_1, & \text{if } \frac{2B}{C_l} \leq \Delta < \frac{3B}{C_l}, \\ \mathcal{F}_2, & \text{if } \frac{3B}{C_l} \leq \Delta < \frac{4B}{C_l}, \\ 1, & \text{if } \frac{4B}{C_l} \leq \Delta, \end{cases} \tag{7.6}$$

where

$$\mathcal{F}_1 = \beta^2 + 2\beta(1-\beta)\left(\frac{C_l}{B}\right)\left(\Delta - \frac{2B}{C_l}\right) + (1-\beta)^2\left(\frac{C_l}{B}\right)^2 \frac{\left(\Delta - \frac{2B}{C_l}\right)^2}{2},$$
$$\mathcal{F}_2 = \frac{1+2\beta-\beta^2}{2} + (1-\beta)^2\left(\frac{C_l}{B}\right)^2 \left[\frac{B}{C_l}\left(\Delta - \frac{3B}{C_l}\right) - \frac{\left(\Delta - \frac{3B}{C_l}\right)^2}{2}\right],$$

with $\beta = \sqrt{(1-\rho)}$, where ρ is the queue utilization factor of any of the queues. Combining (7.4) and (7.6) we obtain the distortion at d_{d1}.

We compare the achieved distortion by both routing strategies, single path and multipath routing, theoretically and experimentally. Figure 7.7 shows the reduction in distortion in dB achieved by multipath routing over single path routing for a maximum delay $\Delta = \frac{2.5}{\mu}$ under different values of the network load ρ given by $\rho = \frac{B}{TC_l}$. Multipath routing clearly achieves a lower distortion than single path routing for all the network load values. The distortion reduction is maximum (about 5.5 dB) when $\rho = 0.5$, which corresponds to the maximum

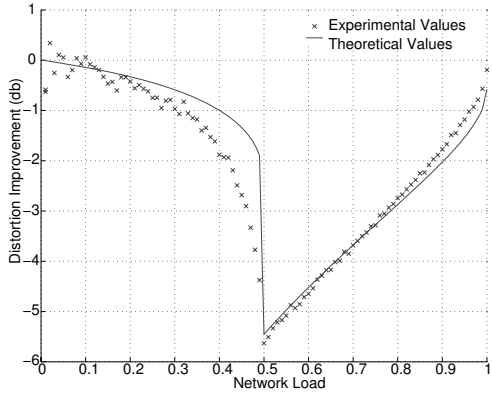


Figure 7.7: Multipath routing distortion improvement with respect to single path routing. The y-axis represents the distortion gap in dB and the x-axis indicates the network load. The solid line represents theoretical values while x-marks are results from simulations. The maximum delay is fixed at $\Delta = \frac{2.5}{\mu}$.

network load that the single path routing strategy can support. As we will show later, this reduction in distortion is more significant for small values of the maximum delay Δ. This gain is due to the more uniform load distribution achieved by multipath routing, which results in a more efficient usage of the network capacity. Note also that the theoretical approximation we derived closely follows the experimental results.

7.3.4 Double description coding and multipath routing

We have already shown the quantitative gain of using multipath routing with respect to single path routing from a distortion point of view. Furthermore, we can exploit the space diversity provided by the multipath routing algorithm and combine it with an MD coding that generates two descriptions, where each description follows a different path toward the destination.

We assume now that sources generate two descriptions of equal size $B/2$, and depending on the network conditions (congestion), sources add the necessary redundancy. These two descriptions are forwarded randomly using the two available paths, such that two descriptions of the same sample are not routed through the same path. We analyze the distortion achieved at the destination by the two descriptions coding schemes discussed in Section 7.2: MDSQ and UEP. Using an encoder based on MDSQ, the total distortion is computed as follows:

$$\begin{aligned} D_{\text{MDSQ}} = \quad & D_c F_{\tau(d_{s1},d_{d1})}(\Delta)^2 + \\ & 2D_s(1 - F_{\tau(d_{s1},d_{d1})}(\Delta))F_{\tau(d_{s1},d_{d1})}(\Delta) + \left(1 - F_{\tau(d_{s1},d_{d1})}(\Delta)\right)^2, \end{aligned} \tag{7.7}$$

where D_s and D_c are computed using (7.1). And using an encoder based on UEP:

$$D_{\text{UEP}} = D_c F_{\tau(d_{s1},d_{d1})}(\Delta)^2 + (D_s^1 + D_s^2)(1 - F_{\tau(d_{s1},d_{d1})}(\Delta))F_{\tau(d_{s1},d_{d1})}(\Delta) + \left(1 - F_{\tau(d_{s1},d_{d1})}(\Delta)\right)^2, \tag{7.8}$$

where D_s^1, D_s^2, and D_c are computed using (7.2).

To obtain the delay cumulative distribution $F_{\tau(d_{s1},d_{d1})}$ incurred by packets to reach the destination, we proceed as in the previous section: we model each link l_i as a $G/D/1$ queue q_i with a deterministic service time $1/\mu = B/2C_l$ and average arrival rate $\lambda = 2/T$, and analyze the resulting queueing network problem. Note that this analysis is identical to the one of previous section, where q_1 and q_2 present half the service time and twice the average arrival rate. Even though both queues present the same utilization factor as in the multipath and SD coding, the service time has been reduced by a factor of two, pointing out that the communication model we are considering presents a better performance for small packets.

We jointly optimize now two elements: given the routing algorithm and the coding scheme, we can calculate the loss probability of any individual description. Then, depending on this loss probability, we adjust the coding scheme to achieve the lowest distortion possible. Combining (7.7) and (7.8) with (7.6), we obtain the different values for the distortion at d_{d1} as a function of the "redundancy" we add in the descriptions. We adapt this redundancy to obtain the lowest possible distortion for every network condition.

We compare the achieved distortion using multi path routing with single and double description coding theoretically and experimentally. Figure 7.8 shows the benefits of using an MD coding over SD coding for different network loads $\rho = B/TC_l$. For each ρ, we compute the distortion achieved using MDSQ and UEP encoders for the central and side distortion pair (D_c, D_s) that achieves the lowest distortion.

The MD encoder clearly outperforms SD coding for all the values of ρ, the distortion improvement being more remarkable for low values of the network load. As expected, MDSQ slightly outperforms UEP, with a maximum distortion improvement of almost 1.6 dB for a network load $\rho = 0.125$. As we will show later, this reduction in distortion is more significant for small values of the maximum delay Δ. Note that as the network load ρ goes to 1, the distortion improvement goes to 0. The reason is that in a highly congested network (high ρ), almost all the packets reach the destination latter that the maximum allowed delay Δ. Consequently, the benefit of MD coding is limited. Note also that the theoretical approximation we derived closely follows the experimental results.

All the interactions between coding and routing we studied in this section are summarized in Table 7.1.

7.3.5 Multiple description coding in large networks

All the coding and routing interactions we studied in this section can be easily applied to more general networks. For illustration purposes, we compare through simulations the average end-to-end distortion achieved by the most sophisticated scheme, i.e., MD source coding using scalar quantizers and a multipath routing algorithm, with the usual SD coding and single path routing in a square grid network. The network model is identical to the one described in Chapter 3 under a uniform communication model. As a single path routing algorithm we use *row-first*, which achieves the maximum rate per node (Chapter 3). As multipath routing algorithm, we use *spreading*, which has the convenient property of using all the available paths between the source and the destination (Chapter 5). Figure 7.9 shows the distortion

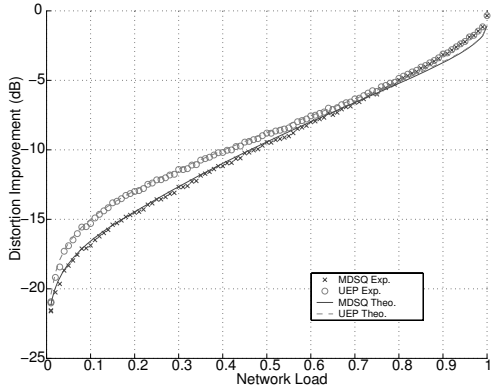


Figure 7.8: Distortion improvement using an MD encoder with respect to SD coding and multipath routing. The y-axis represents the distortion gap in dB and the x-axis indicates the network load. Lines represent theoretical values, x-marks and circles are results from simulations. The maximum delay is fixed at $\Delta = \frac{2.5}{\mu}$. For all network loads we code packets using MDSQ and UEP for the central side distortion pair (D_c, D_s) that achieves the lowest distortion.

achieved by both schemes as a function of the rate per node and the maximum tolerable delay Δ. As expected, the combination of an MDSQ encoder and multipath routing performs significantly better than single path single description encoder. The performance gap is more significant in the case of low rates and small Δ, where packet delay is more critical, and the advantage of using MD coding more evident. As we showed in Chapter 5, row-first is the optimal routing algorithm that achieves the maximum rate per node in a uniform communication scenario. Thus, the throughput achieved by row-first at high rates is higher than the throughput achieved by spreading. Consequently, the distortion improvement of MD coding and multipath routing decreases at high rates.

7.4 Multiple description coding and finite buffers: a practical case

In practice, common devices used in sensor networks have limited storage for the temporary storage of packets, which causes packet losses due to buffer overflow. The resulting loss probability distribution was thoroughly studied in Chapters 4 and 6. This is also a scenario where MD coding can be very useful: by sending descriptions along different paths we try to minimize the effects of overflow losses. Moreover, if these paths are sufficiently different, the quality of the received signal will be less sensitive to local congestion.

Routing	*Coding*	*Distortion Improvement (in dB)*		
		$\rho = 0.25$	$\rho = 0.5$	$\rho = 0.75$
Single Path	Single description	0	0	0
Multipath	Single description	0.6	5.5	3.3
Multipath	UEP: two descriptions	12.1	8.8	5.7
Multipath	MDSQ: two descriptions	13.6	9.4	5.9

Table 7.1: This table summarizes the different possible interactions between coding and routing. The first column is the routing mechanism and the second column shows the coding technique applied to data packets. The third columns indicates the distortion improvement (in dB) with respect to the single path SD coding for three different network load values, ρ= 0.25, 0.5, and 0.75, i.e., for low, moderate and high traffic.

While in the previous section we studied MD coding from a rate-distortion point of view, we concentrate now on the practical aspects of implementing an MD coding scheme in sensor networks with finite buffers where the set of available paths between any source-destination pair is large. To fully exploit the spatial diversity present in the network, we need an MD coding technique capable of generating an arbitrary number of descriptions. We start by reviewing some of the existing techniques to generate more than two descriptions and analyze their computational complexity. We compare the end-to-end distortion that these techniques achieve as a function of the number of descriptions for a simple network model characterized by a fixed probability of failure.

Then, we analyze the application of these coding techniques in a practical situation: a sensor network performing a data-gathering task. We analyze the optimal coding strategy as a function of practical parameters (such as header sizes, buffer sizes, and transmission rates) in addition to the number of descriptions. We show that although from a theoretical point of view the use of more descriptions provides a better performance, this is not the case when we consider practical constraints. We show experimentally that, depending on these practical parameters, there exists an optimal number of descriptions that achieves a minimum distortion.

7.4.1 From 2 to M descriptions

When the set of available routes between any source-destination pair is large, MD coding techniques capable of generating an arbitrary number of descriptions may be necessary. However, most of the previous work that addresses MD coding deals only with two descriptions, in which case close formulas for the rate-distortion curve have been obtained. In the previous section, we discussed two MD coding techniques used to generate two descriptions: Unequal Error Protection (UEP) [63; 64] and Multiple Description Scalar Quantization (MDSQ) [72]. We consider now the generalization of these two techniques to deal with an arbitrary number of descriptions. A generic MD coding scheme for three descriptions is depicted in Figure 7.10.

To compare the distortion achieved by different MD coding schemes and study the optimal number of descriptions, we start from a simple model of the network: we model each

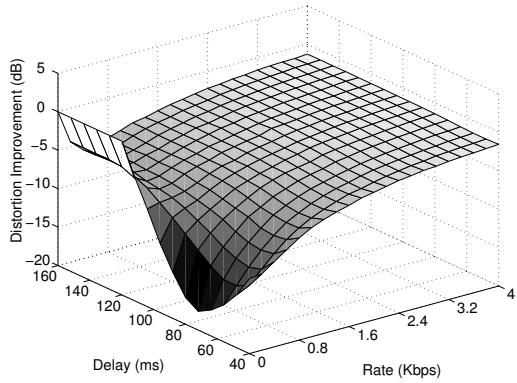


Figure 7.9: Distortion improvement achieved by using an MDSQ encoder and multipath routing with respect to single path and SD coding in a 400 nodes network. The z-axis represents the distortion gap in dB, the y-axis the maximum delay Δ and the x-axis the rate attempted per device. Each sample is encoded using 4 bits and the link capacity is fixed to $C_l = 1000$ bps.

path as a point-to-point channel characterized by a constant probability p_{loss} of losing a packet, independently of the packet size. We also assume that packet losses are independent among paths, and uncorrelated in time. If a packet reaches the destination (with probability $(1 - p_{\text{loss}})$), we assume that it does not contain bit errors, hence it can be correctly decoded. We consider a single source-destination pair with multiple paths between them, where the source is Gaussian with zero mean and variance σ^2. Under this scenario, we analyze the distortion achieved at the destination by different coding techniques as a function of the number of descriptions M.

One possible technique to generate an arbitrary number of descriptions of B_d bits each, consists in generalizing the UEP coding technique explained in the previous section. The information generated by the source is quantized by an MD encoder that generates a progressive bitstream of length MB_d bits and marks it at M different positions, each one corresponding to the attainment of a distortion level D_s^i $(1 \leq i \leq M)$ [63].

Assuming that each description follows a different path toward the destination, and denoting with k the number of descriptions out of M correctly received, the average distortion D at the destination is given by:

$$D = \sum_{k=1}^{M} \binom{M}{k} D_s^k (1 - p_{\text{loss}})^k p_{\text{loss}}^{M-k} + \sigma^2 p_{\text{loss}}^M. \tag{7.9}$$

The M optimal values of D_s^k that minimize (7.9) for a given p_{loss}, can be found using a

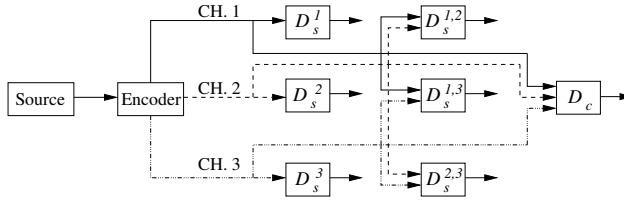


Figure 7.10: An MD system with three descriptions. Central decoder D_c receives all the descriptions and can reconstruct the finest representation of the source. The other decoders $D_s^{\{1,2,3\}}$ receive only a subset of the transmitted data. The quality of the received data is proportional to the number of descriptions received.

minimization method: we optimally tune the redundancy in each description using Lagrange multipliers. The output is then rounded to the nearest integer and the theoretical rate allocation is guaranteed over a long sequence of samples. This coding technique has the computational complexity of finding the optimal distortion values, which is linear with the number of descriptions.

Figure 7.11 shows the distortion achieved by an UEP encoder with the optimal redundancy at the descriptions as a function of the packet loss probability p_{loss} for a Gaussian source and a bitstream of 48 bits. We compare the minimum distortion achieved with 2 descriptions of 24 bits, 4 of 12, and 6 of 8. The 6-description encoder outperforms encoders that generate smaller number of descriptions for all packet loss probability, where the gain is more significant for low values of p_{loss}. These experimental results indicate that an increasing number of descriptions allows for a fine tuning of the redundancy we add, and consequently, achieves a lower distortion.

An alternative practical implementation of an MD coder is Multiple Description Scalar Quantization (MDSQ). A M-description MDSQ encoder is basically as a central quantizer and an *index assignment* in M dimensions: for each source scalar input, the encoder produces M quantization indices. These indices can be seen as indices of row and column of a M-dimensional *index assignment hyper-cube* in which each dimension has size 2_d^B cells. The idea is to fill this *hyper-cube* with no more than 2^{MB_d} numbers according to the packet loss probability p_{loss}, such that the difference of these numbers in all the *hyper-planes* is minimized [9]. If only a subset of indices is received, the decoder can reconstruct the best coarse version of the original signal and the quality increases with the number of received descriptions.

However, the problem of completely filling the hyper-cube while minimizing the difference in each hyper-plane is NP-complete [9]. In practice, to achieve a trade-off between central and side distortions, we only need to fill the hyper-cube partially. Note that completely filling the hyper-cube corresponds to minimizing the central distortion. To the best of our knowledge, there does not exist an efficient optimization algorithm to partially fill the hyper-cube, and the only possible solution is an exhaustive search. Therefore, the complexity of generating the optimal hyper-cuber increases exponentially with the bits per description and with the number of descriptions. The complexity of dynamically adapting an MDSQ

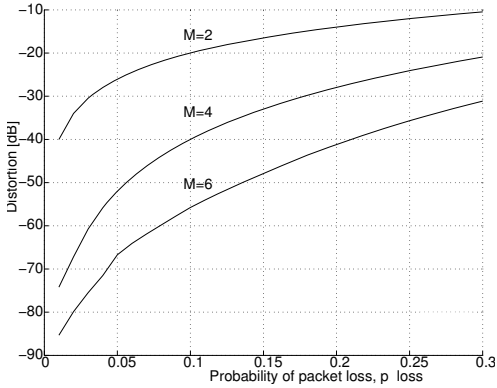


Figure 7.11: Minimum distortion achieved by an UEP encoder with a Gaussian source and a bitstream of 48 bits, for different number of descriptions M as a function of the packet loss probability p_{loss}.

encoder to different operating points discourages a real-time implementation of this method for more than two descriptions. For practical implementations, we thus consider only a two-description MDSQ encoder.

7.4.2 A practical scenario: data gathering

The network model we considered in the previous section was simply characterized by a constant probability p_{loss} of losing a packet, independent of the number of descriptions. Under these assumptions, we showed that a higher number of descriptions achieved a lower distortion at the destination. However, in many practical scenarios, these assumptions are not valid, and a higher number of descriptions does not necessarily lead to a lower distortion. Packet loss probability clearly depends on many network parameters such as packet size and network congestion, and in general, this loss probability is highly correlated in time and among paths. We consider now all these practical parameters and analyze the use of MD coding in a particular situation: central data gathering in a sensor network.

We measure a Gaussian random field X on a square area, with X being uncorrelated in space and time. We uniformly place N sensor devices that sample the field and send all the information to a single device (base-station), which gathers all the information generated by the network and reconstructs the field. The resulting network is shown in Figure 7.12. The network model we consider is identical to the square grid model described in Chapter 3.

We assume that each sensor generates samples uniformly with a rate of $\mathcal{R}$ samples per unit time. More specifically, nodes generate one packet in every interval of duration $(1/\mathcal{R})$ seconds. Within an interval, the exact packet origination time is uniformly distributed from 0 to $(1/\mathcal{R})$, independently among nodes. We further assume that the packet generation process

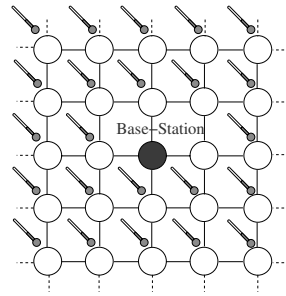


Figure 7.12: Network model: a uniform placement of devices that measure a random field and transmit the data to a base-station located in the center of the network.

is independent among nodes. To increase the communication efficiency, sensors group S_p samples in one single data packet, and send it to the base-station. We assume that S_p is determined by the application requirements. In practice, in addition to the samples (payload of the packet), data packets contain also a header that includes information such as the device identifier, sampling times, or sequence numbers. We denote by H the size in bits of the header.

The sensors encode the generated samples and produce M descriptions. In order to make the comparison fair among different coding schemes, we assume that the total number of bits per sample is fixed. That is, we generate M descriptions of $B_d = B/M$ bits each so that the information rate transmitted by the network is constant regardless of the number of descriptions. As S_p descriptions are included in the same packet, the total packet size K in bits is given by $K = H + S_p \frac{B}{M}$.

The nodes route packets towards the base-station using the spreading algorithm (Chapter 2), which have the convenient property of using all the available paths between the source and the base-station. Note that spreading allows to exploit the network diversity present in the network, making the use of multiple description coding very convenient.

We assume that nodes are equipped with limited buffer capabilities for the temporary storage of Q packets. When packets arrive at a particular node or are generated by the node itself, they are placed into the buffer until the node has the opportunity to transmit them through the required link. If the buffer is full, the newly arrived packet is dropped and no retransmission is attempted. We consider that buffer overflow is the only source of packet loss.

We assume that the packet transmission time between one node and any of its neighbors is proportional to the packet size and model each link as a single-server queue with a deterministic service time equal to the packet transmission. In consequence, the probability p_{loss} of losing a packet for a given source depends on the network parameters such as the transmission rate $\mathcal{R}$, the packet size K, and the number of descriptions M.

With all the information generated by the nodes, the base-station reconstructs the field X. We measure the distortion D between the real field X and the reconstructed field $\hat{X}$ as the

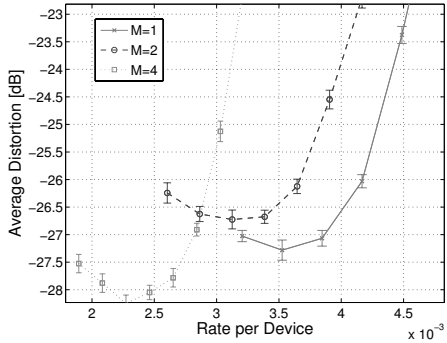


Figure 7.13: Distortion at the base-station as a function of the transmission rate $\mathcal{R}$ in a 25×25 network using a UEP encoder that generates 1,2, and 4 descriptions with $B = 8$ bits.

average mean-square-error (MSE) per device and unit time. That is,

$$D = \frac{\sum_{i=1}^{N} (X(i) - \hat{X}(i))^2}{N\mathcal{R}}. \tag{7.10}$$

7.4.3 Optimal transmission strategy

We investigate now the optimal transmission strategy that achieves the lowest distortion, that is, $\{\mathcal{R}^{\text{opt}}, M^{\text{opt}}, K^{\text{opt}}\} = \arg\min_{\{\mathcal{R},M,K\}} D$. First, note that for any number of descriptions M and packet size K, there exists an optimal transmission rate $\mathcal{R}^{\text{opt}}(M, K)$ that minimizes the distortion. A higher transmission rate per node allows a more accurate field reconstruction at the base-station. However, if the network becomes congested, the packet loss probability increases due to buffer overflow and the distortion increases. The optimal value for $\mathcal{R}^{\text{opt}}(M, K)$ can be computed by analyzing the associated queueing network using the approximation tools shown in Chapter 4. For illustration purposes, Figure 7.13 shows the distortion at the base-station for values of $\mathcal{R}$ around $\mathcal{R}^{\text{opt}}(M, K)$ and different number of descriptions M. We observe how the distortion decreases with $\mathcal{R}$ until the network becomes congested and overflow losses start increasing distortion. These graphs where obtained by simulating a 25×25 network using a UEP encoder with $B = 8$ bits, $H = 36$ bits, and $S_p = 20$.

We analyze now the optimal number of descriptions that minimizes distortion. First, note that due to the practical constraints in real applications, every information packet needs to include a header. Therefore, the traffic transmitted through the network increases with the number of descriptions. Moreover, the characteristics of the input traffic of the queues also changes. If we assume that each description follows a different path towards the base-station, this multipath routing generates an arrival distribution to the queues that increases overflow

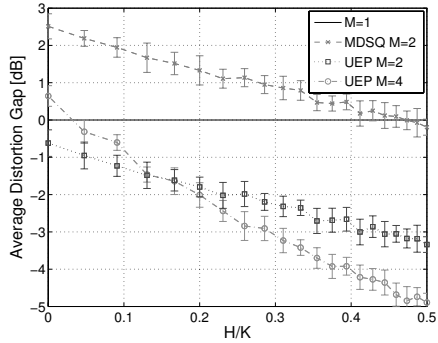


Figure 7.14: Distortion reduction at the base-station using an MD encoder with respect to SD coding as a function of the header-payload ratio using 8 bits per sample.

losses (as discussed in Chapter 4). These two factors increase the packet loss probability and consequently, increase distortion.

On the other hand, as we have shown in the previous section, an increasing number of descriptions allows for a fine tuning of the redundancy we add, and consequently, achieves a lower distortion. Moreover, if we allow for different packet sizes, the size K of each packet is reduced with the number of descriptions. From a queueing point of view, this packet size reduction decreases network congestion and, equivalently, decreases overflow losses. This is illustrated through a simple example in Appendix 7.B.

This suggests that there exists an optimal number of descriptions M^{opt} that depends on the packet size K, and more particularly on the ratio H/K. We compare the achieved distortion using an UEP encoder for different number of descriptions as a function of the ratio H/K. For each (M, K) pair, we let nodes transmit at the optimal rate $\mathcal{R}^{\text{opt}}(M, K)$. Figure 7.14 shows the distortion improvement achieved by using $M = \{1, 2, 4\}$ descriptions with respect to SD coding as a function of H/K in a 25×25 network with $S_p = 20$ and $B = 8$ bits. Similarly, Figure 7.15 shows the distortion for $M = \{1, 2, 3, 4, 6\}$ descriptions with $B = 12$ bits. We compare also the use of UEP and MDSQ encoders. Due to its exponential complexity, we considered MDSQ only for 2 descriptions and $B = 8$.

In the case of $B = 8$ bits, the optimal strategy for $H/K < 0.5$ consists in generating 2 descriptions using an MDSQ encoder. As expected, the benefit of using 2 descriptions decreases as the rate H/K increases due to the traffic increase. For values of $H/K > 0.5$, the lowest distortion is achieved by SD coding. For instance, in some media access mechanisms for wireless networks each packet needs to include a long preamble for its correct reception (B-MAC [60]). In this scenario, the use of MD coding might be disadvantageous.

In the case of $B = 12$ bits, due to the exponential complexity of MDSQ with the bits per sample, we considered only a UEP encoder (even for the simplest case of $M = 2$, an MDSQ encoder requires to generate a $2^{B/2} \times 2^{B/2}$ matrix where $2^{B/2+1} - 1$ diagonals

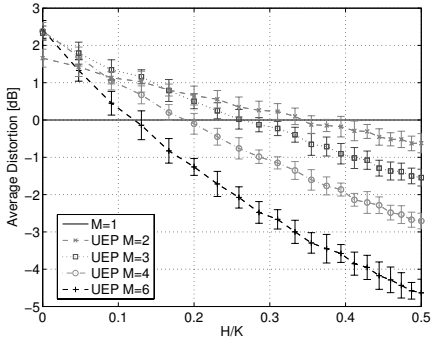


Figure 7.15: Distortion reduction at the base-station using an MD encoder with respect to SD coding as a function of the header-payload ratio using 12 bits per sample.

can be filled). For small H/K values ($H/K < 0.15$), the lowest distortion is achieved by generating three descriptions, for which a gain of almost 2.5 dB is achieved with respect to SD coding. To generate more than three descriptions does not decrease the distortion. As the header size increases, the performance of MD degrades due the traffic increase. Obviously, the more descriptions we generate, the more rapidly it degrades. Consequently, the optimal number of descriptions decreases when the ratio H/K increases: for values of H/K between 0 and 0.15, the optimal strategy consists in generating 3 descriptions. When H/K is between 0.15 and 0.35, the optimal is $M = 2$. Finally, when $H/K > 0.35$ we use SD coding. From previous results we can also conclude that the use of MD coding is clearly more advantageous in systems with a high number B of bits per sample.

These results differ from the theoretical results, where an increasing number of descriptions always achieves a lower distortion. Depending on practical network parameters such as header and packet sizes, there exists an optimal number of descriptions that achieves the lowest distortion. We see here that this optimal parameter depends heavily on these network parameters and need to be explored for each application.

7.5 Summary

In this chapter, we investigated the interaction of the source coding mechanism and the routing mechanism: we showed that there is much to be gained if the upper layers are also considered in the design of routing algorithms.

We presented multiple description (MD) coding as the natural source coding to use over dense networks that provide multiple paths between any source-destination pair. MD coding is a powerful approach to combat packet losses in networks where retransmission is not al-

ways possible. We studied the use of MD coding in two different scenarios: real time data transmission and central data gathering in a network with constrained buffers.

In a real time data transmission scenario, we analytically computed the distortion at the receiver for a simple network model. We showed that the most sophisticated scheme (multiple descriptions source coding and multipath routing) performs significantly better than the usual single path and single description scheme. The improvement is larger in the case of low rates and small hard time constraint.

We studied also the generalization of two well-known MD coding techniques to handle an arbitrary number of descriptions. We applied these techniques in a square grid network with small buffers performing central data gathering. We showed that, depending on the network conditions, there exists an optimal number of descriptions that achieves the minimum distortion. This indicates that practical parameters, such as header sizes and rate per node, should be carefully taken into account when setting up a multiple description coding system.

7.A Queueing network analysis

7.A.1 Single path routing

We analyze in this Appendix the equivalent queueing network problem for the single path single description scheme illustrated in Figure 7.5. We model both links l_1 and l_2 as two $G/D/1$ queues, q_1 and q_2 respectively, with a constant service time $1/\mu = B/C_l$. The average arrival rate to q_1 is given by $\lambda_{q_1} = 2/T$. Since the arrivals to q_2 come only from the output of q_1, the interarrival time between any two arrivals is larger than the service time $1/\mu$. Therefore, we can model q_2 as a constant delay of $1/\mu$.

The system delay probability density function $f_{\tau(d_{s1},d_{d1})}$ can be calculated as the product of these two queues delay probability density functions in the Laplace transform domain:

$$f_{\tau(d_{s1},D1)}(s) = f_{\tau(d_{s1},d_{s2})}(s) f_{\tau(d_{s2},d_{d1})}(s). \tag{7.11}$$

To compute the delay distribution $f_{\tau(d_{s1},S2)}$ associated to the first queue q_1, we start by deriving the arrival process A_i to q_1. Since l_1 represents a simplex channel, q_1 receives the addition of the traffic generated in d_{s1} and in d_{s2}. Therefore, its arrival can be described as:

$$A_i = A_i^1 + A_i^2, \qquad i = [0, 1, ...], \tag{7.12}$$

where,

$$A_i^1 = i.T + a_1(i),$$
$$A_i^2 = i.T + a_2(i),$$

with a_1 and a_2 being two independent random variables uniformly distributed between 0 and T. To compute the interarrival time distribution $A(t)$ induced by this process we consider two consecutive intervals, the $(i-1)$-th and the i-th, of length T and denote by t_i the time between two consecutive arrivals. We compute now the cumulative distribution function $p(t_i \leq d)$, where d takes values in the interval $(0, 2T)$.

We define a new random variable y_i, that represents the arrival order of A_i^1 in the i-th interval. That is, $y_i = 1$ if $A_i^1 < A_i^2$, and 2 otherwise. Then:

$$p(t_i \leq d) = \textstyle\sum_{y_i \in [1,2]} p(t_i \leq d/y_i) p(y_i). \tag{7.13}$$

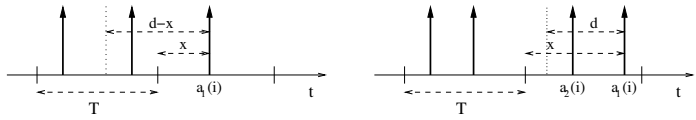


Figure 7.16: (a) Computation of $p(t_i \leq d/y_i = 1) = 0$. (b) Computation of $p(t_i \leq d/y_i = 2) = 0$.

We define:

$$P_1 = p(t_i \leq d/y_i = 1)p(y_i = 1),$$
$$P_2 = p(t_i \leq d/y_i = 2)p(y_i = 2).$$

We analyze now separately two different intervals: $[0 \leq d \leq T]$ and $[T \leq d \leq 2T]$.

- First case: $[0 \leq d \leq T]$:

If $y_i = 1$ and $a_1(i) \geq d$, then clearly $p(t_i \leq d/y_i = 1) = 0$. If $a_1(i) \leq d$, the probability that one arrival from previous interval happens in the period $(A_i^1 - d, A_i^1)$ is given by $1 - (\frac{T-d+x}{T})^2$. This is illustrated in Figure 7.16(a). Putting all together, P_1 is given by the following expression:

$$P_1 = \int_0^d \frac{1}{T^4} \left(\frac{T-x}{T}\right) \left(1 - \left(\frac{T-d+x}{T}\right)^2\right) dx.$$

On the other hand, if $y_i = 2$, that is $a_1(i) > a_2(i)$, then the probability of A_i^2 to arrive in the period $(A_i^1 - d, A_i^1)$ is given by:

$$P_2 = \int_0^{T-d} \frac{1}{T}\frac{d}{T} dx + \int_0^d \frac{1}{T}\frac{x}{T} dx.$$

This situation is illustrated in Figure 7.16(b).

- Second case: $[T \leq d \leq 2T]$:

If $a_1(i) \leq d - T$, given that $T \leq d$, the probability of one arrival in the $(i-1)$-th interval between $(A_i^1 - d, A_i^1)$ is obviously equal to 1. Thus we can divide P_1 in two contributions: $0 \leq a_1(i) \leq d - T$ and $d - t \leq a_1(i) \leq T$. If $d - t \leq a_1(i) \leq T$, then the probability that there is one arrival in the $(i-1)$-th interval between $(A_i^1 - d, A_i^1)$ is given by $1 - (\frac{T-d+x}{T})^2$. Adding both contributions we have:

$$P_1 = \int_0^{d-T} \frac{1}{T}(\frac{T-x}{T})dx + \int_{d-T}^T \frac{1}{T}(\frac{T-x}{T})(1 - (\frac{T-d+x}{T})^2)dx. \qquad (7.14)$$

If A_i^1 is the second arrival in the i interval, that is $a_1(i) > a_2(i)$, the probability of A_i^2 to arrive in the period $(A_i^1 - d, A_i^1)$ is given by:

$$P_2 = \int_0^T \frac{1}{T}\frac{x}{T} dx$$

Putting all together, the interarrival time distribution $A(t)$ in q_1 is given by:

$$A(t) = \begin{cases} \frac{t(t^3-8t^2T+6tT^2+12T^3)}{12T^4}, & \text{if } t \leq T \\ \frac{-4T^4+32tT^3-24t^2T^2+8t^3T-t^4}{12T^4}, & \text{if } T \leq t \leq 2T. \end{cases}$$

Using this interarrival distribution we can easily compute the delay probability distribution $f_{\tau(d_{s1},S2)}$ applying the $G/D/1$ formulas [38]. To simplify the analysis we make the following approximation: under low to moderate load, the system time cumulative distribution function can be approximated by the following linear expression:

$$F_{\tau(d_{s1},d_{s2})}(t) \approx \left[\beta + \mu(1-\beta)\left(t-\frac{1}{\mu}\right)\right]\left(u\left(t-\frac{1}{\mu}\right) - u\left(t-\frac{2}{\mu}\right)\right) + u\left(t-\frac{2}{\mu}\right), \tag{7.15}$$

where we defined $\beta = \sqrt{(1-\rho_1)}$ and ρ_1 is the queue utilization factor of q_1.

Using this approximation and 7.11, the overall delay cumulative distribution function $F_{\tau(d_{s1},d_{d1})}$ is given by:

$$F_{\tau(d_{s1},d_{d1})}(t) \approx \begin{cases} 0, & \text{if } t \leq \frac{2}{\mu}, \\ \beta + \mu(1-\beta)(t-\frac{2}{\mu}), & \text{if } \frac{2}{\mu} \leq t < \frac{3}{\mu}, \\ 1, & \text{if } \frac{3}{\mu} \leq t. \end{cases}$$

7.A.2 Multipath routing

We analyze in this section the equivalent queueing network problem for the multipath single description scheme illustrated in Figure 7.6. We model both links l_1 and l_2 as two $G/D/1$ queues, q_1 and q_2 respectively, with a constant service time $1/\mu = B/C_l$ and an average arrival rate $\lambda = 1/T$. Note that both available paths for the pair (d_{s1}, d_{d1}) are equivalent. We concentrate on the analysis of the path (d_{s1}, d_{s2}, d_{d2}).

To resolve this queues system, we use Kleinrock's independence approximation [37]: several packet streams on a transmission line has an effect akin to restoring the independence of inter arrival times. Given that we have a new incoming flow that arrives at q_2 (the new packets generated at d_{s2} that reach d_{d2} through d_{d1}), we can assume that interval times and packet lengths are independent, and compute the system delay probability density function as the product of two identical probability density functions in the transform domain. That is,

$$f_{\tau(d_{s1},D1)}(s) = f_{\tau(d_{s1},S2)}(s) f_{\tau(d_{s2},D1)}(s) = \left(f_{\tau(d_{s1},S2)}(s)\right)^2. \tag{7.16}$$

The analysis of the queue q_1 is similar to the analysis in the single description case (Appendix 7.A.1). Operating on (7.15), the Laplace transform of the probability density function is given by:

$$f_{\tau(d_{s1},S2)}(s) \approx \beta e^{-\frac{s}{\mu}} + \frac{(1-\beta)\mu}{s}\left(e^{-\frac{s}{\mu}} - e^{-\frac{2s}{\mu}}\right). \tag{7.17}$$

Combining (7.16) and (7.17), the system delay probability density function is given by:

$$\begin{aligned} f_{\tau(d_{s1},d_{d1})}(t) &= \mathcal{L}^{-1}(\left(f_{\tau(d_{s1},S2)}(s)\right)^2) \\ &\approx \mathcal{L}^{-1}\left[\beta^2 e^{-\frac{2}{\mu}s} + \frac{(1-\beta)^2\mu^2}{s^2}\left(e^{-\frac{1}{\mu}s} - e^{-\frac{2}{\mu}s}\right)^2 + \frac{2\beta(1-\beta)\mu}{s}\left(e^{-\frac{2}{\mu}s} - e^{-\frac{3}{\mu}s}\right)\right]. \end{aligned} \tag{7.18}$$

Computing the inverse Laplace transform and integrating, we derive the probability density function common to both descriptions:

$$F_{\tau(d_{s1},d_{d1})}(\Delta) \approx \begin{cases} 0, & \text{if } \Delta < \frac{2B}{C_l}, \\ \mathcal{F}_1, & \text{if } \frac{2B}{C_l} \leq \Delta < \frac{3B}{C_l}, \\ \mathcal{F}_2, & \text{if } \frac{3B}{C_l} \leq \Delta < \frac{4B}{C_l}, \\ 1, & \text{if } \frac{4B}{C_l} \leq \Delta, \end{cases} \tag{7.19}$$

where

$$\mathcal{F}_1 = \beta^2 + 2\beta(1-\beta)\left(\frac{C_l}{B}\right)\left(\Delta - \frac{2B}{C_l}\right) + (1-\beta)^2\left(\frac{C_l}{B}\right)^2 \frac{\left(\Delta - \frac{2B}{C_l}\right)^2}{2},$$
$$\mathcal{F}_2 = \frac{1+2\beta-\beta^2}{2} + (1-\beta)^2\left(\frac{C_l}{B}\right)^2 \left[\frac{B}{C_l}\left(\Delta - \frac{3B}{C_l}\right) - \frac{\left(\Delta - \frac{3B}{C_l}\right)^2}{2}\right],$$

with $\beta = \sqrt{(1-\rho)}$, where ρ is the queue utilization factor of any of the queues.

7.B Queueing example

Consider just a single $M/M/1/Q$ queue with a buffer size of Q packets, average arrival rate λ, and average service time μ. Applying $M/M/1/Q$ formulas, the utilization factor is given by $\rho = \lambda/\mu$, and the probability of packet overflow p_o by:

$$p_o = \frac{1-\rho}{1-\rho^{Q+1}}\rho^Q. \tag{7.20}$$

Suppose now that we generate M times more packets of $1/M$ the size. Given that the service time is proportional to the packet size, this is equivalent to consider $\lambda' = M\lambda$ and $\mu' = M\mu$ and $Q' = MQ$. Obviously, the utilization factor does not change, that is, $\rho' = \rho$. However, the new probability of packet overflow p'_o is clearly reduced:

$$p'_o = \rho^{(M-1)Q}\frac{1-\rho^{Q+1}}{1-\rho^{MQ+1}}p_o < p_o. \tag{7.21}$$

Note also that as the average waiting time in the queue is proportional to $1/\mu$, packet delay is also reduced. Consequently, to reduce the probability of losing packet due to buffer overflow, smaller packets are preferable.

Chapter 8

Conclusions

The difficulties of routing in large scale sensor networks mainly stem from the constraints imposed by the simplicity of sensor devices: limited power, limited communication bandwidth and processing capabilities, and small storage capacity. In this thesis, we concentrated on the study and design of distributed routing algorithms for sensor networks: routing decisions at each node are only based on local information. Particularly, we analyzed the routing problem for common traffic scenarios encountered in sensor networks, namely uniform communications and data gathering. We also studied the routing problem under common limitations imposed by sensor nodes such as small buffers and temporary node failures. For each of these scenarios, we studied capacity limits and derived constructive routing strategies that maximize the transmission rate per node. We now summarize the key results of our work and discuss open issues.

8.1 Summary of main results

Point-to-point routing

We started by studying the problem of designing point-to-point routing algorithms for large scale sensor networks. We formulated the routing problem as a problem of constructing suitable random walks on random dynamic graphs. Particularly, we designed constrained random walks on a particular family of random graphs (random lattice networks) that we chose as an abstraction for the behavior of large sensor networks. We presented distributed algorithms for computing the local parameters of random walks that induced a uniform traffic distribution in the network. Using this routing formulation, we were able to route messages without any notion of discovering / maintaining / repairing routes.

Multiple sources and/or destinations routing

As a natural extension to point-to-point routing, we considered the problem of multiple sources and/or destinations routing in lattice networks. Particularly, we considered three different communication models: uniform communication, central data gathering, and border data gathering. For each of these models, we studied capacity limits and derived constructive routing strategies that achieve this capacity.

Finite buffers

One of the main limitations of sensor nodes is their small memory size, which translates into a limited space for the temporary storage of packets. In this thesis, we analyzed routing algorithms for networks where nodes have a limited buffer space. This analysis required to compute the distribution on the queue size at the nodes, which is a hard problem even for simple networks. We proposed alternative approximation models to the usual Jackson's Theorem (or independence approximation) to obtain a more accurate distribution on the queue size at the nodes. Using these approximations, we derived the routing condition to minimize overflow losses: nodes receive all traffic exclusively from one neighbor. According to this condition, we designed the routing algorithms that maximized the throughput per node under the same three communication models mentioned above.

Robust routing and capacity

We considered also the problem of routing in large and unreliable networks such as those composed of sensors nodes. We studied routing algorithms that maximize the rate per node for dynamic networks under two different failure modes: a Markovian model that reproduces failures due to malfunction in the nodes, and an energy limited model which is related to depletion of communication resources. We showed that achieving robust communications and maximizing the achievable rate-per-node are incompatible goals: while robust communications require using of as many paths as possible between the source and the destination, maximizing the rate per node requires to use only a few of the available paths. We showed how to explore this trade-off, depending on the degree of unreliability of the network. We proposed the use of a particular combination of two routing algorithms, the first one being optimal when there are no node failures at all and the second one being appropriate when the probability of node failure is high. The combination of these two routing algorithms defined a family of randomized routing algorithms, each of them being suitable for a given probability of node failure.

Random networks

Even if random networks are very different in nature from grid networks, we showed that similar principles can be applied to analyze the buffer occupancy. We analyzed first a collision-free network model for both infinite and finite buffers at the nodes. We studied the network capacity and showed that for both infinite and finite buffers, the problem of finding the optimal routing algorithm that maximizes the rate per node is an NP-hard problem. Then, we proposed distributed approximation algorithms that behave close to optimal. These approximation algorithms achieved an average maximum rate considerably higher than the rate achieved with the usual shortest path routing algorithm.

Finally, we discussed the wireless case, where packet transmissions among different nodes can interfere and result in a packet collision. We showed that the overflow loss process is quite similar in both collision-free and wireless models, and proposed a simple method to reduce overflow losses in wireless networks. This method consisted in adapting the back-off window size linearly with the distance to the base station. This mechanism allowed to significantly decrease overflow losses in the network.

Sensor networks and multiple description coding

Finally, we investigated the interaction of the source coding mechanism and the routing mechanism: we showed that there is much to be gained if the upper layers are also considered in the design of routing algorithms. We presented multiple description coding as the natural source coding to use over dense and unreliable networks. Particularly, we considered the use of multiple description coding in two different scenarios: real time data transmission and central data gathering in a network with constrained buffers. We showed that in both scenarios, the jointly optimization of the source coding and the routing mechanisms performed significantly better than the usual source coding and routing schemes. We studied also the generalization of two well-known multiple description coding techniques to handle an arbitrary number of descriptions. We applied these techniques in a square grid network with small buffers performing central data gathering and showed that, depending on the network conditions, there exists an optimal number of descriptions that minimizes distortion.

8.2 Discussion and future research

Delay

The goal of the routing algorithms we proposed in most of the chapters was to maximize the rate per node: We calculated network capacities and presented routing policies that achieved this capacity. However, as we have discussed in Chapter 7, many real applications have a maximum tolerable delay, above which data becomes useless. A system may be designed to deliver high throughput at heavy load, and yet it may experience intolerable delays at light load. In these scenarios, a performance measure that combines delay and throughput is needed. For instance, a compact measure of combined throughput and delay is offered by the concept of "power" [24], whose simplest definition is the ratio of throughput over delays.

Energy

Sensor networks are tightly constrained in terms of energy, and hence a careful resource management is required. In sensor devices, most of the energy is used in the radio communications. Some of the optimal routing algorithms we have presented required a significantly higher number of transmissions than suboptimal algorithms. For instance, the routing algorithm that minimizes overflow losses in lattice networks required $\mathcal{O}(N)$ transmissions, while a shortest path routing algorithm requires only $\mathcal{O}(\sqrt{N})$. It is of interest to explore this trade-off between energy and throughput, and the possible connections with the delay.

Data correlation

Besides path diversity, a dense network provides an enormous quantity of information about the physical process being measured. This information is typically highly correlated in time and space among sensors. This correlation can be exploited to overcome sensor network limitations by designing transport protocols and coding schemes that trade this redundancy. Data correlation also allows to reduce the information rate that nodes inject into the network. For instance, intermediate nodes can perform data aggregation and caching in addition to routing. This could be especially useful in the optimal routing strategies that minimize buffer overflow, where we forced Hamiltonian paths to reach the destination. One line along which

this work could proceed further consists in studying the interaction of the routing problem for finite buffers and the correlation in the data.

Wireless sensor networks

The results presented for wireless random networks are preliminary results of an ongoing research. For instance, we restricted the possible routing algorithms to the class of shortest path routing, i.e., messages transmitted between any two nodes can only be routed following a shortest path. However, this class of routing algorithms does not necessarily lead to the best possible solution in terms of maximizing the rate per node.

Real measurements

Given the difficulty in performing actual measurements in wireless networks, all the algorithms have been evaluated through simulations. Therefore, a logical extension of this work would be to implement and evaluate all the algorithms in a real wireless network.

Bibliography

[1] T. Ajdler, L. Sbaiz, and M. Vetterli. The plenacoustic function and its sampling. *IEEE Transactions on Signal Processing*, 2005.

[2] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, Dec. 2004.

[3] M. Alasti, K. Sayrafian-Pour, A. Ephremides, and N. Farvardin. Multiple description coding in networks with congestion problem. *IEEE Transactions on Information Theory*, 47(3):891–902, March 2001.

[4] E. Arikan. Some complexity results about packet radio networks. *IEEE Transactions on Information Theory*, 30(4), 1984.

[5] E. Ayanoglu, C. Gitlin, and J. Mazo. Diversity coding for transparent self-healing and fault-tolerant communication networks. *IEEE Transactions on Communications*, 41(11):1677–1686, Nov. 1993.

[6] H. G. Badr and S. Podar. An optimal shortest path routing policy for network computer with regular mesh-connected topologies. *IEEE Transactions on Computers*, 38(10), Oct. 1989.

[7] A. Banerjea. On the use of dispersity routing for fault tolerant realtime channels. *European Trans. Telecommun.*, 8(4):393–407, 1997.

[8] G. Barrenetxea, B. Beferull-Lozano, and M. Vetterli. Lattice sensor networks: Capacity limits and optimal routing. *IEEE Transactions on Networking*. (Accepted for publication).

[9] T. Y. Berger-Wolf and M. A. Harris. Sharp bounds for bandwidth of clique products. *CoRR*, cs.DM/0305051, 2003.

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

[11] D. Bhatia, T. Leighton, F. Makedon, and C. H. Norton. Improved algorithms for routing on two-dimensional grids. *Lecture Notes in Computer Science*, 657, 1993.

[12] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Macmillan, London, 1976.

[13] F. Borgonovo and E. Cadorin. Locally optimal deflection routing in the bidirectional Manhattan network. In *INFOCOM, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies*, volume 2, pages 458 –464, 1990.

[14] S. Chen and K. Nahrstedt. Distributed quality of service routing in ad-hoc networks. *IEEE J. Select. Areas Commun.*, 17(8):1488–1505, 1999.

[15] R. D'Andrea and G. E. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9):1478–1495, Sep. 2003.

[16] K.A. Delin. Sensor web in antarctica: Developing an intelligent, autonomous platform for locating biological flourishes in cryogenic environments. In *34th Lunar and Planetary Science Conf. (LPSC 03)*, 2003.

[17] E. J. Duarte-Melo and M. Liu. Data-gathering wireless sensor networks: organization and capacity. *Computer Networks*, 43(4), 2003.

[18] R. Duncan. A survey of parallel computer architectures. *IEEE Computer*, 23(2), 1990.

[19] A. Ephremides and T. V. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Trans. Communications*, 38(4):456–460, April 1990.

[20] A. A. El Gamal and T. Cover. Achievable rates for multiple description. *IEEE Transactions on Information Theory*, IT-28:851–857, November 1982.

[21] A. El Gamal. Trends in cmos image sensor technology and design. In *IEEE International Electron Devices Meeting*, Dec. 2002.

[22] D. Ganesan, A. Cerpa, Y. Yu, W. Ye, J. Zhao, and D. Estrin. Networking issues in sensor networks in the journal of parallel and distributed computing (jpdc). Invited paper in Special issue on Frontiers in Distributed Sensor Networks, Elsevier Publishers, 2003.

[23] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[24] M. Gerla and L. Kleinrock. Flow control: a comparative survey. *IEEE Transactions on Communications*, 28(4):553–574, April 1980.

[25] V. K. Goyal. Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine*, pages 74–93, Sept 2001.

[26] M. D. Grammatikakis, D. F. Hsu, M. Kraetzl, and J. F. Sibeyn. Packet routing in fixed-connection networks: A survey. *Journal of Parallel and Distributed Computing*, 54(2):77–132, 1 November 1998.

[27] M. E. Grismer. Field sensor networks and automated monitoring of soil-water sensors. *Soil Science*, 154(6):482–489, 1992.

[28] Y. Gsottberger, X. Shi, G. Stromberg, T. F. Sturm, and W. Weber. Embedding low-cost wireless sensors into universal plug and play environments. In *EWSN*, 2004.

[29] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. Info. Theory*, 46(2):388–404, March 2000.

[30] B. Hajek. Balanced loads in infinite networks. *Annals of Applied Probability*, 6:48–75, 1996.

[31] M. Harchol-Balter and P. Black. Queueing analysis of oblivious packet-routing networks. In Daniel D. Sleator, editor, *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 583–592, Arlington, VA, January 1994. ACM Press.

[32] J. P. Hubaux, T. Gross, J. Y. Le Boudec, and M. Vetterli. Towards self-organized mobile ad hoc networks: the Terminodes project. *IEEE Communications Magazine*, 31(1):118–124, 2001.

[33] NAB (Network in A Box). http://nab.epfl.ch.

[34] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, February 2003.

[35] N. Jain, D. Madathil, and D. Agrawal. Energy aware multi-path routing for uniform resource utilization in sensor networks. In *IPSN*, April 2003.

[36] F. P. Kelly. Network routing. *Proc. 13th int. teletraffic cong. Supplementary paper workshop on stochastic modelling*, 1991.

[37] L. Kleinrock. *Communication Nets*. McGraw-Hill, New York, 1964.

[38] L. Kleinrock. *Queueing Systems, Volume I: Theory*. Wiley, New York, 1975.

[39] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (Eds). *The traveling salesman problem: A guided tour of combinatorial optimization*. Wiley & Sons, New York, 1985.

[40] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan-Kaufman, 1991.

[41] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA-02)*, 2002.

[42] D. Marco, E. J. Duarte-Melo, M. Liu, and D. L. Neuhoff. On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data. In *IPSN*, 2003.

[43] N. F. Maxemchuk. Comparison of deflection and store-and-forward techniques in the Manhattan street and shuffle-exchange networks. In *INFOCOM, Eighth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, 1989.

[44] N. F. Maxemchuk. Dispersity routing in high-speed networks. *Computer Networks and ISDN Systems*, 25(6), 1993.

[45] S. McCanne and M. Vetterli. Joint source/channel coding for multicast packet video. In *International Conference on Image Processing*, pages 25–28, 1995.

[46] S. McCanne, M. Vetterli, and V. Jacobson. Low-complexity video coding for receiver-driven layered multicast. *IEEE Journal on Selected Areas in Communications*, 15(6):983 –1001, Aug. 1997.

[47] J. McQuillan, I. Richer, and E. C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transaction on Communications*, 28(5):711–719, May 1980.

[48] G. Mergen and L. Tong. Capacity of regular ad hoc networks with multipacket reception. In *Proc. Allerton conf. on Commun., control and comp.*, 2002.

[49] M. Mitzenmacher. Bounds on the greedy routing algorithm for array networks. In *Proceedings of the 6th Annual Symposium on Parallel Algorithms and Architectures*, pages 346–353, New York, NY, USA, June 1994. ACM Press.

[50] E. Modiano and A. Ephremides. Efficient algorithms for performing packet broadcasts in a mesh network. *IEEE/ICM Transactions on Networking*, 4(4):639–648, August 1996.

[51] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *MOBICOM*, 2000.

[52] J. J. Narraway. Shortest paths in regular grids. *IEEE Proc. Circuits Devices Syst.*, 145(5), Oct. 1998.

[53] M. J. Neely and C. E. Rohrs. Equivalent models and analysis for multi-stage tree networks of deterministic service time queues. In *38th Annual Allerton Conference on Communication, Control and Computing*, Oct. 2000.

[54] M. J. Neely, C. E. Rohrs, and E. Modiano. Equivalent models for analysis of deterministic service time tree networks. Technical Report P-2540, MIT - LIDS, March 2002.

[55] L. Ozarow. On a source-coding problem with two channels and three receivers. *The Bell System Technical Journal*, 59(10), Dec. 1980.

[56] V. Th. Paschos. Polynomial approximation and graph-coloring. *COMPUTG: Computing (Archive for Informatics and Numerical Computation)*, 70, 2003.

[57] K. Pawlikowski. Steady-state simulation of queueing processes: A survey of problems and solutions. *CSURV: Computing Surveys*, 22, 1990.

[58] M. R. Pearlman and Z. J. Haas. Determining the optimal configuration for the zone routing protocol. *IEEE J. Select. Areas Commun.*, 17(8):1395–1414, 1999.

[59] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *2nd IEEE Workshop Mobile Comp. Syst. Applic.*, 1999.

[60] J. Polastre, J. Hill, and D. Culler. Versatile low-power media access for wireless sensor networks. In *Proceedings of the ACM Sensys Conference*, Los Angeles, CA, 2003.

[61] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5), May 2000.

[62] S. S. Pradhan, R. Puri, and K. Ramchandran. (n,k) source-channel erasure codes: Can parity bits also refine quality? In *Proc. Conf. Inform. Sciences and Syst.*, 2001.

[63] R. Puri, T. Kim, and K. Ramchandran. Multiple description source coding using forward error correction. In *33rd Asilomar Conference on Signals, Systems and Computer*, 1999.

[64] R. Puri, S. S. Pradhan, and K. Ramchandran. n-channel symmetric multiple descriptions: New rate regions. In *Proc. Int. Symposium Inform. Theory*, 2002.

[65] Y. E. Sagduyu and A. Ephremides. The problem of medium access control in wireless sensor networks. *IEEE Wireless Communications*, 11(6):44–53, Dec. 2004.

[66] Crossbow Products: Wireless sensor networks.

[67] S. D. Servetto and G. Barrenetxea. Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks. In *Proc 1st ACM Int. Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sept. 2002.

[68] J. F. Sibeyn. Overview of mesh results. Technical Report MPI-I-95-1-018, Max-Planck-Institut für Informatik, Saarbrücken, 1995.

[69] J. Sun and E. Modiano. Capacity provisioning and failure recovery for low earth orbit satellite constellation. *International journal of satellite communications and networking*, 21:259–284, June 2003.

[70] J. Sun and E. Modiano. Routing strategies for maximizing throughput in leo satellite networks. *IEEE journal on selected areas in communications*, 22(2), Feb. 2004.

[71] J. H. Upadhyay, V. Varavithya, and P. Mohapatra. Efficient and balanced adaptive routing in two-dimensional meshes. In *HiPC96, Proceeding of High Performance Computing*, pages 112–121, 1996.

[72] V. A. Vaishampayan. Design of multiple description scalar quantizers. *IEEE Transactions On Information Theory*, 39(3), May 1993.

[73] E. Varvarigos and D. P. Bertsekas. Performance of hypercube routing schemes with or without buffering. *IEEE/ICM Transactions on Networking*, 2(9):299–311, June 1994.

[74] H. S. Witsenhausen. An achievable region for the breakdown degradation problem with multiple channels. In *Bell Laboratories Technical Memorandum, TM-81-112117-3*, Jan. 1981.

[75] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *MOBICOM*, 2001.

Curriculum Vitae

Guillermo BARRENETXEA
Av. de Tivoli, 2D
1007 Lausanne
Switzerland

URL: http://lcavwww.epfl.ch/∼gbarrene
e-mail: guillermo.barrenetxea@epfl.ch
Date of Birth : October 26, 1976
Nationality : Spanish

Education

- *Ph.D. Candidate* (since 2001),
  Audio-Visual Communications Laboratory, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
- *Doctoral School in Communication Systems* (2000-2001),
  School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
- *Master Thesis* (1999-2000),
  Corporate Communications Department, Institut EURECOM, Sophia-Antipolis, France.
- *B.S. and M.S. in Telecommunication Engineering* (1994-1999),
  Universidad Publica de Navarra (UPNA), Pamplona, Spain.

Professional Experience

- *Research and Teaching Assistant* (since 2002),
  Audio-Visual Communications Laboratory, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
- *Summer Internship* (Summer 2000),
  BT-Exact department, British Telecom, Ipswich, UK.
- *Research Assistant* (1999),
  Corporate Communications Department, Institut EURECOM, Sophia-Antipolis, France.
- *Summer Internship* (Summer 1999),
  Electrical Engineering Department ,University of Navarra, Pamplona, Spain.

Awards and Distinctions

- *Award for academic excellency* (2000),
  University of Navarra, Spain. Best telecommunication engineering (first place graduation award).
- *Graduate Fellowship* (1999),
  Institut EURECOM, Sophia-Antipolis, France.
- *Reseach Fellowship* (1998),
  Spanish Ministry of Education and Culture Fellowship.

Publications

Journals

1. G. Barrenetxea, B. Beferull-Lozano and M. Vetterli,*Lattice Networks: Capacity Limits, Optimal Routing and Queueing Behavior*, Accepted to IEEE/ACM Transactions on Networking.

Conferences

1. E. Baccaglini, G. Barrenetxea and B. Beferull-Lozano, *Interaction between multiple description coding and sensor networks with finite buffers*, IEEE International Conference on Multimedia & Expo (ICME), Amsterdam, The Netherlands, July 2005.
2. G. Barrenetxea, B. Beferull-Lozano and M. Vetterli, *Efficient routing with small buffers in dense networks*, International Conference on Information Processing in Sensor Networks (IPSN), Los Angeles, CA, April 2005.
3. G. Barrenechea, B. Beferull-Lozano and M. Vetterli, *Lattice Sensor Networks: Capacity Limits, Optimal Routing and Robustness to Failures*, International Conference on Information Processing in Sensor Networks (IPSN), Berkeley, CA, April 2004.
4. G. Barrenechea, B. Beferull-Lozano, A. Verma, P. L. Dragotti and M. Vetterli, *Multiple Description Source Coding and Diversity Routing: A Joint Source Channel Coding Approach to Real-Time Services over Dense Networks*, Proc. 13th International Packet Video Workshop, Nantes, France, April 2003.
5. S. D. Servetto and G. Barrenechea, *Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks*, Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), Atlanta, GA, USA, September 2002.