Reliable and time-constrained communication in wireless sensor networks

Fei Yang

Numéro d'ordre: 2011-ISAL-0005 Year 2011

THESIS

RELIABLE AND TIME-CONSTRAINED COMMUNICATION IN WIRELESS SENSOR NETWORKS

defended at
L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

for the degree of
DOCTOR OF PHILOSOPHY

Ecole doctorale : INFORMATIQUE ET MATHÉMATIQUES

submitted on 16 November 2010

by
Fei YANG

Defended on 25 January 2011 in front of the following jury

Ye-Qiong SONG	Professeur	Rapporteur	LORIA-TRIO, France
André-Luc BEYLOT	Professeur	Rapporteur	IRIT/ENSEEIHT, France
Jean-Marc THIRIET	Professeur	Examinateur	Université Joseph Fourier, France
Ji-Ming CHEN	Docteur	Examinateur	Zhejiang University, China
Tanguy RISSET	Professeur	Directeur	CITI, INSA-LYON, France
Isabelle AUGE-BLUM	Docteur	Co-encadrante	CITI, INSA-LYON, France

This thesis was prepared at Centre d'Innovation en Télécommunications et Intégration de Services (CITI), INSA de Lyon - INRIA Rhône-Alpes

Acknowledgments

First of all, I would like to thank Doctor Isabelle Augé-Blum and Professor Tanguy Risset, for providing me the opportunity to work on the fantastic topic and guiding me throughout my Ph.D. study. I would like to give special thanks to Isabelle Augé-Blum who taught me how to research in the viewpoint of academia.

I thank Professor Ye-Qiong SONG and Professor André-Luc BEYLOT for having accepted to review my thesis. I thank Professor Jean-Marc THIRIET and Doctor Ji-Ming CHEN for having accepted to be the examiners of my thesis defence.

I would also like to thank my colleagues at CITI laboratory for kindly sharing the experience on their study and work. Professor Bo LI introduced me a lot of his research experiences during his visit to CITI. Thomas WATTEYNE and Elyes BEN HAMIDA helped me to choose and use the simulator. The discussion with Ruifeng ZHANG on the topics of wireless communication and networking helped me a lot. I would like to thank Quentin and Alexandre for giving me many good comments on my manuscript. I would also like to thank Ibrahim, Ahmad, Riadh, Anis, Guowei, Lusheng, Meiling, Zheng, Yuanyuan, Yufang, Fuda and other colleagues in CITI, with whom I had a very happy time.

The last but not the least, I would like to thank my family for their unselfishness supports for the many years of my study.

Abstract

Wireless Sensor Networks (WSNs) are composed of a large number of battery-powered sensor nodes that have the ability to sense the physical environment, compute the obtained information and communicate using the radio interfaces. Because sensor nodes are generally deployed on a large and wild area, they are powered by embedded battery. And it is difficult to change or recharge the battery, thus to reduce the energy consumption when sensors and protocols are designed is very important and can extend the lifetime of WSNs. So sensor nodes transmit packets with a lower transmission power (e.g. 0dBm). With this transmission power, a packet can only be transmitted dozens of meters away. Therefore, when a sensor detects an event, a packet is sent in a multi-hop, ad-hoc manner (without fixed infrastructure and each sensor is able to relay the packet) to the sink (specific node which gathers information and reacts to the network situation).

Generally, WSNs have the following characteristics: (1) Wireless links among low power radios are highly unreliable. (2) The scheme that lets sensor nodes go to sleep to save energy incurs an additional communication delay. (3) Many applications have real-time constraints, which means the sink has to be informed before a deadline (known and bounded) after an event occurs. (4) The topology in WSNs is dynamic, e.g. some nodes may die because they run out of energy or some nodes may be added. This thesis contributes toward the design of reliable and real-time constrained protocols for WSNs.

First, we classify the main existing MAC (Medium Access Control) and routing protocols for WSNs in the literature. Most of them are energy-efficient. Few of them consider unreliable links and very few of them take the real-time constraints into account. To the best of our knowledge, no one considers those three points together. The aim of this thesis is to design this kind of protocols.

Through the study on the state-of-the-art protocols, we underline three key points to design reliable and real-time constrained protocols for WSNs, namely virtual coordinate, duty-cycle and unreliable links. The first work presented in this thesis is the study on the relationships between the three key points, which provides some guidelines to design the protocols. At first, we study the impacts of duty-cycle on the performances of routing protocols. We find that it is better to have cross-layer (MAC/routing) protocols for duty-cycle WSNs. Then, we study the impacts of unreliable links on the performances of routing protocols. The conclusion of this

study is that the virtual coordinate based routing protocols have good performances in terms of delivery ratio if the unreliable links are taken into consideration when constructing the virtual coordinates. Based on this study result, we propose three new simple and effective methods to construct virtual coordinates under unreliable links in WSNs. We find, through simulation results, that the proposed schemes can have better performances than the existing method that does not take unreliable links into consideration without introducing any additional cost.

By considering real-time constraints and the study result that it is better to have cross-layer protocols for duty-cycle WSNs, we propose a novel forwarding scheme (WSEDR) based on a distributed wakeup scheduling algorithm. This protocol can guarantee bounded delay on the messages that are delivered, and can have higher delivery ratios for ultra-low duty-cycle WSNs under unreliable links. The sensor nodes in WSEDR are considered to be locally synchronized. This is possible but expensive in the number of exchanged messages and energy. WSEDR schedules the wakeup time of each node according to two parameters: the hop count between the sensor and the sink and expected delivery ratio to the sink depending on the link probability. We model the forwarding scheme and analyze its properties by mathematics and simulations. Simulation results are in line with the mathematical results and show that WSEDR has good performances in terms of delivery ratio, end-to-end delay and energy efficiency.

WSEDR can have good performances for duty-cycle WSNs, however, it is difficult or expensive to synchronize sensor nodes in large-scale WSNs. Therefore, asynchronous protocols are needed for some large-scale WSNs. To this end, we propose a Real-Time Constrained Forwarding (RTCF) protocol for ultra-low duty-cycle WSNs. RTCF is an asynchronous version of WSEDR, and adopts the preamble sampling technique to communicate and dynamically adjusts the number of potential relay nodes according to the real-time constraints of the packets to increase the delivery ratio. Simulation results show that RTCF has a high delivery ratio and the sent packets can reach the deadline.

For WSNs, the topology is dynamic because some sensor nodes may die due to energy depletion or some new nodes may be deployed, therefore, to be robust is a very important requirement for those WSNs. The previous proposed protocols have very good performances, but can not autonomously update with topology changes. At last, we propose a robust forwarding (RF) protocol which is able to adapt autonomously its parameters according to the topology changes. The simulation results show that RF has very good performances in terms of delivery ratio and robustness when the topology changes dramatically.

This thesis proposes and evaluates energy-efficient, cross-layer and virtual coordinate based protocols for reliable and real-time communications in WSNs under unreliable links and topology changes. To the best of our knowledge, no existing protocol today gives this solution in that context.

Résumé

Les réseaux de capteurs sans fils (WSN) sont composés d'un très grand nombre de composants électroniques (les capteurs), capables de mesurer des paramètres physiques de l'environnement, de mettre en forme l'information obtenue et de la communiquer aux autres capteurs grâce à une interface radio. Les capteurs étant en général déployés sur de très grandes étendues géographiques, l'énergie nécessaire pour les faire fonctionner est fournie par une batterie embarquée sur le capteur. En règle générale, il est difficile de recharger les batteries une fois les capteurs déployés. Economiser l'énergie est donc une préoccupation constante lors de la conception des capteurs et des protocoles de communication utilisés, de manière à prolonger la durée de vie du réseau. Dans ce but, les capteurs transmettent leurs données avec des puissance d'émission très faibles (par exemple, 0 dBm). Avec de telles puissances d'émission, un message ne peut être transmis que sur quelques dizaines de mètres. De ce fait, lorsqu'un capteur détecte un événement, le message est transmis en mode ad-hoc multisauts (c'est à dire sans infrastructure fixe, chaque capteur étant capable de relayer le message) jusqu'au puits, un noeud spécifique du réseau, qui récolte toutes les informations et est capable de réagir de manière adéquate.

En régle générale, les WSN ont les carastéristiques suivantes: (1) Les liens sans fils sont très peu fiables, surtout lorsque les puissances d'émission sont faibles. (2) Pour économiser l'énergie, les capteurs sont endormis la plupart du temps (duty-cycle). Mais le fait d'endormir les capteurs implique des délais importants dans les communications. (3) Or, de nombreuses applications ont des contraintes temps-réel, c'est à dire que le puits doit être informé avant un temps connu et borné qu'un événement s'est produit. (4) La topologie du WSN est dynamique, car, en plus des liens non fiables, des capteurs peuvent disparaitre car ils sont à cours d'énergie, ou au contraire des capteurs peuvent être rajoutés. La contribution de cette thèse est de proposer de nouveaux protocoles, fiables et temps-réel, pour WSN.

Dans un premier temps, nous proposons une classification des principaux protocoles de niveau MAC (Medium Access Control) et de routage existants pour les WSN. Si la plupart tiennent compte des contraintes énergétiques, peu sont concus pour fonctionner avec des liens radio non fiables. Et très peu de propositions permettent des communications temps-réel. A notre connaissance, aucun protocole ne permet à la fois des communications temps-réel, en tenant compte des contraintes énergétiques et de non fiabilité des liens radio. C'est sur ce

manque que cette thèse se propose d'apporter des contributions.

A partir de l'étude de l'état de l'art, nous avons dégagé trois caractéristiques qui nous semblent nécessaires pour concevoir des protocoles fiables et temps-réel, à savoir l'utilisation de coordonnées virtuelles pour le routage, l'utilisation de cycles d'endormissement des capteurs (duty-cycle) pour économiser l'énergie et la prise en compte des liens radio non fiables dès la conception des protocoles. Le premier travail présenté dans cette thèse est l'étude des liens existants entre ces trois paramètres clé pour la conception de protocoles. Tout d'abord nous étudions l'impact du cycle d'endormissement sur les performances des protocoles de routage. Cela nous permet de montrer qu'il est préférable d'utiliser des protocoles cross-layer (MAC-routage) dans un soucis d'économie d'énergie. Ensuite, nous étudions l'impact des liens non fiables sur les performances des protocoles de routage. Ces études permettent de conclure que le routage par coordonnées virtuelles permet d'avoir de bonnes performances en terme de taux de livraison des messages, lorsque les liens non fiables ont été pris en compte dans la construction des coordonnées virtuelles. Dans la continuité de ce résultat, nous avons proposé trois nouvelles méthodes de construction des coordonnées virtuelles, tenant compte des liens non fiables, pour WSN. Par simulation, nous montrons que ces coordonnées ont de meilleures performances que celles existantes ne tenant pas compte des caractéristiques de liens, sans introduire de coûts additionnels.

En considérant les contraintes temps-réel et le fait qu'il est préférable d'avoir des protocoles cross-layer pour les WSN, nous avons proposé un nouveau protocole, WSEDR (Wakeup Scheduling according to Expected Delivery Ratio), basé sur un algorithme localisé de ordonnancement des instants de réveil des capteurs dans le duty-cycle. Ce protocole garantit le respect d'un délai connu et borné pour les messages qui sont délivrés au puits. De plus, ce protocole permet un taux de délivaison élevé pour les WSN avec économie d'énergie et liens radio non fiables. WSEDR nécessite que les capteurs soient synchronisés, au moins localement (ce qui signifie qu'ils ont une référence temporelle commune). Cela est possible mais a un coût en terme de messages échangés et donc en terme d'énergie. WSEDR ordonnance les instants de réveil de chaque capteur en fonction de deux paramètres: le nombre de sauts séparant le capteur du puits, et le taux de délivraison attendu, dépendant des caractéristiques des liens radio utilisés pour atteindre le puits. Nous avons modélisé ce protocole et analysé ses propriétés mathématiquement et par simulation. Les résultats des simulations sont en accord avec ceux obtenus mathématiquement, et montrent que WSEDR a de bonnes performances en termes de taux de livraison des messages, de délai de bout en bout et d'économie d'énergie.

Bien que WSEDR ait de bonnes performances, il nécessite que les capteurs soient synchronisés, ce qui entraine des coûts et une complexité accrue lorsqu'on s'intéresse à des WSN avec de très nombreux capteurs. Les protocoles ne nécessitant pas de synchronisation sont donc mieux adaptés aux réseaux à grande échelle. C'est pourquoi nous avons proposé une version

asynchrone de WSEDR, RTCF (Real-Time Constrained Forwarding). RTCF utilise des techniques d'échantillonnage de préambule pour communiquer et adapte dynamiquement le nombre de capteurs succeptibles de relayer le message en fonction des contraintes temporelles du message. Cela permet d'augmenter le taux de livraison des messages, en considérant les contraintes temporelles associées aux messages. Les résultats de simulation montrent que RTCF a un taux important de livraison et permet de respecter les deadlines d'une proportion importante des messages émis.

A cause des capteurs qui meurent lorsqu'ils sont à court d'énergie et des capteurs déployés durant le fonctionnement des WSN, la topologie de ces réseaux est dynamique, et le réseau doit être capable de s'adapter aux changements. La robustesse aux changements de topologie est donc une propriété importante pour les WSN. Les protocoles proposés précédemment ont de bonnes performances, mais celles-ci se dégradent lorsqu'il y a des changements topologiques dans le réseau. Nous avons donc proposé RF (Robust Forwarding), capable de mettre à jour de manière autonome ses paramètres pour s'adapter aux changements. Les résultats de simulation montrent que RF a de bonnes performances en termes de taux de livraison et de robustesse, malgré de fortes perturbations de la topologie du réseau.

Cette thèse propose donc et évalue des protocoles économes en énergie et basés sur des coordonnées virtuelles permettant des communications fiables et temps-réel dans les WSN, malgré des liens non fiables et d'importants changements dans la topologie du réseau. A notre connaissance, aucun autre protocole à ce jour n'apporte de solution dans ce contexte.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of Wireless Sensor Networks

With the advancement of Micro-Electro-Mechanical Systems (MEMS) technology [Gad-el Hak, 2002], wireless communications and digital electronics, Wireless Sensor Networks (WSNs) are becoming more and more realizable and applicable to many areas [Akyildiz et al., 2002].

WSNs are composed of a large number of battery-powered sensor nodes that have the ability to sense the physical environment, compute the obtained information and communicate using the radio interfaces. A specific node, called sink, is in charge of collecting and processing the information. Because sensor nodes are generally deployed on a large and wild area, they are powered by battery. And it is difficult to change or recharge the battery, thus to reduce the energy consumption and to extend the lifetime of WSNs are very important points. In order to extend the lifetime, sensor nodes transmit packets with a lower transmission power (for example, 0dBm). With this transmission power, a packet can only be transmitted dozens of meters away in one hop. Therefore, a packet from the source node in WSNs will be sent to the sink node in a multi-hop, ad-hoc manner (every node could be the relay node and there is no infrastructure in the network).

Due to the characteristics of low cost and easy to deploy, WSNs have attracted more and more attention and have been used in many different applications [Akyildiz et al., 2002]. In the following subsections, we will first present some typical WSNs applications. Then, the characteristics of WSNs and the challenges of WSNs will be discussed.

1.1.1 Applications of WSNs

WSNs have been applied to many different areas. The applications of WSNs can be generally classified into the following three categories: (1) target detection and tracking, (2) monitoring and (3) assisted-living. Figure 1.1 shows the classification of WSNs applications. Some related survey papers about WSNs applications can be found in [Kuorilehto et al., 2005; Xu, 2002]. In the following subsections, we will describe those categories in detail.

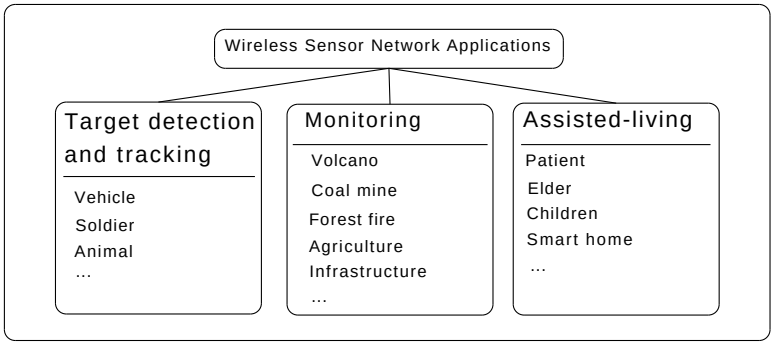


Figure 1.1: Classification of wireless sensor networks applications

Target detection and tracking

The goal of this kind of applications is to follow the location of a target in an environment, for example, a soldier in a battlefield or a lion in a savannah. In those applications, a large number of sensor nodes are deployed on a large area. Those sensor nodes have different sensors which can measure different parameters, for example, magnetic, acoustic and so on. With those measured parameters, a target can be detected. If a target moves into the area monitored by the WSN, it will be detected by a few sensor nodes. Those sensor nodes will collaborate with each other to verify and classify the target. Then, the detected information will be reported to a base station. After collecting the information, the base station can track the target. The characteristics of target detection and tracking applications are that the target is moving and the sensor nodes are usually fixed.

We want to introduce some typical applications by focusing on how the detected information be sent to the base station. A line in the sand [Arora et al., 2004] presents the results of an experimentation consisting of 90 sensor nodes. The root node in the network periodically sends packets to other nodes in order to construct a routing tree. After detecting a target, the corresponding messages will be sent to the root node via the constructed routing tree. DCTC [Zhang and Cao, 2004] proposes a different way to route the messages. One of the nodes which has detected the target is chosen as the reporter. After aggregating, the chosen reporter will send the target's information to the base stations. As the target is moving, some nodes are becoming far from the target and some other nodes are becoming near the target. Then a new reporter is chosen. In order to bound the delay of detecting a target, VigilNet [He et al., 2006b] proposes to let the aggregation node send the aggregated message to the nearby base station in one hop.

Monitoring

The goal of monitoring applications is to detect an abnormal event (e.g. forest fire detection [Doolin and Sitar, 2005]) or to collect the status information of a monitored object (e.g. railway [Chebrolu et al., 2008] and volcano [Song et al., 2009]). For the monitoring application

aiming at detecting an abnormal event, sensor nodes only send alarm packets after detecting an event. For the monitoring application aiming at collecting the status information, sensor nodes periodically sample the parameters and send the information to the control center.

In general, in forest fire detection applications, the decision of the occurrence of a fire can be done on sensor nodes. For example, if the sampled temperature is higher than a predefined threshold, an alarm packet will be sent to the control center. This method can consume less energy because less packets are sent. On the other hand, periodically sampling can also be used for forest fire detection. In this method, every node periodically samples the related parameters (e.g. temperature, humidity and pressure) and sends them to the control center where the decision is taken [Hefeeda and Bagheri, 2008; Kosucu et al., 2009; Son et al., 2006].

There are some other applications which aim at collecting the status information of an object. [Mainwaring et al., 2002] deploys a WSN consisting of 32 nodes on an island to monitor seabird nesting environment and behavior. [Chebrolu et al., 2008] proposes a WSN system, called BriMon, to monitor the health of railway bridges. [Kim et al., 2007] deploys 64 nodes to monitor the Golden Gate Bridge. [Tolle et al., 2005] deploys a WSN on a 70-meet tall redwood tree to monitor the microclimate surrounding it. [Ceriotti et al., 2009] deploys some sensor nodes in Torre Aquila, a medieval tower in Trento (Italy), to monitor the environmental conditions of the heritage building. 27 sensor nodes are deployed to monitor a coal mine [Li and Liu, 2009]. [Song et al., 2009] deploys a WSN to monitor the volcano.

Assisted-living

For assisted-living applications, the sensor nodes are generally installed on objects, for instance, human body or some objects. The characteristic of this kind of applications is that the network is small-scale. This kind of WSNs has been widely used to assist the doctor to know the situation of the patients [Lorincz et al., 2009]. This belongs to a new hot topic called wireless body area network (BAN) [Su and Zhang, 2009; Xiao et al., 2009]. In addition to the BAN applications, assisted-living applications also include some applications that are used to monitor the residential parameters. For example, ALARM-NET [Wood et al., 2006] integrates the BAN and the sensor network installed in the residential environment together. The former is used to collect the body's information (e.g. pulse) and the latter is used to monitor the residential parameters (e.g. temperature and dust).

For all those types of WSNs, reliability is the basic requirement. In addition to the reliability, time-constraint is another important requirement, for example, in some target detection and tracking applications, a report must be sent to the sink node within a deadline (it depends on the velocity of the target), otherwise, the report will be useless because a new report which has more recent information will be sent. Those requirements challenge the protocol design in WSNs. We will discuss the challenges of WSNs in Section 1.1.4.

1.1.2 Types of data flows in WSNs

The aforementioned three categories in the previous subsection are classified according to the background of the applications. The applications can also be classified into the following four categories according to the types of the data flows in the network [Tilak et al., 2002].

(1) Periodic. Every sensor node periodically sends packets to the destination. Most of the aforementioned monitoring applications belong to this kind of application.

(2) Event-driven. Sensor nodes only send packets to the destination when they detect an event. For example, in forest fire monitoring applications, a node will send an alarm packet after it detects an abnormal temperature. In target detection and tracking applications, nodes do not send packets until the target (e.g. a soldier) is detected.

(3) Query-driven. Sensor nodes send packets to the destination when they receive a query packet from the control center, for example, in assisted-living applications, doctor may query the data of one or some patients.

(4) Hybrid. This means that one or more of those types are included in one application. Some monitoring applications can be classified into this category, for example, in a railway bridge monitoring application [Chebrolu et al., 2008], the sensors start to sample and send packets when a train is coming. And the sampling process will last for a certain period.

1.1.3 Characteristics of WSNs

Characteristics of sensor nodes

Before describing the characteristics of sensor node, a simple architecture of sensor node will be introduced.

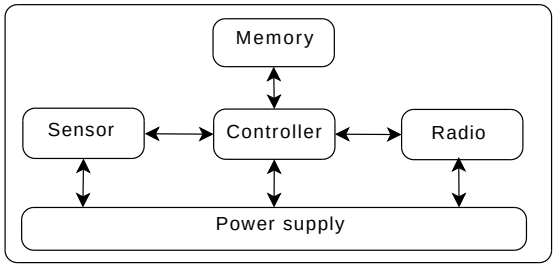


Figure 1.2: A simple architecture of sensor node

As we can see from Figure 1.2, the sensor node is composed of the following five modules: (1) sensor, (2) memory, (3) radio, (4) power supply and (5) controller. The most distinct point of a sensor node is the sensor module which is tightly related to the requirements of applications and the detection algorithms. For example, temperature, relative humidity and wind speed sensors are installed to monitor the fire behavior in [Hartung et al., 2006]. Magnetic, acoustic

and motion sensors are equipped on the sensor nodes in VigilNet [He et al., 2006a] to monitor the moving vehicles. Other modules are the general components of embedded systems which will not be explained one by one here.

In addition to the special sensors equipped on the sensor nodes, they mainly have the following characteristics when comparing with the traditional wireless devices, such as the mobile phone and the portable computer:

(1) Small in size. The size of a sensor node is very small compared with the traditional wireless ad-hoc network nodes. Sometimes a sensor node has the same size as a coin. For example, it is better to make the sensor node as small as possible in target detection and tracking applications, especially for the military applications. And because a WSN is composed of a large number of sensor nodes that cooperate with each other to accomplish a task, a single sensor node does not need strong abilities and should have a low cost. As a result, sensor nodes are designed very small.

(2) Battery-powered. WSNs are generally deployed on a wild area where the only possible source of energy is battery, for example, the sensor network deployed on the Great Duck Island which is a 237 acre island near Maine [Mainwaring et al., 2002].

(3) Limited memory and process capability. As we have discussed in the previous two parts, sensor nodes are generally designed small and should have a low cost. Therefore, the speed of the processors used by sensor nodes are not very fast and the size of the memory of the sensor nodes is very small.

(4) Low-power. Because the sensor nodes are powered by batteries and it is almost impossible to recharge or change the batteries, the radio on the sensor node should have a low power, for example, 0 dBm. The range of the transmission power of some typical sensor nodes is list in Table 1.1.

There exist some well-known sensor nodes whose detailed parameters are shown in Table 1.1. Mica2 is a type of low-power sensor nodes designed by CROSSBOW company [Mica2, 2002]. Different from Mica2, MicaZ uses a much higher frequency in order to increase the data rate [MicaZ, 2005]. Imote2, also designed by CROSSBOW, has a much stronger processor and much larger memories [Imote2, 2007]. Telos [Polastre et al., 2005] and Tmote Sky [Tmote-sky, 2006] are two types of sensor nodes developed by Moteiv corporation. Tmote Sky is a replacement of Telos.

Characteristics of WSNs

In the previous subsection, we have presented the characteristics of sensor nodes. Besides the differences of a single node, WSNs are different from other wireless networks (e.g. mesh or ad-hoc networks) in many other points. For example, mesh networks [Akyildiz et al., 2005] are static and not power-constrained. Ad-hoc networks [Kumar et al., 2006; Mauve et al., 2002]

Table 1.1: The parameters of some well-known sensor nodes

	Mica2	**MicaZ**	**Telos**	**Tmote Sky**	**Imote2**
Processor	ATmega128L	ATmega128L	TI MSP430	TI MSP430	Intel PXA271
Memory	4K RAM 128K Flash	4K RAM 128K Flash	10K RAM 48k Flash	10k RAM 48k Flash	256k SRAM 32M FLASH 32M SDRAM
Output power(dBm)	-20 to 5	-24 to 0	-25 to 0	-25 to 0	-24 to 0
Radio Frequency(MHz)	868/916	2400	2400	2400	2400
Sensor	expandable	expandable	Humidity Temperature	Humidity Temperature Light	expandable
Battery	2 AA	2 AA	2 AA 2 AAA coin cell	2 AA coin cell	3 AAA

are generally mobile and power-constrained. WSNs are always very power-constrained and large-scale [Akyildiz et al., 2002]. We will summarize some characteristics of WSNs in the following points:

(1) Application-related. WSNs are driven by applications. After being successfully applied on the Great Duck Island [Mainwaring et al., 2002], WSNs have attracted more and more attention and are widely used in many different applications, as we have presented in Chapter 1.1.1. Different applications have different requirements, which leads to different network architectures and different protocols. For example, the network architecture of the WSN deployed on a tree [Tolle et al., 2005] is very different from the network architecture of the WSN deployed in a heritage building [Ceriotti et al., 2009].

(2) Easy-to-deploy. Because sensor nodes can automatically organize into a network after being deployed, it is not necessary to deploy some key nodes (e.g. access points) in advance. So, we can see that it is very easy to deploy WSNs, for example, sensor nodes are dropped from the helicopter in some WSNs applications.

(3) Large-scale. WSNs can have hundreds or thousands of sensor nodes to monitor a large area although some existing prototype systems have only dozens or hundreds of nodes. In some target detection and tracking applications and monitoring applications, dozens of sensor nodes are deployed. For example, over 90 sensors are used in [Arora et al., 2004] and 150 sensor nodes are used in [Szewczyk et al., 2004]. In the near future, more large-scale WSNs (more than 1000 nodes) would be deployed.

(4) Dynamic topology. Due to the low-power characteristic of sensor nodes, the wireless links between two sensor nodes are highly unreliable and dynamic. Moreover, some nodes may run out of energy and some other new nodes may be added, which leads to the dynamicity of WSNs.

(5) Without human interaction. WSNs are sometimes deployed in an area without human interaction. As a result, WSNs should be self-organizing which means that WSNs do not need to configure in advance and the sensor nodes can organize into a network automatically after deploying. Another requirement is to be self-healing which indicates that

WSNs can reconfigure the network automatically after some nodes die or new nodes are added.

(6) Convergecast traffic. Most of the WSNs applications have a sink node (or multiple sink nodes) which collects the data from other sensor nodes. In wireless ad-hoc and mesh networks, traffics can be from any node to any other nodes.

1.1.4 Challenges of WSNs

In the previous subsections, we have presented some typical WSNs applications and the characteristics of WSNs. In this subsection, we will focus on the challenges of WSNs. As discussed in previous sections, WSNs have the following characteristics: (1) large-scale, (2) low power and (3) battery-powered. Because of those characteristics, it is a challenging problem to design protocols for WSNs, especially to design the protocols under the following constraints:

(1) Reliability. It is defined as the ability of a protocol to keep running without error, for example, when the link quality is very bad, whether a protocol can still successfully transmit packets to the sink node. For large-scale WSNs, to increase the reliability under unreliable links is a challenging problem. And the situation is deteriorated when the sensor nodes go to sleep in order to save energy.

(2) Robustness. We define robustness as the ability of a protocol to cope the changing topology and to survive when the topology is changed. As we have presented in the previous subsections, some WSNs are dynamic, for example, some sensor nodes die and some new sensor nodes are deployed. As a result, to ensure robust transmission under those conditions is a challenging problem.

(3) Real-time. In addition to the reliability and robust requirements, some WSNs have real-time constraints, for example, a packet should be sent to the destination within a specified deadline, otherwise, the packet may be useless. When the sensor nodes periodically go to sleep in order to save energy and the wireless links are unreliable, it is very challenging to design real-time protocols for WSNs.

(4) Security. As we know, wireless networks are much easier to suffer attacks compared to wired networks. For WSNs, because every sensor node has limited resources, e.g. memory and the capability of computation, the complicated encryption algorithms can not be directly used. As a result, to ensure secure communications is a challenging problem [Wang et al., 2006].

In this thesis, we just consider the challenges of reliability, robustness and real-time. The challenges of security can be found in [Chong and Kumar, 2003; Stankovic et al., 2003].

1.2 Thesis Contributions

In this thesis, the following contributions are made:

(1) In order to provide reliable and real-time constrained communications for WSNs under unreliable links and energy efficiency constraints, we detailedly survey the main existing MAC, routing and cross-layer forwarding protocols in WSNs. Those protocols are classified into different categories according to their own characteristics. This survey shows what the drawbacks of the state-of-the-art protocols are and gives some guidelines for this thesis.

(2) We present a study to define which solutions are better in order to have reliable and real-time constrained communication protocols. This study shows that it is necessary to consider unreliable links in routing protocol, especially for virtual coordinate based routing protocols. It also shows that it is better to have cross-layer (MAC/Routing) protocols for duty-cycle WSNs.

(3) By considering the duty-cycle and unreliable links simultaneously, a synchronous forwarding protocol for ultra-low duty-cycle WSNs with real-time constraints is proposed. The proposed protocol includes an efficient wakeup scheduling algorithm which can schedule the wakeup slot of every node according to their own information.

(4) An asynchronous forwarding protocol for ultra-low duty-cycle WSNs is proposed. This protocol does not need synchronization and is easy to be implemented. This protocol can dynamically adjust the number of candidate nodes (the potential relay nodes) according to the remaining time before a designated deadline.

(5) Dynamicity is an important characteristic in some types of WSNs, for example, link quality is changing very fast and some nodes may die due to energy exhaustion. A robust forwarding protocol for ultra-low duty-cycle WSNs is proposed.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

(1) In Chapter 2, we survey the existing MAC, routing and cross-layer forwarding protocols in WSNs and classify those protocols into different categories according to their own characteristics.

(2) In Chapter 3, We study the relationships between virtual coordinates, unreliable links and duty-cycle. Then, three methods for constructing virtual coordinates under unreliable links are presented. Simulation results show that the proposed methods have good performances in terms of delivery ratio.

(3) In Chapter 4, a synchronous and an asynchronous forwarding protocol for ultra-low duty-cycle WSNs are proposed. This chapter is divided into two parts, synchronous forwarding and asynchronous forwarding. In the synchronous forwarding part, we present the proposed synchronous forwarding protocol (WSEDR). WSEDR assumes that every node is locally synchronized to their neighbors. Every node wakes up in one slot to listen to the channel. WSEDR can schedule the wakeup slot of every node only according to their own information. And the performances of WSEDR are shown by both theoretical and simulations. Then, the asynchronous forwarding protocol (RTCF) is presented. In RTCF, preamble sampling technique is adopted to wake up the neighboring nodes. RTCF can dynamically adjust the number of candidate nodes by considering the remaining time before a designated deadline. We show the performances of RTCF by simulation and find that RTCF has a high delivery ratio and the sent packets can reach the deadline.

(4) In Chapter 5, a robust forwarding (RF) protocol for ultra-low duty-cycle WSNs is proposed. RF inherits some schemes from RTCF, for example, the preamble sampling technique used in RTCF. The first step of RF is to construct cost information for each node. This cost information will be used as a virtual coordinate to calculate the backoff sleep time when forwarding. Then, we introduce the algorithm used to calculate the backoff sleep time. An important step of RF is to update the cost information. Two methods of updating the network when a new node is deployed in the network are proposed. At the end of this chapter, the performances of RF are shown by simulations.

(5) In Chapter 6, the conclusions of this thesis and some future research directions are given.

Chapter 2

Background and Motivation

In Chapter 1, we have given a general overview of WSNs, including the characteristics and the typical applications of WSNs. The following characteristics are most important and result in different protocols needed by WSNs:

(1) battery-powered and difficulty to recharge the battery: this leads to the fact that all the protocols designed for WSNs should be as energy-efficient as possible.

(2) low power transmission: because sensor nodes are powered by battery, the transmission power of a sensor node can not be high. Therefore, the transmission distance of one hop will be very short (generally, the average transmission distance is about 20m when the transmission power is 0dBm [Zuniga and Krishnamachari, 2004]). And the hop count from a source node to a destination will be increased. Moreover, the link quality of low power transmission is much more unreliable, which is proved by some experimental studies [Woo et al., 2003; Zhao and Govindan, 2003].

In this chapter, we will first discuss the two basic considerations induced by the previous two points. Due to those two considerations, the protocols designed for WSNs are very different from the protocols in traditional ad-hoc networks. Then, we will give a detailed review of the routing, MAC and cross-layer protocols in WSNs. At last, the motivation of the thesis will be presented.

2.1 Basic considerations of WSNs

2.1.1 Energy efficiency

Due to the fact that sensor nodes are battery-powered and it is difficult to recharge the battery, to be energy efficiency is an very important objective for all the protocols designed for WSNs.

Let us first show the energy consumption of each state of a sensor node by taking MicaZ [MicaZ, 2005] for example. As it is shown in Table 2.1, radio dominates the energy consumption. And the energy consumption of transmitting, receiving, listening and sleeping is very

Table 2.1: Typical values of a MicaZ node

	Current Draw
Processor active mode	8 mA
Processor sleep mode	< 15 μA
Radio transmit at 0 dBm	17.4 mA
Radio receive	19.7 mA
Radio listen	19.7 mA
Radio sleep	1 μA

different. The current draw of the radio in sleep state is only 1 μA while the current draw of listen state is 19.7 mA. It is obvious that turning off the radio when it is not needed can save the energy and prolong the lifetime of WSNs to a great extent [El-Hoiydi and Decotignie., 2004; Polastre et al., 2004; Van Dam and Langendoen, 2003; Ye et al., 2002].

The protocols that let sensor nodes sleep for a certain period to save energy are called duty-cycle protocols. Duty-cycle is the ratio of the time period in which the sensor is on to the length of the whole cycle. The definition of the duty-cycle is shown in Figure 2.1. The length of the wakeup period is T_{awake}, and the length of the sleep period is T_{sleep}. So the duty-cycle is $\frac{T_{awake}}{T_{awake}+T_{sleep}}$. For some event-rare applications, there is no event most of time. In order to maximize the lifetime of WSNs, it is better for the protocols to have an ultra-low duty-cycle (e.g. 0.1%, that means the nodes sleep in the remaining 99.9% period).

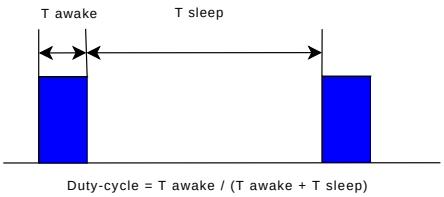


Figure 2.1: Definition of the duty-cycle

In recent years, duty-cycle is shown to be a very energy efficient solution for some WSNs, for example, duty-cycle at application [Ye et al., 2003] or MAC [Polastre et al., 2004] layer. Besides the aforementioned duty-cycle, other issues can also be considered in order to reduce the energy consumption:

(1) For routing protocols, different paths consume different amount of energy. For example, the length of one path is 5 hops while the length of another path is 8 hops. So, the paths that can consume less energy are chosen among the paths which meet other constraints.

(2) For MAC protocols, reducing the collision probability is a way of being energy efficient. If there are many collisions, one transmission might fail, so the packet has to be resent, which will consume more energy.

In addition to the aforementioned energy efficiency, another very important issue, unreliable links, must be considered for WSNs. This is because unreliable links have strong negative impacts on upper layer protocols, for example, the shortest path chosen under UDG (Unit Disk Graph) model may not be the best path under unreliable links. In the next subsection, we will show the characteristics of unreliable links.

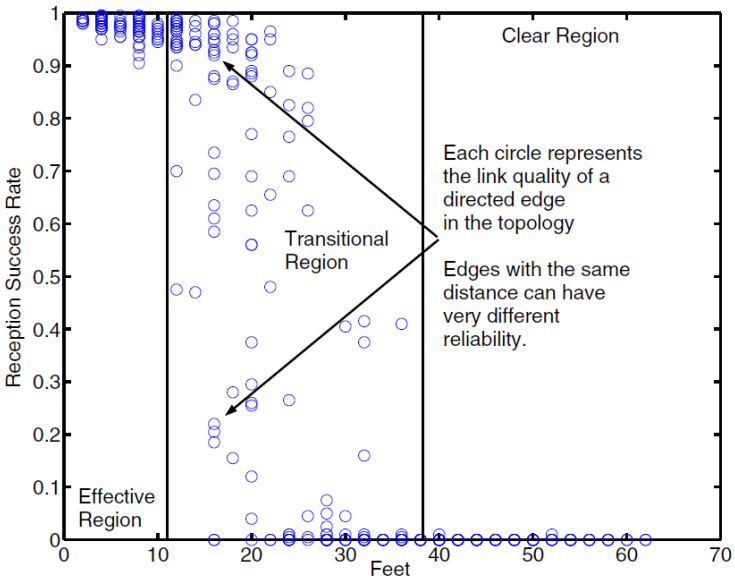


Figure 2.2: Characteristics of unreliable links (experimental results in [Woo et al., 2003])

2.1.2 Unreliable links

Several experimental studies on wireless ad-hoc and sensor networks have shown that wireless links among low-power radios are highly unreliable [Becker et al., 2009; Cerpa et al., 2005; Lin et al., 2006; Sang et al., 2007; Woo et al., 2003; Zhao and Govindan, 2003; Zhou et al., 2004]. However, UDG (Unit Disk Graph) model is used by many protocols. The basic idea of UDG model is that two nodes can always communicate with each other if they are within the transmission range of each other. They can not communicate if the distance is bigger than the transmission range. Experimental results, in contrast, show that the reachable area of a radio in WSNs is not a circle. The node outside the range used in UDG model can receive the packet sent by the sender with a certain probability while the node inside may not receive the packet with a certain probability. We can see the characteristics of unreliable links from Figure 2.2 which is drawn by [Woo et al., 2003]. In their experimentations, sensor nodes are put linearly and the distance between two neighboring nodes is 2 feet. Each node transmits 200 packets at a given power level at 8 packets/s. In order to avoid collisions and interferences, at any time, there is only one transmitter. At the end, every node can count the number of packets received from the transmitters as well as the reception success ratio as a function of distance.

In Figure 2.2, we can see that there exists a wide range of transitional region where the reception success rate varies significantly, for example, the reception success rate ranges from 0 to 1 when the distance is 20 feet (6.1m). And the transitional region is from about 10 feet (3.05m) to 40 feet (12.19m). If the distance between two nodes is bigger than 40 feet, the reception success rate is almost 0. And the reception success rate is not 100% even when the distance between two nodes is very small, e.g. 5 feet.

Compared with the aforementioned characteristics of unreliable links, the UDG model which was adopted by most of WSNs protocols is not realistic. As a consequence, the protocols proposed based on UDG model should be redesigned or improved [Stojmenovic et al., 2005].

Because of the aforementioned two basic considerations, i.e. duty-cycle and unreliable link, the protocols for WSNs should be specially designed. For example, how to increase the throughput of duty-cycle MAC protocols? How to choose a better next hop node for a routing protocol with duty-cycle nodes? In a word, duty-cycle affects the design of both routing and MAC protocols. Moreover, under unreliable links, a next hop node chosen with UDG model may not be the best choice because the link quality between this node and the sender is not good. So, routing protocol must take the link quality into consideration. Another problem caused by unreliable links is that a node with a smaller distance to the sender may not receive the packet sending by the sender while a node with a bigger distance can receive the packet. Some opportunistic protocols have been proposed to cater this problem [Sanchez et al., 2007; Shah et al., 2005]. In summary, it is better to consider the duty-cycle and unreliable links when we design the protocols for WSNs.

In the following subsections, we will give a detailed review about the MAC, routing and cross-layer protocols by the following five categories.

(1) General MAC protocols. This category includes the MAC protocols which focus on energy efficiency or throughput.

(2) General routing protocols. This category represents the routing protocols which aim to find the shortest path in order to reduce the energy consumption.

(3) Cross-layer protocols. This part reviews the protocols which integrate more than one layer protocols. We will focus on the routing/MAC cross-layer protocol in this thesis.

(4) Real-Time protocols. In this part, we want to present some real-time MAC and routing protocols.

(5) Robust protocols. We will review some protocols specially designed to be self-organizing and self-healing when some nodes are dead or new added.

2.2 General MAC protocols

As discussed in Chapter 2.1, energy efficiency is a basic consideration for WSNs and duty-cycle is the most energy efficient solution for WSNs. Therefore, more attention will be paid to duty-cycle MAC protocols. Before describing the duty-cycle MAC protocols, we want to first present some typical MAC protocols which do not take duty-cycle into consideration. Then, we will give a detailed review of the duty-cycle MAC protocols.

In wireless networks, MAC protocols do not consider the duty-cycle problem because they have enough energy and it is easy for them to recharge the battery compared to sensor networks.

Their objectives are to reduce collision probability and increase throughput, such as CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) in wireless LAN. In order to reduce the impacts of the hidden terminal and exposed terminal problems, some RTS (Request To Send) and CTS (Clear To Send) based protocols are proposed, e.g. IEEE 802.11 DCF [IEEE-Standard, 1999].

As we have discussed in Chapter 1.1.1, different applications have different types of data flows. For event-driven WSNs, a few sensor nodes might detect an event at the same time when an event happens. This phenomenon will lead to congestion within the event area if the aforementioned CSMA based protocols are used.

By considering the burst characteristic of event-driven WSNs, which means many nodes may be ready to send the packets at the same time, a CSMA based protocol with an optimal non-persistent probability distribution that nodes use to select the contention slots is proposed in [Tay et al., 2004]. The key difference between this protocol and other CSMA based protocols is that the contending nodes in this protocol choose a slot within the contention window with a specially designed probability while the contending nodes in other protocols choose the slot with uniform probability. And the contention window in this protocol is fixed. In SIFT [Jamieson et al., 2006], the authors propose a MAC protocol which adopts the optimal probability distribution proposed in [Tay et al., 2004] to distribute the burst transmission for event-driven WSNs. Their simulation results show that SIFT has a much lower delay to transmit certain number of reports.

Some researchers find that the sensed data within an event area is largely correlated due to the high density of WSNs. Therefore, it is not necessary to transmit all the nearby sensed data to the sink node. By using this characteristic of event-driven WSNs, some correlation-based MAC protocols have been proposed.

CC-MAC (Correlation-based Collaborative MAC) [Vuran and Akyildiz, 2006] is a MAC protocol which considers the burst transmission and correlation features of event-driven WSNs. CC-MAC contains two components, E-MAC (Event MAC) and N-MAC (Network MAC). Based on the spatial correlation characteristics, the sink node can calculate the correlation radius by using the INS (Iterative Node Selection) algorithm. Then the correlation radius is broadcast to the whole network. When an event happens, the sensors that detect the event will use E-MAC to send the event reports. And E-MAC can select the representative nodes which will ultimately send the report by using the correlation radius. The node which received a report packet from the other node will relay the event report using the N-MAC. And when a node has a packet to relay and has an original packet to transmit, the packet to be relayed will be given priority.

The aforementioned MAC protocols can be efficiently used in the event-driven WSNs which have burst traffics, but they are not energy efficient enough because every node wakes up all the time and idle listening consumes almost the same amount of energy as transmitting. In contrast, duty-cycle MAC protocols let sensor nodes sleep most of the time. As a result,

they are much more energy efficient. In the next subsections, we will detailedly review some duty-cycle MAC protocols.

2.2.1 Duty-cycle MAC

In Duty-cycle MAC protocols, every node sleeps most of the time. So they can wake up at a certain time by considering the wakeup time of their neighbors' (synchronous duty-cycle) or they can select their own wakeup time without knowing other nodes' information (asynchronous duty-cycle).

(1) Synchronous duty-cycle. Every node is synchronized to their neighbors. For example, all the neighboring nodes are synchronized to their neighbors and wake up at the same slot in S-MAC [Ye et al., 2002]. Scheduled duty-cycle MAC protocols can further be classified into the following two types: 1) Contention-free schedule. The wakeup slots of the neighboring nodes are scheduled without collision, e.g. DPS [Chen and Fleury, 2007]. 2) Contention-based schedule. Although the wakeup slots of the neighboring nodes are scheduled, contention still exists, e.g. S-MAC [Ye et al., 2002], DMAC [Lu et al., 2004] and DESS [Lu et al., 2005].

(2) Asynchronous duty-cycle. Every node does not need to synchronize to their neighbors and chooses their own wakeup slot without knowing the wakeup slots of their neighbors. Asynchronous duty-cycle can further be classified into: 1) Two radio based. One radio is used for paging and the other radio for transmitting the data packets [Schurgers et al., 2002; Yang and Vaidya, 2004]. 2) Single radio based. Only one radio is adopted. Preamble sampling technique is used to wake up the neighboring nodes [Buettner et al., 2006; Polastre et al., 2004].

Table 2.2: Classification of duty-cycle MAC

Duty-cycle	**Synchronous**	Contention-free	TRAMA [Rajendran et al., 2003] DPS [Chen and Fleury, 2007] LEMMA [Macedo et al., 2009]
		Contention-based	SMAC [Ye et al., 2002] DMAC [Lu et al., 2004] DESS [Lu et al., 2005]
	Asynchronous	Two-radio based	STEM [Schurgers et al., 2002] PTW [Yang and Vaidya, 2004] LEEM [Dhanaraj et al., 2005]
		Single-radio based	BMAC [Polastre et al., 2004] X-MAC [Buettner et al., 2006] RI-MAC [Sun et al., 2008]

The aforementioned categories are shown in Table 2.2. Synchronous duty-cycle protocols can have a lower delay while asynchronous duty-cycle protocols have a much higher delay. This is because asynchronous duty-cycle protocols need to wake up the candidates and synchronous duty-cycle protocols know the exact wakeup time of the candidates. However, it

is difficult or expensive to synchronize the sensor nodes in large-scale WSNs. In contrast, asynchronous duty-cycle protocols are much easier to be implemented. In the following subsections, we will introduce those duty-cycle protocols detailedly.

Synchronous

——Contention-free schedule

In order to be contention-free in duty-cycle MAC protocols, a good scheduling algorithm is needed. The general idea to be contention-free is to synchronize the neighboring nodes and schedule the transmission time of the neighboring nodes in order to avoid collisions. We can see the general idea from Figure 2.3. Here, four nodes, A, B, C and D want to transmit packets to the Sink node. A simple schedule can be as follow: node A and B transmit the packets in slot 1 and 2 respectively. Node C and D send the packets both in slot 3 because they do not have collision and interference. And node A and B sleep in slot 2 and 1 respectively. Both C and D sleep in slot 1 and 2.

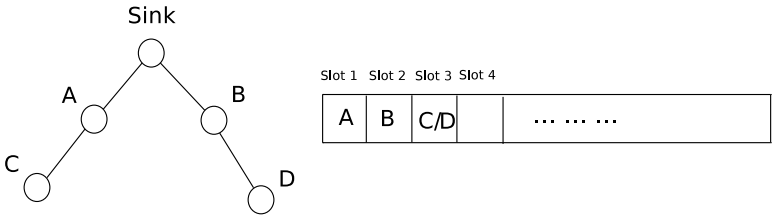


Figure 2.3: A simple example of contention-free schedule

The scheduling algorithms of duty-cycle MAC protocols are not always distributed, for example, TSMP [Pister and Doherty, 2008]. For contention-free duty-cycle MAC protocols, it is important and necessary to construct a schedule for each node distributedly. DPS [Chen and Fleury, 2007] proposes a distributed duty-cycle scheduling algorithm which integrates collision-avoidance and duty-cycling simultaneously by scheduling the transmission and reception slots of each node using graph coloring theory. Nevertheless, it is better for the scheduling algorithm to take the traffic into consideration. For example, if every node periodically sends a fixed length of packets, the schedule constructed after deployment can be used for a long time. However, if the traffic at each node is changing, how to efficiently schedule the transmission slots for each node?

TRAMA [Rajendran et al., 2003] proposes a traffic-adaptive, collision-free MAC protocol for WSNs. It schedules the transmission and reception slots of each node to avoid collisions and interferences according to the node's traffic information and their two hop neighbors' schedule information. When some nodes are assigned to neither transmit nor receive, they will switch to sleep state to save energy. The difference between this scheduling algorithm and the one proposed in DPS is that DPS aims to have a much lower diameter (the maximum one among the shortest delays from any node to any other others) while TRAMA just consider the one hop

delay.

As we have discussed in Chapter 2.1.2, UDG is not realistic in low-power WSNs. However, the previous presented two scheduling algorithms are based on UDG model. And they adopt 2-hop interference model to schedule the sensor nodes. 2-hop interference model means the nodes can not interfere each other if the distance between them is greater than 2 hop. Different from the 2-hop interference model used in those protocols, LEMMA [Macedo et al., 2009] adopts a more realistic interference model to schedule the transmission slots. For example, two nodes that are 5 hops away from each other can have interferences while the nodes that are 2 hops away may not have interferences. Based on this realistic interference model, all the nodes are scheduled to transmit in a certain slot and receive in the slots in which their child nodes transmit. In order to save the energy, all the nodes go to sleep when they do not have to transmit and receive.

——Contention-based schedule

Contention-free duty-cycle protocols can avoid the collisions and they can have a higher throughput when the traffic is high. However, it is expensive to achieve a network-wide collision-free schedule. Moreover, in the situation where the traffic is not so high, contention-free scheduling protocols are not as efficient as contention-based protocols because some scheduled slots are not used.

The basic idea of contention-based duty-cycle protocols is that all nodes or some neighboring nodes wake up at the same slot and they adopt collision-avoidance techniques to contend for the slot. We want to take the same example as in Figure 2.3 to show the basic idea of contention-based duty-cycle protocol. In Figure 2.4, all the nodes are scheduled to wake up in slot 1. When they want to transmit packets, collision-avoidance schemes (e.g. RTS and CTS) are used to avoid collisions. The key point of contention-based duty-cycle protocol is to schedule the wakeup slot of each node according to certain rules, for example, waking up at the same slot as their neighbors' in SMAC [Ye et al., 2002].

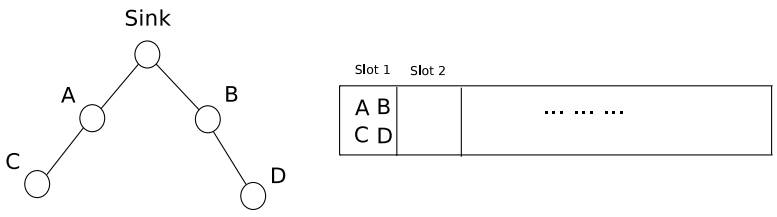


Figure 2.4: A simple example of contention-based schedule

SMAC [Ye et al., 2002], the first work to duty-cycle the sensor nodes for WSNs, assigns each node the same wakeup slot as their neighbors'. Both transmitting and receiving are scheduled in the slot. Due to the fact that the neighboring nodes have the same wakeup slot, RTS/CTS/DATA/ACK scheme is adopted to reduce the collision probability. SMAC's

duty-cycle can save the sensor nodes' energy, but the end-to-end delay drastically increases with the number of hops.

DMAC [Lu et al., 2004], which is specially designed for data gathering applications, has a much shorter end-to-end delay compared to SMAC. In DMAC, the network is organized into a tree. Unlike SMAC, DMAC schedules the nodes' wakeup slots according to their hop count to the sink. The receiving slot of one node is synchronized to the transmitting slot of its child node whose hop count is greater by 1. DMAC assigns the same wakeup slots to the nodes with the same hop count. Those nodes with the same wakeup slots use RTS/CTS-like mechanism to avoid collisions.

Different from the scheduling methods in SMAC and DMAC, DESS [Lu et al., 2005] schedules the wakeup slot (only for receiving, one node can send packets in any other slots in which the next hop node wakes up) of every node in order to reduce the maximum delay from any node to any other nodes. It is similar to the scheduling algorithm proposed in DPS [Chen and Fleury, 2007]. The difference between DPS and DESS is that DPS schedules both transmitting and receiving slots in order to avoid collisions while DESS only schedules the receiving slots. If there are more than one node sending packets at the same slot, the collision or interference will happen.

Similar to the streamlined wakeup technique used in DMAC, PR-MAC [Chen et al., 2007] proposes a bidirectional pipeline technique to schedule the wakeup slots of the nodes along the routing path in order to have a bounded delay of data transmission. Different from DMAC, the sink node in PR-MAC is the coordinator of the network to allocate the frequencies to different paths and to calculate the wakeup offset of the nodes along a routing path. When there are many packets contending to use the same channel, the similar RTS/CTS-like mechanism used in DMAC is used to avoid collisions.

In the previous presented protocols, every node only chooses one next hop candidate node that has a later wakeup slot. If a packet arrives at the sender just after the next hop candidate node goes to sleep, the sender has to wait until the wakeup slot of the receiver in the next period. Thus the delay will be increased a lot. In [Keshavarzian et al., 2006], the delays of some existing scheduling methods are analyzed and a new method called multi-parent technique (one node has two parents which have later wakeup slots) is proposed. By using two parents, the end-to-end delay can be reduced. Similar to the multi-parent scheme in [Keshavarzian et al., 2006], [Zhou and Medidi, 2007] proposes a scheduling algorithm to make each child node have several parent nodes in order to further reduce the end-to-end delay. We can see the reason why the multi-parent scheme has a lower delay by a simple example shown in Figure 2.5.

In this example, node A has a packet to send but the packet arrival time may be different. Node A has two neighboring nodes that could be the next hop node. In the left side of Figure 2.5, node A only chooses the node B as the next hop candidate node while both node B and node C are chosen in the right side of Figure 2.5. As it is shown in the left side of Figure 2.5, the delay of transmitting a packet to a next hop node is delay1 if the packet arrives at t1. And

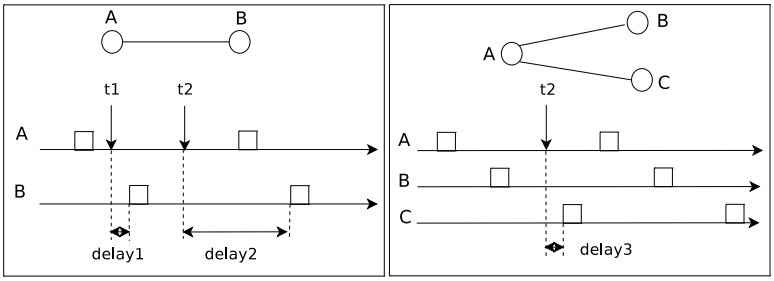


Figure 2.5: Explanation of the basic idea of multi-parent wakeup scheduling

the delay is increased a lot if the event arrives at t2 (delay2 $>$ delay1). But the delay can be decreased dramatically even when the packet arrives at t2 if node C is considered (delay2 $>$ delay3), which is shown in the right side of Figure 2.5.

In addition to the low duty-cycle, some scheduling protocols consider the delay constraints. Recently, a new protocol for low-duty-cycle WSNs has been proposed in [Gu et al., 2009]. In order to reduce the deadline miss ratio and be more energy efficient, the authors propose three methods, called bits augmentation, sink station augmentation and hybrid. Bits augmentation is to add more wakeup slots in a cycle. Sink station augmentation means to add more sinks in the network, which will result in less number of hops. Hybrid combines the previous two methods. Compared with the streamlined wakeup scheme proposed in [Cao et al., 2005], the proposed schemes in [Gu et al., 2009] can consume the less energy with different deadlines and can also have the lower deadline miss ratio.

The previous presented duty-cycle protocols schedule the sensor nodes once after deployment. But when the network conditions change dynamically, those protocols can not react efficiently. DutyCon [Wang et al., 2010] proposes a dynamic duty-cycle control protocol which can have a lower average duty-cycle with an end-to-end delay guarantee. They first decompose the end-to-end delay requirement into a set of single-hop delay requirements. Then the wakeup interval of a receiver's sleep schedule is dynamically changed based on the decomposed single-hop delay requirement by using feedback control theory.

Asynchronous

——Two radio based

The previous described synchronous duty-cycle protocols can have a lower end-to-end delay. Nevertheless, to synchronize the sensor nodes is a challenging and expensive problem for large-scale and low-traffic WSNs because periodically sending time information is needed [Hong and Scaglione, 2005; Sommer and Wattenhofer, 2009]. As a result, asynchronous duty-cycle protocols are more suitable for low-traffic WSNs.

Some protocols which randomly duty-cycle the sensor nodes exist. This kind of scheme does not need synchronization and is easier to implement. The key point of this kind of protocol

is how to wake up the desired sleeping node because one node does not know the wakeup time of another node. Some protocols propose to use two radios, one radio for sending data, and the other low-power radio for sending wakeup signal, e.g. STEM [Schurgers et al., 2002], PTW [Yang and Vaidya, 2004] and LEEM [Dhanaraj et al., 2005].

The basic idea of two radio based asynchronous duty-cycle protocols is shown in Figure 2.6. When one node, say A, wants to send a packet, it will try to wake up the target node, say B, by sending beacons via the wakeup channel. The beacons contain the address or the identification of node B. Node A will continue sending the beacons until an Ack from node B is received. After receiving the Ack from node B, node A transmits the DATA packet through the DATA channel. And node B replies with an ACK after receiving the DATA packet.

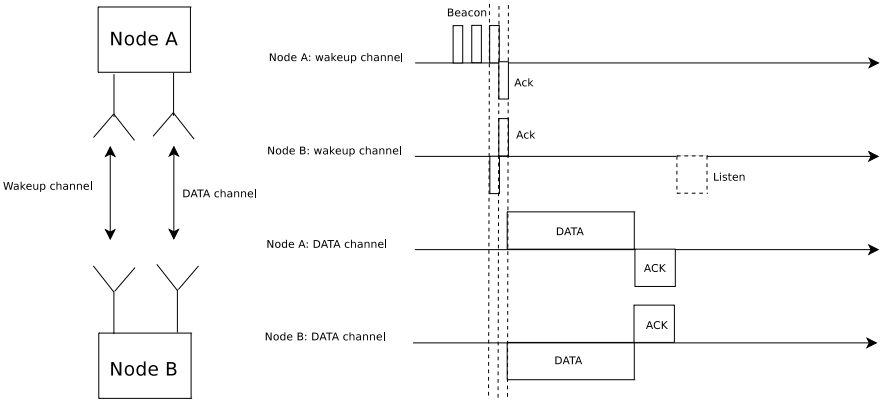


Figure 2.6: Basic idea of two radio based asynchronous duty-cycle protocols

STEM [Schurgers et al., 2002] is the first protocol that proposes to use two radios for WSNs. In STEM, one radio is used for waking up the target node and another is used for transmitting DATA packets. Two versions of STEM are proposed, STEM-B and STEM-T. In STEM-B, beacons are adopted to wake up the target node, as shown in Figure 2.6. In STEM-T, the beacons are replaced by a series of wakeup tone. In this case, the sender will wake up all the nodes that have listened the wakeup tone. STEM can have a very lower energy consumption, but the delay introduced by waking up the target node is great.

PTW (Pipelined Tone Wakeup) [Yang and Vaidya, 2004] adopts the same idea as STEM-T, using a sequence of wakeup tones to wake up the target node. The improvement of this protocol compared to STEM is that they construct a wakeup pipeline to overlap the wakeup procedures. In STEM, a node starts to wake up its next hop node after it successfully receives the DATA packet. In PTW, in order to reduce the delay caused by waking up the next hop node, when a node starts to receive a DATA packet via the data channel, it sends the wakeup tones through the wakeup channel simultaneously. As a consequence, the end-to-end delay of a packet transmitted along a certain path is reduced.

The same as the method used in STEM-B, LEEM [Dhanaraj et al., 2005] uses beacons to

wake up the target node via the wakeup channel. The key idea of LEEM is similar to PTW, waking up the next hop node via the wakeup channel when transmitting the DATA packet via the data channel. Two variances, 1-HAR (One-Hop Ahead Reservation) and N-HAR (N-Hop Ahead Reservation), are proposed in LEEM. In 1-HAR, only the next hop node is woke up when the sender is transmitting the DATA packet. In N-HAR, the next N hop nodes are woke by knowing the routing path. And 1-HAR is used for the case where the length of the wakeup period is shorter than the time to transmit the DATA packet. Otherwise, N-HAR is adopted.

——Single radio based

The protocols described in the previous subsection can have a lower delay, but using two radios for sensor nodes is expensive for large-scale WSNs. In contrast, single radio is much more suitable for large-scale WSNs. For single radio sensor nodes, the preamble sampling technique is adopted by most of the asynchronous duty-cycle protocols. Figure 2.7 shows the basic ideas of three representative asynchronous duty-cycle MAC protocols, i.e. LPL (Low Power Listening) [El-Hoiydi, 2002], X-MAC [Buettner et al., 2006] and RI-MAC [Sun et al., 2008].

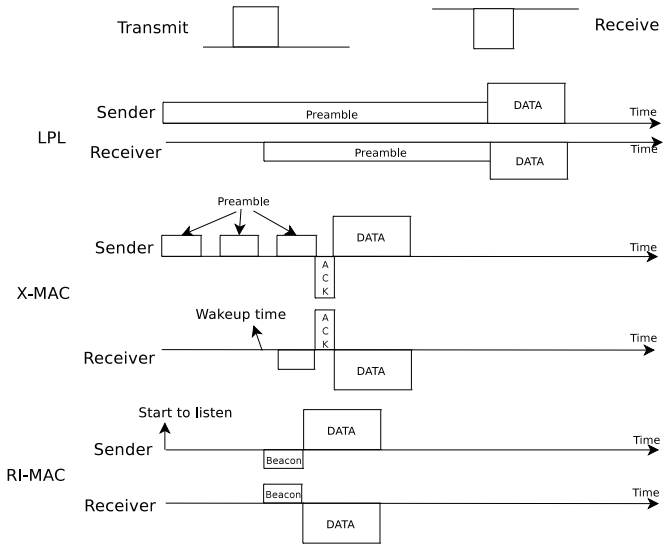


Figure 2.7: Basic ideas of LPL, X-MAC and RI-MAC

As we can see from Figure 2.7, LPL sends a long preamble to wake up the next hop sleeping node before sending the DATA packet. [El-Hoiydi, 2002; El-Hoiydi and Decotignie., 2004; Polastre et al., 2004] belongs to this category. In addition to the basic preamble sampling technique, BMAC [Polastre et al., 2004] proposes some flexible interfaces for the upper layers to dynamically adjust the parameters of the MAC and physical layers. Although the long preamble can make sure that the desired next hop node is woke up, it consumes more energy and incurs lower throughput. The nodes in WiseMAC [El-Hoiydi and Decotignie., 2004] can reduce the length of the preamble by learning the wakeup time of their neighbors.

In order to further reduce the impacts of the long preamble, X-MAC [Buettner et al., 2006]

divides the long preamble into short preambles. A short listening period is inserted between two adjacent short preambles. After the sender receives an ACK packet, it will stop sending the short preamble and start to transmit the DATA packet. The process of X-MAC is shown in Figure 2.7.

Due to the long preamble, the throughput of a random duty-cycle protocol will be decreased when the traffic is heavy. Another kind of asynchronous duty-cycle MAC protocol, receiver-initiated MAC, is proposed, for example, RICER [Lin et al., 2004], RI-MAC [Sun et al., 2008] and DW-LPL [Na et al., 2008]. In receiver-initiated protocols, instead of sending the long preamble, the node that has a packet to send will listen to the medium for a period which has the same length as the preamble. The sender will send the packet after receiving the request from the target node. The process of RI-MAC is shown in Figure 2.7.

In Chapter 2.2.1, we have presented some duty-cycle MAC protocols in different categories. Those protocols have their own advantages and disadvantages which are shown in Table 2.3. It shows that synchronous duty-cycle MAC protocols can have a lower end-to-end delay and be more energy efficient, but they need synchronization. Compared with contention-based protocols, contention-free protocols can have a lower end-to-end delay and be more energy efficient.

For asynchronous duty-cycle MAC protocols, it is easier to implement, but the delay and energy consumption are greater compared to the synchronous duty-cycle protocols. Because two-radio based protocols can use a very low power radio to trigger the radio wakeup of a destination node, it is more energy efficient than single-radio based protocols [Gu and Stankovic, 2005]. And two-radio based protocols can have a lower end-to-end delay than the single-radio based protocols.

Table 2.3: Advantages and disadvantages of duty-cycle MAC protocols

Category		**Delay**	**Energy consumption**	**Difficulty to implement**
Synchronous	Contention-free	- -	- -	+ +
	Contention-based	-	-	+
Asynchronous	Two-radio based	+	+	-
	Single-radio based	+ +	+ +	- -

2.3 General routing protocols

In the previous sections, we have reviewed some general MAC protocols in WSNs. Besides the different MAC protocols, routing protocols for WSNs are also very different due to the following characteristics: (1) low-power node, (2) small memory and (3) large-scale network.

Generally, according to the architecture of routing protocol, it can be classified into the following two categories:

(1) hierarchical routing (or cluster routing). The basic idea of this type of protocol is as

follows: all the nodes are distributedly organized into clusters. One node inside a cluster will be chosen as the cluster head which will send the packets to the base station. All the packets within a cluster are sent to the cluster head. Because a node will run out of energy if it is always the cluster head, the nodes inside a cluster can take turn to be the head. Some famous cluster routing protocols are proposed in [Heinzelman et al., 2002; Younis and Fahmy, 2004].

(2) flat routing. Different from the aforementioned hierarchical routing protocol, there is no hierarchical organization in flat routing protocols. Every node can be the router for other nodes. The packets are forwarded hop by hop until it reaches the sink node.

If we consider the approach of constructing a routing path, routing protocols can be categorized into: (1) proactive routing, (2) reactive/on-demand routing and (3) opportunistic forwarding. Proactive means the routing path is constructed before a packet is generated. Reactive indicates that the protocol will construct the path when a packet is needed to be transmitted. Opportunistic forwarding constructs the routing path after the packet is transmitted.

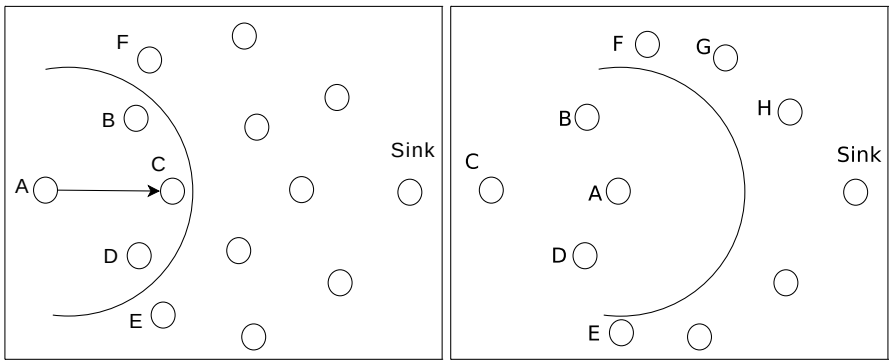
According to the metric adopted by the protocol, routing protocols can also be classified into the following two types: (1) Geographic routing and (2) Virtual coordinate routing. The principles of those two routing protocols are the same. The main difference between them is the used coordinate. Geographic routing uses the exact location to forward the packets while virtual coordinate routing protocols adopt a logical coordinate to route the packets, for example, hop count.

Because cluster routing has to reorganize the network in order to choose new cluster heads for each cluster and it is not energy efficient when the network is very large, we focus on flat routing. In the following subsections, we will introduce two typical flat routing protocols: geographic routing and virtual coordinate routing.

2.3.1 Geographic routing

Geographic routing where every node just needs to know the locations of its one hop neighbors, itself and the sink, is a good choice for large-scale WSNs where hundreds or thousands of sensor nodes are deployed on a large area.

Greedy routing is a very important scheme for geographic routing protocols. Most of the geographic routing protocols are based on this scheme. The basic idea and the problem of greedy routing are shown in Figure 2.8. Figure 2.8(a) shows the basic idea of greedy routing. Node A will choose node C as the next hop because node C is nearest from the Sink node. The famous problem of greedy routing is shown in Figure 2.8(b). When node A wants to send a packet, it can not find a next hop node which is closer than itself from the Sink node. This phenomenon is the so-called Void problem. It can dramatically affect the delivery ratio of geographic routing [Watteyne et al., 2007]. Many different schemes have been proposed to avoid the Void problem.



(a) Basic idea of greedy routing (b) Problem of greedy routing

Figure 2.8: The basic idea and the problem of Greedy routing

GPSR [Karp and Kung, 2000] and GFG [Bose et al., 2001] are the two famous protocols that can avoid the Void problem. When there are no void in the network, the greedy routing scheme is used to forward packets. When a node encounters the Void problem, right-hand rule is adopted to route around the void. Right-hand rule traverses a face in clockwise order, for example, in Figure 2.8(b), node A will traverse the void by the following path: A,B,F,G,H,Sink. In this thesis, we just review some typical geographic routing protocols. More protocols about geographic routing can be found in [Stojmenovic, 2002].

Although geographic routing protocols have many advantages discussed above, they still have the problem about how to get the geographic information. Generally, one can obtain the geographic information by using GPS or other location algorithms [Hightower and Borriello, 2001]. However, using GPS on each sensor node is very expensive and the accuracy of the existing location algorithms is not very high [Seada et al., 2004a]. Therefore, geographic routing protocol also faces some difficulties to be used in large-scale WSNs. For example, they will have worse performances if the location information is not precise [Seada et al., 2004a]. As a result, another kind of routing protocol, virtual coordinate based routing which does not need to know the location information and is based on network connectivity, has been proposed and is more suitable for large-scale WSNs. In the next subsection, we will introduce some typical virtual coordinate routing protocols.

2.3.2 Virtual coordinate routing

Due to the aforementioned problems of geographic routing protocols, virtual coordinate routing protocol is proposed to be used in WSNs. In this subsection, we will review some virtual coordinate routing protocols. According to the dimension of coordinate, virtual coordinate routing protocols can be classified into the following two types: (1) one-dimensional and (2) multi-dimensional.

One-dimensional virtual coordinate

As the name indicates, one-dimensional virtual coordinate means each node only uses an one-dimensional value to represent the coordinate. Some typical metrics used by the one-dimensional virtual coordinate protocols are as follows: (1) hop count, (2) cost, (3) expected transmission count (ETX) and (4) link quality indicator (LQI). Hop count represents the number of hops from the source to the destination. Generally, cost is used to indicate the estimated energy consumption from the source to the destination. ETX indicates the expected transmission count to the destination. LQI is used to reflect the link quality from the source to the destination.

The distance vector based routing protocols which are widely used in wired and wireless networks can be classified into this type, for example, DSDV [Perkins and Bhagwat, 1994], AODV [Perkins and Royer, 1999] and DSR [Johnson and Maltz, 1996]. All these protocols use the hop count as a main metric to route the packets. The differences between these protocols are that DSDV belongs to proactive routing protocol, AODV and DSR belong to reactive routing protocol as we have discussed in the beginning of Chapter 2.3.

For WSNs, one-dimensional virtual coordinate routing protocols usually adopt the hop count to the sink node as the coordinate [Chen et al., 2008; Han et al., 2004]. Some other one-dimensional virtual coordinate routing protocols use the cost information to the sink node [Huang et al., 2009; Ye et al., 2005].

MintRoute [Woo et al., 2003] is an one-dimensional virtual coordinate routing protocol which is implemented in TinyOS. MintRoute uses the ETX metric to route the packet to the sink node. In order to calculate the ETX metric, a node needs to know the round-trip link qualities between itself and its neighboring nodes. And the ETX metric can be calculated by $\frac{1}{Link\ quality}$ where $Link\ quality$ indicates the round-trip link quality. For example, if the round-trip link quality between two nodes is 0.2, the ETX is 5. Due to the fact that to get the ETX metric is not energy efficient, MultihopLQI [MultihopLQI, 2004] is proposed as another routing protocol implemented in TinyOS, where LQI (Link Quality Indicator) is adopted as the coordinate to forward the packet to the sink node. In MultihopLQI, every node can get the LQI after receiving a packet because many off-the-shelf radios have this functionality, for example, [CC2420, 2007]. So MultihopLQI can be much energy efficient than MintRoute.

GRAB [Ye et al., 2005] and SGF [Huang et al., 2009] use the cost information as the coordinate to forward the packet. In GRAB, the sink node broadcasts the initialization packet to all the other nodes. Then, every node can get a cost value to the sink node. When a node transmits a packet, all the neighboring nodes that has decreasing costs will receive this packet. But GRAB only let some of them resend the packet in order to reduce the redundancy and save the energy. The number of nodes that resend the packet is decided by a credit. So the credit can control the trade-off between robustness and energy efficiency.

Although GRAB can have a higher delivery ratio, the energy consumption is also high

because there are multiple paths for sending one packet. SGF uses the same cost information to forward the packet. Different from GRAB, SGF adopts an opportunistic scheme to reduce the energy consumption. Only one neighbor will be chosen as the next hop. Because SGF is a robust protocol which can cater the cases where there are dead nodes or new added nodes, it will be described in Chapter 2.6.

Those one-dimensional virtual coordinate routing protocols are suitable for data gathering applications where all the packets will be sent to the sink node, for instance, MintRoute [Woo et al., 2003] and MultihopLQI [MultihopLQI, 2004]. In some other applications where the packets may be sent from any nodes to any other nodes, one-dimensional virtual coordinate routing is not efficient. Although the protocols like AODV [Perkins and Royer, 1999] and DSR [Johnson and Maltz, 1996] could be used, they are not energy efficient for large-scale WSNs. However, multi-dimensional virtual coordinate routing is very suitable for this situation. In the following subsection, we will review some typical multi-dimensional virtual coordinate routing protocols.

Multi-dimensional virtual coordinate

Two-dimensional virtual coordinate is used to approximate the real coordinate of each node [Leong et al., 2007; Rao et al., 2003; Watteyne et al., 2007]. One or many nodes are chosen as the anchor nodes. They know the exact locations of themselves. Other nodes who do not know their location just average the locations of their neighboring nodes. After iterating many times, every node can get an approximate location. This kind of virtual coordinate updating algorithm can also be seen as a location algorithm.

For other multi-dimensional virtual coordinate routing, generally speaking, the dimension of virtual coordinate is related to the number of landmarks. The hop counts to each landmark are used as the virtual coordinate. Figure 2.9 shows the basic idea of multi-dimensional virtual coordinate routing protocol. In Figure 2.9, there are four landmarks, i.e. node A, B, C and D. Because node A is 2 hop (4 hop and 2 hop) to node B (node C and node D), the virtual coordinate of node A is (0,2,4,2).

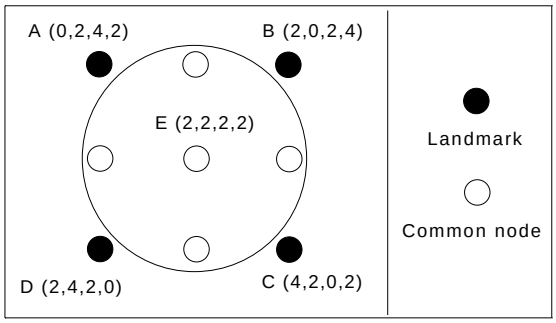


Figure 2.9: The basic idea of multi-dimensional virtual coordinate

For multi-dimensional virtual coordinate routing protocols, the important points are to

choose the landmark and to define the distance between nodes. LCR [Cao and Abdelzaher, 2004] and BVR [Fonseca et al., 2005] are the two typical multi-dimensional virtual coordinate routing protocols. They use the same method as shown in Figure 2.9 to construct the virtual coordinate, but they use different distance metrics to forward the packets.

The aforementioned two protocols do not pay attention to the selection of landmarks. VCap [Caruso et al., 2005] proposes a virtual coordinate assignment protocol which can select three landmarks. And Euclidean distance is adopted in VCap. GLDR [Nguyen et al., 2007] presents the problem of landmark selection and proposes a distributed landmark selection protocol.

Although those virtual coordinate routing protocols can guarantee delivery, they are studied with perfect links. The performances of those protocols under unreliable links are not known. Few multi-dimensional virtual coordinate routing protocols which consider the unreliable links exist.

Based on the results of LCR [Cao and Abdelzaher, 2004], LCRv2 [Cao and Abdelzaher, 2006] proposes an extension for the protocol to be used under unreliable links. The basic idea of this extension is very similar to the idea of ETX. In the virtual coordinate, the hop counts to the landmarks are replaced by the ETX.

By using the idea of ETX metric, ETX-embedding [Wang et al., 2007] proposes a multi-dimensional virtual coordinate routing protocol in which the ETX to each landmark is used as the virtual coordinate. First, some beacon nodes are chosen. And they send beacons to all the other nodes in the network. Then, all the nodes know the ETX to those beacon nodes. In order to reduce the dimension of the virtual coordinate, the MDS (MultiDimensional Scaling) technique is used. They show that the performances of the proposed virtual coordinate based routing are better than that of geographical routing protocols.

In the previous two subsections, geographic and virtual coordinate routing protocols have been presented. In the virtual coordinate routing protocols, one-dimensional virtual coordinate is suitable for data gathering applications. Multi-dimensional virtual coordinate and geographic routing protocols are suitable for the applications where data flows are from any nodes to any other nodes.

2.4 Cross-layer protocols

As discussed in the previous sections, low duty-cycle and unreliable links are two important characteristics of WSNs. Those two characteristics lead to the developments of cross-layer protocols. A survey paper about the cross-layer protocols in WSNs can be found in [Melodia et al., 2006]. In this section, we will review some cross-layer protocols in WSN. More attention will be paid to the cross-layer protocols with the duty-cycle. Because duty-cycle is considered, we want to present the cross-layer protocols in WSNs by the following two categories: synchronized and asynchronous.

Synchronized

By using the synchronization technique, each node can know the wakeup time of their neighbors and send a packet at the right time. The candidate node that wakes up first will be chosen as the next hop node. When the link is unreliable, the packet will be sent to the next candidate node that wakes up secondly to increase the delivery ratio if the previous transmission fails. This is the basic scheme of the synchronized cross-layer protocols. We want to use a simple example in Figure 2.10 to show the basic idea. In this example, five nodes, A, B, C, D and Sink, are deployed. The wakeup time of those nodes in one period is labeled in the figure (Sink is always on). When node D wants to send a packet, node B will be chosen as the next hop node because node D knows the wakeup time of both node B and C, and node B has a earlier wakeup time. The research points of this kind of protocols are to schedule the wakeup slots of each node and to choose the candidate set.

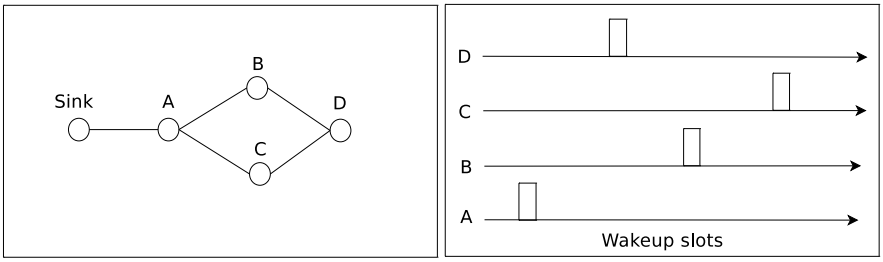


Figure 2.10: Basic ideas of synchronized cross-layer protocol with duty-cycle

In Dwarf [Strasser et al., 2007], every node randomly chooses a wakeup slot. And every node knows the hop count from the sink node. When a node has a packet to send, the nodes with a later wakeup slot and a lower hop count are selected as the candidates. The one with an earliest wakeup slot in the candidate set will first be chosen.

In AMR [Wakamiya et al., 2009], different algorithms to assign wakeup slots for all the nodes are proposed. After every node is assigned a wakeup slot, they include the neighboring nodes with the later wakeup slots into the candidate set.

Although the previous protocols can work under unreliable links, the performances will decrease dramatically. If the protocols can make use of the link quality, the performances will be increased. In DSF [Gu and He, 2007], every node randomly choose a wakeup slot (they assume the wakeup slot is assigned by upper layer protocols, for example, topology control protocol). Every node knows the link quality between themselves and their neighboring nodes. DSF can calculate the optimized candidate set for different objectives, e.g. optimized delivery ratio and optimized end-to-end delay. For example, in Figure 2.11, the expected delivery ratio of node A is $0.9\times0.2+(1-0.9)\times0.4\times0.8=0.212$ if both node B and node C are included in the candidate set. However, the expected delivery ratio of node A is $0.4\times0.8=0.32$ which is greater when only node C is included. Therefore, the optimized candidate set to get optimized delivery ratio is C.

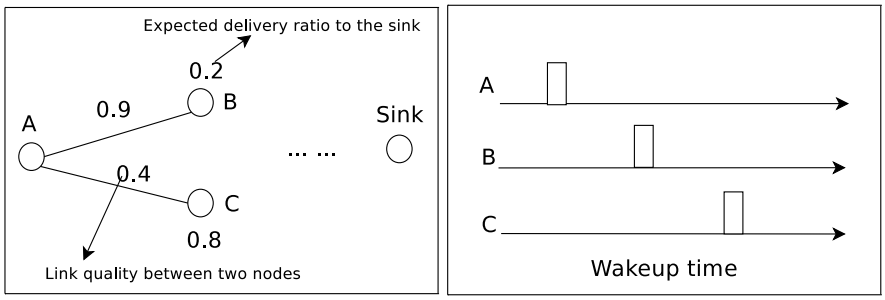


Figure 2.11: A simple example to describe DSF [Gu and He, 2007]

Asynchronous

For asynchronous cross-layer forwarding in duty-cycle WSNs, because every node does not know the wakeup time of neighboring nodes, they have to wait until desired next hop nodes wakes up. After the nodes wake up, the communication between them can start. Generally, the three-way handshake scheme is adopted. When a node has a packet to send, it transmits a RTS-like packet, the nodes received the RTS-like packet send the CTS-like packet with different backoff time. The one that replies first will be chosen as the next hop. The key problem of those protocols is to choose a backoff scheme in order to have a better performance.

In general, those asynchronous forwarding protocols can be classified into two categories: geographic information based forwarding and virtual coordinate (e.g. hop count) based forwarding. In the following parts, some geographic information based forwarding protocols will be introduced.

GeRaF [Zorzi and Rao, 2003] divides the neighboring nodes into different priorities according to the distance to the sink node. The node which is closer to the sink node will be given a higher priority. After receiving the RTS packet, the nodes with a higher priority will send the CTS packet with a shorter backoff time. Similar to GeRaf, the nodes in the forwarding set are given different priorities according to the distance to the sink in [Zeng et al., 2007]. The size of the forwarding set is optimized to make the protocol more energy efficient.

Different from the previous two forwarding protocols, IGF is compatible with IEEE 802.11 networks. In IGF [He et al., 2007], only the neighboring nodes located within $\pm$30-degree angle of the line connecting the sender and the destination are chosen to be the candidate nodes. The nodes within this area choose a random backoff which is between the length of SIFS (Short InterFrame Space) and DIFS (DCF InterFrame Space) in order to compatible with IEEE 802.11 protocol.

In addition to the forwarding protocols which are based on geographic information, there are some other forwarding protocols that adopt virtual coordinate (e.g. hop count) for forwarding, for example, AIMRP [Kulkarni et al., 2006]. The basic handshake scheme in AIMRP is similar to that in the previous described protocols. The difference of AIMRP is that each node is scheduled to wake up with exponential distribution while every node wakes up randomly in GeRaF. And the wakeup parameters of AIMRP can be set according to delay requirements.

It can have a much lower energy consumption compared with SMAC under the same delay requirement.

Although those forwarding protocols are asynchronous, they are not proposed for low duty-cycle WSNs (e.g. 0.1%). For example, it is shown in IGF [He et al., 2007] that the delivery ratio decreases dramatically when the duty-cycle becomes low. This is because a sender can not receive a reply after sending a RTS-like packet due to the fact that most of its neighbors is sleeping. For low duty-cycle WSN, preamble sampling technique is a good choice when the traffic is low.

As we have shown in this section, cross-layer forwarding protocols are very suitable for low duty-cycle WSNs. In the cross-layer forwarding protocols, synchronized forwarding protocols can have a lower delay yet is a bit difficult to implement because of the synchronization. In contrast, asynchronous forwarding protocols are easy to implement yet introduces a greater delay.

2.5 Real-Time protocols

Some WSNs applications have real-time constraints, for example, a fire alarm should be sent to the control center within a required delay deadline. To be real-time, an application has to have both real-time MAC and real-time routing protocols [Li et al., 2007; Stankovic et al., 2003]. In the following subsections, some typical real-time MAC and routing protocols will be presented.

Before introducing real-time protocols, we want to first give the definitions of hard real-time and soft real-time. Hard real-time means a packet must be sent to the destination before the deadline expires, or else, the packet is considered useless. In contrast, soft real-time indicates the system can tolerate certain latency.

2.5.1 Real-time MAC

For real-time MAC protocols, they have to bound the delay to access the medium. If the MAC protocol can not bound the access delay, the upper layer can not guarantee the end-to-end delay. In general, TDMA (Time Division Multiple Access) and FDMA (Frequency Division Multiple Access) are the two typical methods to realize real-time MAC protocols.

I-EDF [Caccamo et al., 2002] is one of the MAC protocols that have hard real-time guarantee. The basic architecture of I-EDF is shown in Figure 2.12. The cellular structure is adopted as the network architecture. FDMA (Frequency Division Multiple Access) is used in inter-cell communication to avoid interferences between cells and TDMA is adopted in intra-cell transmission. Seven different frequencies are used to make sure that there are no interferences between any two neighbor cells. The nodes inside each cell are assumed to be fully connected, which means a node can transmit packets to any other nodes via just one hop. And there is

a router node which is located in the center area of a cell. The schedule of the router node is shown at the bottom of the figure. Although I-EDF can guarantee bounded end-to-end delay, some drawbacks still exist. One of which is that it does not consider the energy consumption of sensor nodes which is one of the most important constraints for WSNs. Another weak point is that the cell structure is difficult to realize for WSNs.

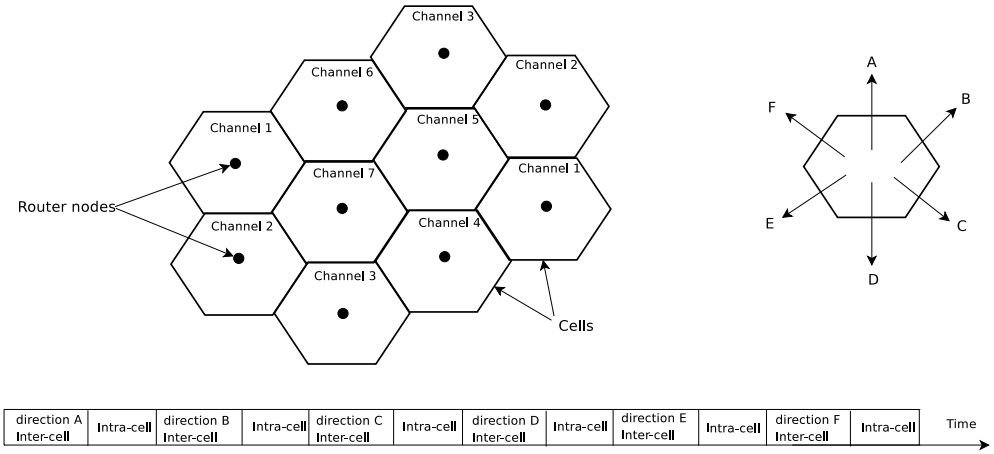


Figure 2.12: The architecture of I-EDF

PEDAMACS [Ergen and Varaiya, 2006] is a TDMA-based MAC protocol designed for traffic monitoring which can guarantee bounded end-to-end delay as well as reduce energy consumption to a large extent. It assumes that the packets from the sink node can reach other nodes in just one hop and the packets from other sensor nodes have to experience many hops to reach the sink node. After collecting the topology information of the whole network, the sink node generates a schedule and broadcasts it to other nodes. Compared with the cell architecture of I-EDF, PEDAMACS is easier to implement and more flexible. However, it is also difficult to implement PEDAMACS in large-scale WSNs because the scheduling algorithm is centralized and the schedule is generated by the sink node.

RT-LINK [Mangharam et al., 2007] is a TDMA-based protocol and applicable to the networks which require predictability in throughput, latency and energy consumption. AM (Amplitude Modulation) radio carrier-current and atomic clock receiver are used in the indoor and outdoor environments respectively to synchronize the sensor nodes. Synchronization pulses are sent periodically. The duration between two synchronization pulses is called a cycle which consists of some frames. One frame consists of some slots classified into two categories: scheduled and contention slots. A node that wants to transmit data sends an HELLO message by randomly selecting a slot in contention slots. This HELLO message is transmitted via multi-hop to the sink node which is responsible for network-wide slot assignment. The slot schedule of the whole network will be distributed to the network. Every node will be active in the assigned transmission and reception slot while asleep in the other slots in order to save energy. Similar to the problem of PEDAMACS, the scheduling algorithm of RT-LINK is centralized.

DUAL MODE [Watteyne and Augé-Blum, 2005] is a hard real-time MAC protocol proposed for linear WSNs (sensor nodes are linearly deployed). The authors assume that the accurate location of every node is known and the sink node is at the end of the linear network. Two modes, unprotected and protected, are used for different situations. Alarm messages are transmitted in unprotected mode if there is no collision. Otherwise, they will be transmitted in protected mode. The protocol runs an initialization phase at first in which nodes are organized into cells. Then, it will run in unprotected mode where the messages can be transmitted faster. When collision happens, the protocol rapidly switches to protected mode which is contention-free but may be much slower. The advantage of this protocol is that it is a distributed protocol. One of the drawback of this protocol is that it does not consider the energy consumption. Furthermore, it has another problem that it is only designed for linear network.

Different from the previous real-time MAC protocols, f-MAC [Roedig et al., 2006] is the only real-time MAC protocol guaranteeing bounded bandwidth and delay without requiring synchronization. In f-MAC, each message is transmitted using small data packets called framelets. The frequency (e.g. three framelets per second) at which framelets are transmitted varies from node to node. With its neighbors having different assigned frequencies, a given node will be able to successfully receive at least one framelet from a source even when all its neighbors are transmitting. Moreover, nodes can be grouped into clusters. All the nodes in the same cluster can communicate with each other at given frequencies which are different from neighbor clusters. The biggest drawback of this approach is its poor bandwidth utilization as the same information is sent in multiple framelets. Furthermore, the worst case delay will increase exponentially with the number of nodes within the same collision domain. This protocol is not suitable for large scale and dense sensor networks.

2.5.2 Real-time routing

In addition to the real-time MAC protocols, real-time routing protocols are also needed in order to have a real-time network. Real-time MAC can bound the one hop delay. Real-time routing protocols must select a path which can satisfy the deadline. In large-scale WSNs, geographic information is very scalable to be used in routing protocols. In the following subsections, some real-time routing protocols which are based on geographic information will be presented.

RAP [Lu et al., 2002], a real-time architecture for WSNs, proposes velocity concepts to differentiate the contenting neighbors. Velocity is defined as the ratio of the distance to the sink to the time left for the packet. The node with a greater velocity will be given a higher priority. Together with the enhanced IEEE 802.11 DCF MAC protocol, RAP has a much lower miss ratio compared with other protocols.

Based on the velocity concepts proposed by RAP, SPEED [He et al., 2003] proposes a new real-time routing protocol. SPEED calculates the velocities that can be provided by all the neighbors. Then, the required velocity will be compared with the velocities provided by the

neighbors. The neighbor which can provide the highest velocity will be chosen as the next hop. By using the neighborhood feedback loop and backpressure rerouting techniques, SPEED [He et al., 2003] has a much lower miss ratio compared with other routing protocols. And the delivery ratio of SPEED is much higher than other routing protocols when the network has the void problem.

Because SPEED drops the packet when a node can not find a next hop node, the miss ratio may be reduced if a better path can be chosen. THVR (Two-Hop Velocity-based Routing) [Li et al., 2009] proposes a two-hop neighboring information based real-time routing protocol. Besides the two-hop velocity based routing decision method, an initiative drop control scheme is proposed. By using the initiative drop control scheme, whether a packet should be dropped when a node can not find a neighboring node that can provide the required velocity is decided by the following two points: (1) the position of the node, (2) the packet loss ratio of all the nodes in the candidate set. The authors have shown by simulation that THVR has a much lower miss ratio and is more energy-efficient compared with SPEED.

In SPEED, the authors only take the real-time constraint into account. In addition to the real-time constraint, MMSPEED [Felemban et al., 2006] also takes the reliability constraint into account by adopting the multi-path technique. SPEED only chooses the neighbor which has the highest velocity as the next hop. Due to unreliable links or void problems, a packet may be dropped. Thus, the probability of reaching the deadline is decreased. In MMSPEED, every node can estimate the average packet delivery ratio to their neighbors. With this estimated value, MMSPEED can calculate the number of required paths. MMSPEED has a higher probability to reach the deadline compared to SPEED.

The aforementioned real-time routing protocols consider the situation where the transmission power is constant. However, the off-the-shelf sensor radios support many different power level, for example, Chipcon CC2420 [CC2420, 2007]. RPAR [Chipara et al., 2006] proposes to adjust the transmission power dynamically to meet the real-time requirements. Beside the adjustable transmission power technique used in RPAR, MCRT [Wang et al., 2009] adopts multi-channels to further reduce the deadline miss ratio.

Although those routing protocols can have good real-time performances, they are based on geographic information. However, it is difficult or expensive to get the geographic information in some WSNs, some virtual coordinate based real-time routing protocols are proposed. For example, [Boughanmi and Song, 2008] proposes a routing metric which can satisfy the energy and delay constraints simultaneously.

In this section, we have presented some real-time MAC and routing protocols. Although those real-time MAC protocols can guarantee the deadline and some of them are very energy efficient, all of them do not take the unreliable links into consideration. In the presented real-time routing protocols, they consider the impacts of unreliable links, but the impacts of the duty-cycle are not considered.

2.6 Robust protocols

Robustness is a very important requirement for some WSNs. In order to be robust, protocols should have the ability to maintain a high reliability when the environment is changed, for example, changed link quality or topology. In the following subsection, some related robust protocols which consider the dead nodes and the new added nodes will be presented.

SOFA [Lee et al., 2006] and SGF [Huang et al., 2009] are two asynchronous forwarding protocols which can update when there are topology changes in the network. SOFA is a solicitation-based forwarding protocol for WSNs. Hop count information is used in this protocol. When a node has packets to send, it will first send a STF (Solicit To Forward) packet. The neighboring nodes who receive the STF packet and are with a lower hop count will reply an ATF (Accept To Forward) packet with a backoff time which is related to the remaining energy. A node will cancel sending its ATF packet during the backoff period if it hears an ATF packet from one of the other nodes. After receiving an ATF packet, the sender will begin to send a DATA packet. If a node can not receive the ATF, it will resend the STF. After a certain times failed, the sender will increase its hop count by 1 and restart the process again.

SGF is a state-free gradient-based forwarding protocol proposed for WSNs. Each node knows the cost to the sink node. After a sender sends the ORTS (Open Request To Send) packet, the receiver will send a CCTS (Competing Clear To Send) packet with a backoff time that is a function of cost. Before a node sends its CCTS, it listens to the channel. The node will give up the CCTS if the channel is sensed busy. After the sender receives the CCTS, it will update the cost according to the link cost and the cost embedded in the CCTS packet.

The characteristic of the aforementioned two protocols is that they are asynchronous. However, both of them have drawbacks. SOFA uses the hop count information to forward the packet. They do not consider the impacts of unreliable links. SGF adopts cost information to forward. The cost information can indicate the link quality. However, they do not pay attention to the low duty-cycle. Some synchronous robust forwarding protocols exist. Because of the synchronization, sensor nodes in those protocols can have a low duty-cycle, as a consequence, they are more energy efficient. The only problem is that it is expensive to synchronize the sensor nodes in large-scale WSNs under unreliable links.

FlexiTP [Lee et al., 2008] and TSMP [Pister and Doherty, 2008] are two typical synchronous forwarding protocols. In those protocols, time is divided into slots. Sensor nodes are synchronized to their neighbors. In order to update the network when some nodes are dead or new nodes are added, special slots are assigned for detecting dead nodes and new nodes. In those special slots, new nodes can send requests to join in the network while some nodes can detect the absence of their neighboring nodes and change the schedule.

2.7 Industrial standards

Nowadays, WSNs are widely used in many different industrial applications. Some industrial standards about WSNs are proposed, for example, ZigBee/ZigBeePRO [ZigBee-Alliance, 2008], WirelessHART [IEC-Standard, 2010], 6loWPAN [IETF, 2010a] and ROLL [IETF, 2010b]. In this subsection, we want to simply introduce those industrial standards. Most of them are based on the IEEE 802.15.4 standard [Baronti et al., 2007; IEEE-Standard, 2003].

2.7.1 IEEE 802.15.4

IEEE 802.15.4 specifies the physical layer and the MAC layer for low-rate wireless personal area networks (LR-WPANs). The physical layer supports the following frequency bands: 868MHz, 915MHz and 2.4GHz. And there is a single channel between 868 and 868.6MHz, 10 channels between 902 and 928MHz and 16 channels between 2.4 and 2.4835GHz. In the MAC layer, it can use the superframe structure (shown in Figure 2.13) which contains an active and an inactive portion. The active portion of the superframe consists of Contention Access Period (CAP) and Contention Free Period (CFP). The contention free period contains Guarantted Time Slots (GTSs) which can guarantee the transmission delay. IEEE 802.15.4 can be used in some WSNs which have real-time constraints.

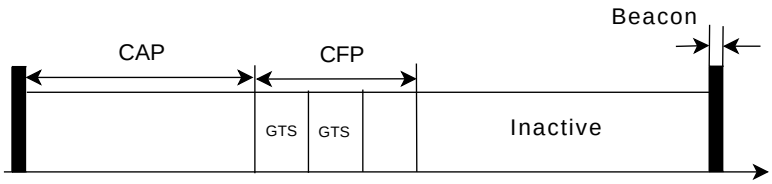


Figure 2.13: An example superframe structure

2.7.2 ZigBee/ZigBeePRO

ZigBee and ZigBeePRO are specifications proposed by ZigBee Alliance in order to provide the upper layers of the protocol stack (from the network to the application layer) using small, low-power radios based on the IEEE 802.15.4 standard for LR-WPANs. In ZigBee routing layer, hierarchical routing strategy is adopted. The ZigBee routers and coordinators are responsible to discover routes. For example, they can use the similar idea to AODV to discover the routes. However, ZigBee routing layer does not consider the real-time constraints.

2.7.3 WirelessHART

WirelessHART is a new industrial standard which has been approved by the International Electrotechnical Commission (IEC). It is mainly used in process control scenarios where users need simple, reliable, secure and cost-effective methods to deliver new measurement values to control systems without the need to run more wires. The radio used in WirelessHART complies

with IEEE 802.15.4. In WirelessHART standard, there is a network manager which is responsible to configure the network, schedule communications between devices. WirelessHART can guarantee the transmission delay with a very high reliablility (e.g. 99.99%), but it is a centralized protocol. It is difficult to be used in large-scale WSNs.

2.7.4 6LoWPAN

6LoWPAN is a working group of IETF which focuses on transmitting IPv6 packets over low-power wireless personal area networks. This Working Group intends to ensure interoperable implementations of 6LoWPAN networks and define the necessary security and management protocols, paying particular attention to protocols already available.

2.7.5 ROLL

ROLL is a working group of IETF which means Routing Over Low power and Lossy networks. Similar to the 6LoWPAN working group, ROLL focuses on IPv6 routing architectural framework for low power and lossy networks. The Framework takes various aspects including high reliability in the presence of time varying loss characteristics and connectivity into consideration while permitting low-power operation in networks potentially comprising a very large number (several thousands) of nodes. This working group pay particular attention to routing security and manageability (e.g., self routing configuration) issues. ROLL also does not take the real-time constraints into consideration.

In summary, although the existing standards have defined the criteria of physical layer, MAC layer or cross-layer, they can not be used in large-scale WSNs because most of those protocols are centralized.

2.8 Motivation of this thesis

In the previous sections, we have presented some typical protocols in WSNs, including MAC, routing and cross-layer forwarding protocols. For those duty-cycle MAC protocols, they are either based on synchronization or based on UDG model. For those routing protocols, most of them does not take the impacts of duty-cycle into consideration. As a consequence, cross-layer forwarding protocol is more suitable for low duty-cycle WSNs. In those cross-layer forwarding protocols for low duty-cycle WSNs, some use UDG model and some other consider the unreliable links. However, none of them takes the real-time constraints into account. Moreover, some existing robust cross-layer forwarding protocols can not work under low duty-cycle. The objective of this thesis is to design protocols for low duty-cycle, event-driven WSNs which have real-time constraints. We want to design new robust and real-time constrained protocols for low duty-cycle WSNs in this thesis.

As presented in Section 2.3, virtual coordinate has advantages to be used in large-scale WSNs. However, most virtual coordinate based protocols are based on UDG model. In order to make virtual coordinate based protocol more reliable, unreliable links should be taken into consideration. We think virtual coordinate based routing protocol under unreliable links is interesting for our problem.

It is shown in Section 2.4 that the layered protocols are not suitable for low duty-cycle WSNs. It is better to have cross-layer protocols. For low duty-cycle WSNs, MAC/routing cross-layer protocol will have a better performance.

In addition to the energy-efficient and real-time constraints, another important characteristic of some WSNs is the dynamic environment, for example, some nodes may die because of running out of the energy or some new nodes may be deployed in the network. As a result, the protocols for WSNs should take this into consideration. This motivates the development of robust (self-healing) forwarding protocol.

2.9 Simulation environment and model

Before presenting the proposed protocols in this thesis, we want to introduce the simulator and the simulation models adopted in this thesis.

2.9.1 Simulator

All our protocols are simulated using WSNet [1], an event-driven simulator specially designed for large-scale WSNs. WSNet has the similar functionalities as those in other event-driven simulators [GloMoSim, 2000; GTNetS, 2008; NS2, 2008]. Different from other simulators, WSNet provides a wide range of physical layer models, including a basic perfect physical layer model and some realistic physical layer models.

2.9.2 Propagation model

When wireless waveforms propagate in the air, it may be diffracted, reflected and scattered. These effects result in different kinds of fading, for example, flat fading, frequency selective fading, slow fading and fast fading. Because of the low-power transmission and the low symbol rates, it is reasonable to assume flat fading channels in WSNs [Karl and Willig, 2005; Rappaport, 1996]. There are log-normal fading, Rayleigh fading and Ricean fading in flat fading models. And some empirical studies have shown that the log-normal shadowing model can provide more accurate channel models than Rayleigh model for indoor environments [Zuniga and Krishnamachari, 2004]. As a result, we adopt the log-normal shadowing model in our simulations.

[1] The WSNet simulator is available at:http://wsnet.gforge.inria.fr/

Here, we want to shortly describe the log-normal shadowing model that we use in our simulations. More details about this model can refer to [Zuniga and Krishnamachari, 2004]. In our simulations, Equation (2.1) is adopted to calculate the path loss (the loss of power when wireless waveforms are propagating). In this model, n is the path loss exponent. X_σ denotes a zero-mean Gaussian random variable with a standard deviation σ^2. The parameters n and σ can be set to different values according to different environments. For example, [Sohrabi et al., 1999] measures the values of those two parameters in different conditions. d_0, which is a reference distance, and $PL(d_0)$ are set to 1m and -52dBm respectively. d is the distance between the transmitter and the receiver.

$$PL(d) = PL(d_0) + 10nlog_{10}\frac{d}{d_0} + X_\sigma \tag{2.1}$$

2.9.3 Confidence interval

The simulation results shown in the figures of this thesis are the average value of 100 rounds of simulations with 95% confidence interval. The details of calculating the confidence interval are presented as follows.

Suppose X_1,..., X_n are independent samples from a normally distributed population with mean μ and variance σ^2. Then we can get the sample mean $\bar{X}$ and sample variance S^2 which are given by

$$\bar{X} = (X_1 + ... + X_n)/n \tag{2.2}$$

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2 \tag{2.3}$$

Then, $T = \frac{\bar{X}-\mu}{S/\sqrt{n}}$ has a t-distribution with $n-1$ degrees of freedom [Ross, 2004]. Because we have 100 samples, here T has 99 degrees of freedom. According to the t-distribution table, we can know the value of c which can satisfy the equation (2.4).

$$Pr(-c < T < c) = 0.95 \tag{2.4}$$

Because $T = \frac{\bar{X}-\mu}{S/\sqrt{n}}$, Equation (2.4) can be written as follows

$$Pr(\bar{X} - \frac{cS}{\sqrt{n}} < \mu < \bar{X} + \frac{cS}{\sqrt{n}}) = 0.95 \tag{2.5}$$

Because we know the sample mean $\bar{X}$, the sample variance S^2 and the value of c, the confidence interval is known as follows

$$[\bar{X} - \frac{cS}{\sqrt{n}}, \bar{X} + \frac{cS}{\sqrt{n}}] \tag{2.6}$$

2.10 Summary

In this chapter, we first discussed the two basic considerations in WSNs, i.e. duty-cycle and unreliable links. Duty-cycle can significantly extend the lifetime of WSNs and Wireless links are very unreliable in WSNs. Then we reviewed the related protocols in detail, including MAC protocol, routing protocol and cross-layer protocol. We classified those protocols into different categories according to their own characteristics. For example, real-time MAC and general MAC, real-time routing and general routing. We further pointed out the advantages and the disadvantages of each category. Then the motivation of this thesis, which is to design reliable and real-time constrained protocols for WSNs, was given. At last, the simulator and the propagation model were presented.

Chapter 3

Routing with virtual coordinate

In the previous chapter, we review the main MAC, routing and cross-layer forwarding protocols in WSNs. We find that unreliable links, virtual coordinate and duty-cycle are very important points to have reliable and real-time communication protocols. In this chapter, we will study about the relationships between the three key points.

The objective of a routing protocol is to find the best path from a source to a destination. In WSNs, every node could be the router for other nodes. Therefore, the functionality of the routing protocol in WSNs is to choose a better next-hop node to the destination. Two kinds of routing protocols are used in computer networks, i.e. Distance Vector Routing and Link State Routing [Tanenbaum, 2003]. In Distance Vector Routing protocol, every router has to remember the distance vectors (e.g. hop count) to all the other routers without knowing the network topology. However, in Link State Routing protocol, every router has to keep the link state information of the network. Thus, the size of memory used to remember the distance vector or the link state information will be increased dramatically if the number of routers in a network is increasing. However, the size of memory in WSNs is small and the number of nodes in some WSNs is very large. As a result, those routing protocols are not suitable to be directly used in WSNs.

Besides the problem of memory, those routing protocols need to exchange many packets in order to get the information of other nodes. Those exchanged packets consume a lot of energy. There are some routing protocols proposed for mobile ad-hoc networks which use the location information to route the packet, for example, LAR [Ko and Vaidya, 2000]. The source node of those location based routing protocols only needs to know the location of the destination, the location of itself and the location of its neighboring nodes. The basic idea of this kind of routing protocol is that every node just has to send the packet to the neighbor node which is closest to the sink. Therefore, location based routing (also called geographic routing) protocol is very suitable for WSNs, for instance, GPSR [Karp and Kung, 2000].

As the authors of GPSR have shown, geographic routing protocol without recovery scheme will have a worse delivery ratio when the network has a void problem (the node that has a packet

to send can not find a next-hop node which is closer to the sink, but a path exists). Although the recovery scheme can increase the delivery ratio, the hop count from the source to the destination is also increased. Moreover, it is very expensive to install GPS module on every sensor node in large-scale WSNs. And the accuracy of some location algorithms is not very high [Patwari et al., 2003]. The performances of geographic routing protocols are significantly deteriorated if the location accuracy is low [Seada et al., 2004a; Watteyne et al., 2007]. However, virtual coordinate based routing protocols are based on connectivity and do not need GPS. Therefore, they are proposed to be used in large-scale WSNs.

The lower layers, e.g. the MAC layer and the physical layer, can have negative impacts on the performances (delivery ratio and end-to-end delay) of virtual coordinate based routing protocols. This is because, for example, the unreliable links can lead to the worse virtual coordinates, and duty-cycle in the MAC layer can cause the longer one-hop delay. They can have a higher reliability if the unreliable links are taken into account, and they will have a lower delay if the duty-cycle is considered. In this chapter, the impacts of low duty-cycle and unreliable links on the virtual coordinate routing protocol will be presented.

Virtual coordinate based routing protocols have better performances than geographic routing protocols in many cases because virtual coordinate based routing protocols can cope the void problem efficiently because they are based on connectivity. But the impacts of lower layers can not be ignored. In Section 3.1.1, the impacts of duty-cycle will be presented.

Wireless links are unreliable [Cerpa et al., 2005; Srinivasan et al., 2008]. In order to have a higher reliability, routing protocols should consider the impacts of the physical layer (e.g. unreliable links). If the routing protocol can take the unreliable links into consideration when it chooses the next hop node, a higher reliability can be achieved. This is the so-called routing/physical cross-layer protocol. In Section 3.1.2, we will show the impacts of unreliable links on virtual coordinate routing protocols.

3.1 Impacts of MAC and physical layers

3.1.1 the impacts of duty-cycle

Low duty-cycle MAC protocols (better for saving energy) have a greater negative impact on the end-to-end delay performance of routing protocols. As a consequence, the routing protocols have to consider the impacts of the MAC layer in order to reduce the end-to-end delay. In this section, we will show the impacts of duty-cycle on the delay performance of routing protocol. As we focus on the rare-event scenario, most of the time, there is no traffic in the network. So, we only consider the impact of duty-cycle with a perfect link. In order to show the impacts of the duty-cycle MAC protocol, we will choose two types of duty-cycle protocols, synchronized and asynchronous, as we have presented in Chapter 2.2.1.

The following representative duty-cycle MAC protocols are chosen to show their impacts

on the delay of routing protocols: S-MAC [Ye et al., 2002], S-MAC-AL [Ye et al., 2004], D-MAC [Lu et al., 2004], BMAC [Polastre et al., 2004] and XMAC [Buettner et al., 2006]. S-MAC, S-MAC-AL and D-MAC belong to synchronous duty-cycle MAC protocols. BMAC and XMAC are asynchronous duty-cycle MAC protocols. And S-MAC-AL and D-MAC are cross-layer protocols which consider the routing layer information.

The details about how to select the path are the objectives of routing protocol. We only assume there exists a h-hop path from the source node to the sink node. Every node has the same duty-cycle, waking up in one slot and sleeping in the remaining slots. For one period, the total number of slots is T. Therefore, the duty-cycle of all the nodes is 1/T. Thus, we can calculate the minimum delay, the average delay and the maximum delay of a packet sent from the source to the sink.

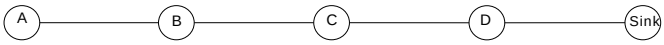
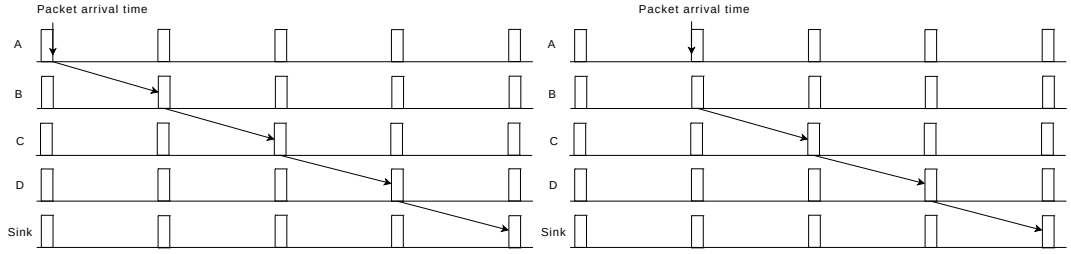


Figure 3.1: A simple topology to explain the delays

In the following section, we want to show the calculation of those delays by using a simple topology as shown in Figure 3.1 which is a selected path in a network. In this simple example, five nodes, A, B, C, D and Sink, are used. Node A tries to send a packet to the Sink after an event happens. And the event could happen at any time. Because we consider the situation that every node has a very low duty-cycle (e.g. 1%), the delay introduced by the awaken period is very small compared to the length of the whole period when we calculate the end-to-end delay.

S-MAC

S-MAC is the most famous MAC protocol in WSNs. The details of S-MAC can be found in [Ye et al., 2002]. The wakeup time of every node is synchronized to their neighbors. Because the packet driven by an event can arrive at any time, we can calculate two types of delay, the maximum delay and the minimum delay which will be shown in Figure 3.2(a) and Figure 3.2(b) respectively.



(a) Maximum delay (b) Minimum delay

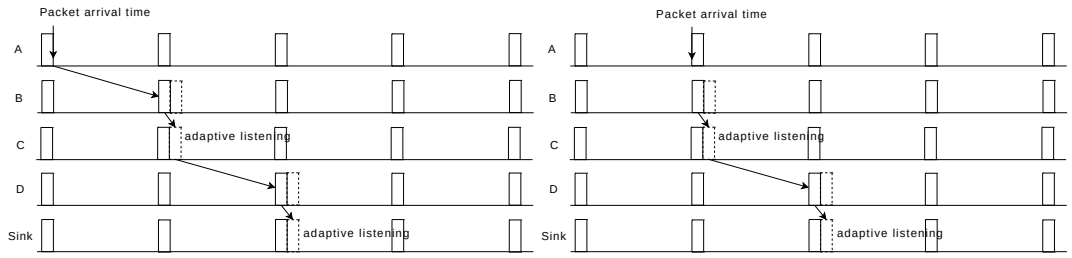
Figure 3.2: The maximum and minimum delays of S-MAC

As we can see from Figure 3.2(a), if the packet arrival time is just after the time when the node wakes up, the delay in this case is the maximum delay, which is hT slots. It is also shown

in Figure 3.2(b) that if the packet arrives just before the time when the node wakes up, the minimum delay, $(h-1)T$ slots, is got. Therefore, the average delay is $(h-0.5)T$ slots.

S-MAC with adaptive listening

Because every node is synchronized to their neighbors, the end-to-end delay is proportional to the number of hops, which is shown in the previous subsection. In order to reduce the end-to-end delay, the authors of S-MAC propose an improvement called S-MAC with adaptive listening (S-MAC-AL). The details can be found in [Ye et al., 2004]. When a node wants to send a packet, it will send a RTS packet during the wakeup period. After a node receives a RTS packet, a CTS packet will be sent. This CTS packet will be used to tell the next hop node to keep awake for another wakeup period. By using this technique, the end-to-end delay is reduced.



(a) Maximum delay (b) Minimum delay

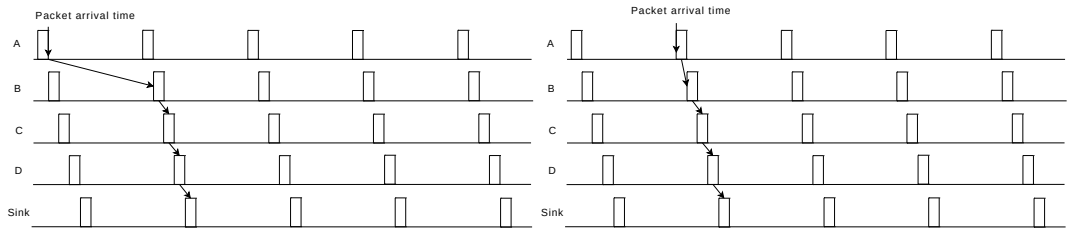
Figure 3.3: The maximum and minimum delays of S-MAC with adaptive listening

We can see from Figure 3.3(a) that the maximum delay is $(\lfloor \frac{h-1}{2} \rfloor T + T)$ slots when the packet arrives just after the time when the node wakes up. The minimum delay is $(\lfloor \frac{h-1}{2} \rfloor T)$ slots when the packet arrival time is just before the time when the node wakes up. The end-to-end delay of S-MAC-AL is about half of the end-to-end delay of S-MAC. The decrease of the delay is because of the consideration of the routing information.

DMAC

In S-MAC-AL [Ye et al., 2004], the authors adopt the cross-layer-like method to reduce the end-to-end delay. They wake up the next hop node in advance. In order to further reduce the end-to-end delay, DMAC [Lu et al., 2004] proposes to schedule the wakeup time as a streamline. The node with a greater hop count will have an earlier wakeup time.

We can see from Figure 3.4(a) that the maximum delay is T slots when the packet arrival time is just after the time when the node wakes up. It is shown in Figure 3.4(b) that the minimum delay is h slots (it is much smaller than the length of one period when the duty-cycle is very low, e.g. 0.1%) when the packet arrives just before the time when the node



(a) Maximum delay (b) Minimum delay

Figure 3.4: The maximum and minimum delays of DMAC

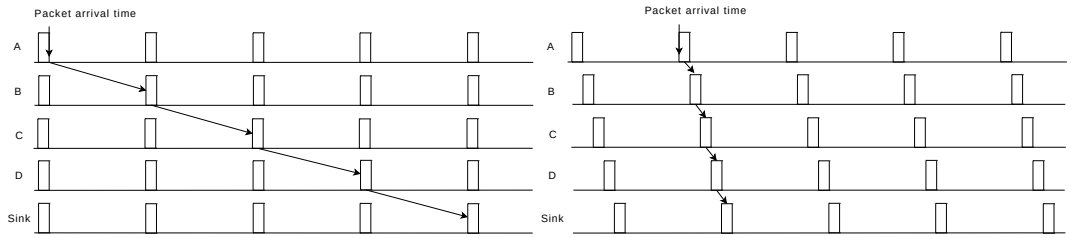
wakes up. Because every node is with very low duty-cycle, the delay introduced by the packet transmission and the wakeup period are neglected.

BMAC

The aforementioned MAC protocols are based on synchronization. In recent years, many asynchronous duty-cycle MAC protocols have been proposed. The most famous one is BMAC [Polastre et al., 2004]. Because BMAC needs to use preamble sampling technique to wake up the next hop node, both the maximum and minimum delay of BMAC are hT slots.

X-MAC

In addition to BMAC, X-MAC [Buettner et al., 2006] is another very famous asynchronous duty-cycle MAC protocol. More information about this type of protocol can be found in Chapter 2.2.



(a) Maximum delay (b) Minimum delay

Figure 3.5: The maximum and minimum delays of X-MAC

As we can see from Figure 3.5(a), the maximum delay reaches hT slots when the wakeup time of all the nodes is the same and the packet arrival time is just after the time when the sender wakes up. And it is shown in Figure 3.5(b) that the minimum delay is h slots when the wakeup pattern is like the pattern in DMAC.

In the previous subsections, we have calculated the maximum delay and the minimum delay of those duty-cycle MAC protocols when they work with a selected path. The delays of all the protocols are listed in Table 3.1.

In order to show the relationships more clearly, the delay and energy consumption rate of those protocols are shown in Figure 3.6. Here, we assume there are 5 hops between the source

Table 3.1: The impacts of duty-cycle on the delay of routing protocol

Protocol	Minimum delay (slots)	Average delay (slots)	Maximum delay (slots)
S-MAC	$(h-1)T$	$(h-0.5)T$	hT
X-MAC	h	$hT/2$	hT
BMAC	hT	hT	hT
S-MAC-AL	$\lfloor(h-1)/2\rfloor T$	$\lfloor(h-1)/2\rfloor T+T/2$	$\lfloor(h-1)/2\rfloor T+T$
D-MAC	h	$T/2$	T

and the sink. The length of one slot is 10ms. We change the duty-cycle from 0.1% to 1%. If we know the length of one slot and the duty-cycle, the length of the total period can be calculated. For instance, the length of the total period is 1s if the duty-cycle is set to 1%. As we focus on the rare-event scenario, most of the time, there is no traffic in the network. So we only consider the period when there is no event. In this case, the energy consumption rate is only related to the duty-cycle. Although the energy consumptions of different protocols are different when they are transmitting packets, it is not taken into consideration because we focus on the rare-event scenario. As a result, we simply use duty-cycle as the energy consumption ratio. If the duty-cycle is greater, the energy will be exhausted much faster.

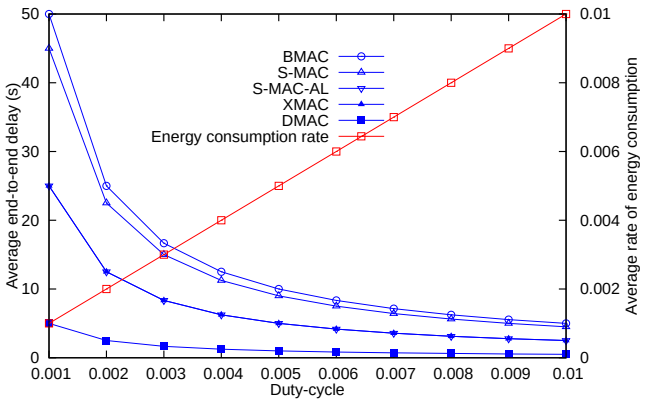


Figure 3.6: The impacts of duty-cycle on the delay of routing protocol

It is shown in Figure 3.6 that the end-to-end delays of all those protocols are decreasing when the duty-cycle is increasing while the energy consumption ratio is increasing as the duty-cycle is increasing. We can see that BMAC has the greatest delay among the MAC protocols. This is because a packet in BMAC can only be forwarded one hop in one cycle. And S-MAC has the second greatest delay because a packet can have a very small delay when the event happens during the wakeup period. Compared to S-MAC, S-MAC-AL has a much lower delay due to the cross-layer technique. XMAC has the same average end-to-end delay as S-MAC-AL because every node in XMAC has asynchronous duty-cycle. DMAC has the lowest delay among those protocols because it takes the routing information into consideration.

We also want to show the average end-to-end delays of those protocols with different number of hops. Here, the duration of the wakeup period is 10ms. The duty-cycle is set to 0.01.

The number of hop between the source and the sink is changing from 1 to 10.

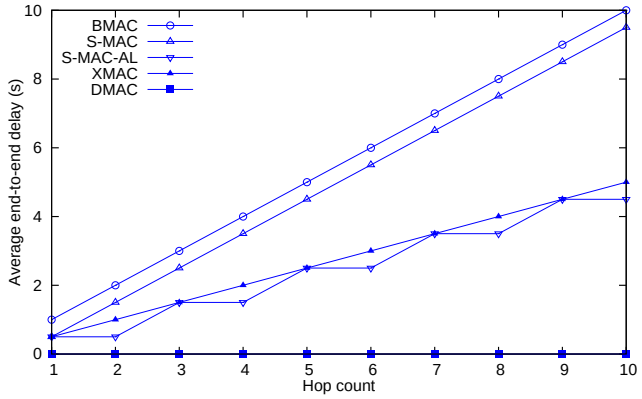


Figure 3.7: Delay vs hop count

It is shown in Figure 3.7 that the average end-to-end delays of all the protocols increase with the increase of the hop count except the delay of DMAC. In the protocols that do not consider the routing layer information, when a node has packets to send to the next hop node designated by the routing protocol, the designated next hop node may be in sleep state. As a result, the induced delay will be much higher. And the reason why the delay of S-MAC-AL has a ”jump” every 2 hops is because S-MAC-AL can forward a packet 2 hops in one period. From Figure 3.6 and Figure 3.7, we can see that the duty-cycle MAC protocol that considers the routing layer information has a lower delay.

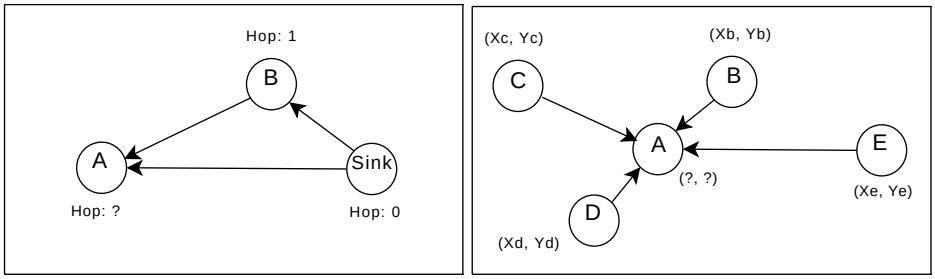
In this section, we have shown the impacts of duty-cycle MAC protocols on the delay performance of routing protocol. From the results we can conclude that it is better to have a MAC/routing cross-layer protocol if a lower end-to-end delay is wanted. However, we only consider the impacts of duty-cycle on routing protocol with a perfect link model in the previous subsections. In the following section, we will show the impacts of unreliable links on routing protocol.

3.1.2 the impacts of the physical layer

As we discussed before, virtual coordinate based routing protocols have some advantages over geographic routing protocols. In this thesis, we focus on virtual coordinate routing protocol.

Unreliable link has a greater impact on the performances (delivery ratio) of virtual coordinate routing protocol because both the process of constructing virtual coordinate and the process of routing are affected dramatically.

The impacts of unreliable links on the process of constructing virtual coordinates are as follows: (1) If the virtual coordinates are based on hop count, how to decide the hop count of a node? (2) If the virtual coordinate is two dimension which aims to approach the real position, which kind of neighbor should be taken into consideration when a node updates its virtual coordinate?



(a) Hop count based virtual coordinate (b) Two dimension virtual coordinate

Figure 3.8: The impacts of unreliable links on constructing virtual coordinate

The impacts of unreliable links on constructing virtual coordinate can be illustrated by Figure 3.8. We use the simple example in Figure 3.8(a) to show the impacts of unreliable links on constructing hop count based virtual coordinate. In this example, if node A receives the hop count information of both sink and node B, how does node A decide its hop count? Under traditional UDG model, the hop count of node A should be 1. This is because node A updates its hop count to 1 when it receives the hop count information of the Sink node. And node A will not change its hop count after it receives the hop count information of node B. However, if we consider the unreliable links, the hop count of node A should be changed according to the link qualities. In Chapter 4.2.4, we will present a way to construct hop counts under unreliable links.

For the two-dimensional virtual coordinate which aims to approach the real position (e.g. [Rao et al., 2003]), how to update the virtual coordinate after one node collects the virtual coordinates of all its neighbors? Does the node need to average the virtual coordinates of all the neighbors? For example, in Figure 3.8(b), node A may be not necessary to consider the virtual coordinate of node E because the link quality between node A and E is very bad. In Chapter 3.2, some new schemes that can solve this kind of problem will be presented.

In this subsection, the impacts of unreliable links on constructing virtual coordinate have been explained by two simple examples. As we discussed before, besides the impacts on constructing virtual coordinate, unreliable links also have some negative impacts on routing. There are many routing protocols which take the link quality into consideration, e.g. [Seada et al., 2004b]. In this Chapter, we just focus on the impacts of unreliable links on constructing virtual coordinate.

3.2 Two-dimensional virtual coordinate under unreliable links

By considering the advantages of virtual coordinate based routing protocols and the impacts of unreliable links, three simple and effective strategies to construct virtual coordinates under unreliable links are proposed. In the proposed strategies, link quality is adopted when every node updates the virtual coordinate. The proposed strategies have better performances than

the protocol proposed in [Rao et al., 2003] where UDG model is used without introducing any additional cost.

3.2.1 Basic simulation parameters

Before presenting the motivation of this work, we want to introduce the basic simulation environment and parameters because the motivation comes from the simulation results. The simulation parameters are listed in table 3.2.

Table 3.2: Simulation parameters for comparing the virtual coordinate updating schemes

Modulation	FSK	Output power	0dBm
Propagation	n = 4 σ = 4	Noise Floor	-105dBm
Sensibility	-105dBm	Data rate	19.2kbps

In the simulations, 40 perimeter nodes are placed on the margins of a square area $100m \times 100m$. Every margin has 11 perimeter nodes. The distance between any two neighboring perimeter nodes is 10m. In fact, the perimeter nodes of a network can be detected automatically by some techniques which can be found in [Leong et al., 2007; Rao et al., 2003]. In this thesis, we do not discuss how to detect the perimeter nodes. The sink node is located at the bottom left corner of the square area and the location is (0, 0). Except the perimeter nodes, other nodes are randomly placed inside the square area. The total number of nodes in the simulations is 1000. The virtual coordinate of the sink is (0, 0) and constant. The virtual coordinate of the perimeter nodes is the same as their real coordinate and constant. The initial virtual coordinate of all other nodes is (50, 50) and they will be updated several times. The position of the perimeter nodes is shown in Figure 3.9. In every simulation, all the nodes except the perimeter nodes send one packet to the sink. Every result is the average value of 100 simulations with 95% confidence interval. Because we focus on showing the improvements of the proposed schemes, all the simulated schemes do not use retransmission.

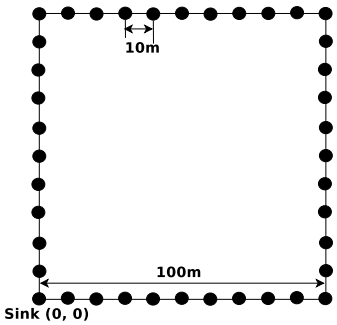


Figure 3.9: The position of the perimeter nodes

3.2.2 Process of virtual coordinate routing protocol

In this subsection, we want to describe the basic process of virtual coordinate routing protocol which mainly includes the following two parts: (1) an initialization phase where every node can get their virtual coordinates, (2) routing with the virtual coordinates.

Initialization phase for getting virtual coordinates

At the beginning, every node except the sink node and the perimeter nodes which know their exact locations randomly chooses a two-dimensional virtual coordinate (for example, (50, 50) in our simulations). Every node sends the packet containing the virtual coordinates of themselves at different time in order to avoid collisions. As it is shown in Figure 3.10, every node is scheduled a different sending time (they could send the packet at the time that is chosen according to their identifications, e.g. node 1 sends first and node 1000 sends last). During one round (the duration when every node sends one packet and updates its virtual coordinate is called one round), every node will receive the virtual coordinates of part or all of its neighbors and update the virtual coordinate of itself according to different schemes.

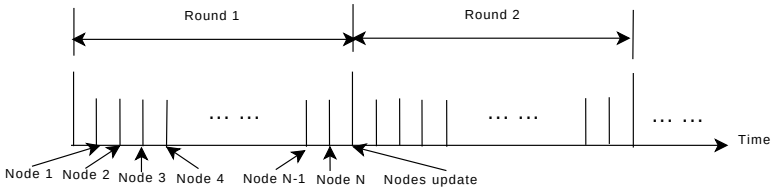


Figure 3.10: A simple example to explain the initialization phase

During the period when the nodes exchange their virtual coordinates, these packets can be used to estimate the link quality by counting the number of packets received from their neighbors. So, every node can get the virtual coordinate and the link quality simultaneously during the initialization phase.

Routing with virtual coordinates

After the initialization phase, each node knows both its neighbors' and its own virtual coordinates as well as the link qualities between them. Because every node needs to exchange packets in order to update their virtual coordinate, those packets can be used to measure the link quality (e.g. PRR) without introducing any additional cost. The PRR is calculated as the ratio of the number of received packets to the number of iterations. Because PRR can precisely represent the link quality compared with the RSSI (Receive Signal Strength Indicator), it is used as the routing metric. During the routing process, greedy routing with some enhanced strategies is adopted. In [Seada et al., 2004b], the metric called "$Dist \times PRR$" ($Dist$ denotes the distance between the sender and the receiver. PRR indicates the packet receive ratio of the receiver) has

the best performances among the other different metrics [Seada et al., 2004b]. Those metrics are listed as follows:

(1) Greedy: chooses the neighboring node which is closer than itself and closest to the destination among the neighboring nodes.

(2) Distance-based: first blacklists the neighboring nodes which are above a chosen distance from itself, then chooses the neighboring node which is closest to the destination from the remaining neighboring nodes.

(3) Absolute Reception-based: first blacklists the neighboring nodes which are below a certain reception ratio from itself, then chooses the neighboring node which is closest to the destination from the remaining neighboring nodes.

(4) Relative Reception-based: first blacklists some neighboring nodes which have the lowest reception ratio (for example, the 20% neighboring nodes which have the lowest reception ratio), then chooses the neighboring node which is closest to the destination from the remaining neighboring nodes.

(5) Best Reception: without blacklisting, chooses the neighboring node which has the highest reception ratio.

We have implemented the "$Dist \times PRR$" metric for routing with the virtual coordinates constructed by "Average-all" [Rao et al., 2003]. But the metric does not have a better performance under virtual coordinate based greedy routing. So an improved metric, "$Dist \times PRR$ and PRR Blacklisting" (PRR Blacklisting means that only the node with a PRR bigger than or equal to a threshold will be considered), is used, which has a better performance than the "$Dist \times PRR$" metric. Here, the proposed metric means that the node which can be considered to be the next hop not only has to have a bigger "$Dist \times PRR$" but also needs to have a bigger PRR. Then, the node with the biggest "$Dist \times PRR$" will be chosen as the next hop.

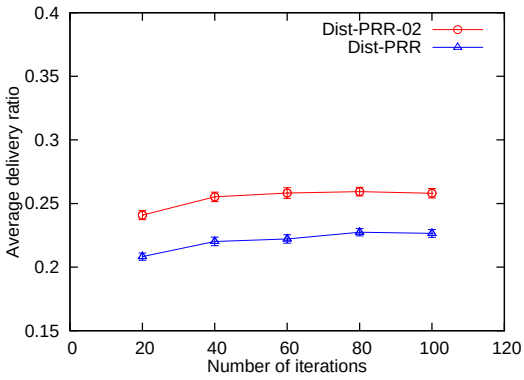


Figure 3.11: The delivery ratio of "$Dist \times PRR$" and "$Dist \times PRR$ and PRR Blacklisting"

In Figure 3.11, it is shown that the delivery ratio of "$Dist \times PRR$" which is denoted by "Dist-PRR" is smaller than the delivery ratio of "$Dist \times PRR$ and PRR Blacklisting" which is denoted by "Dist-PRR-02". In the simulations, only the node with PRR greater than or equal to 0.2 can be considered. As we have presented in Section 2.9.2, log-normal shadowing model is adopted. In our simulations, the physical layer environment is set harsh, for example, the path loss exponent (n) is set to 4 (the greater the exponent is, the harsher the environment will be. Generally, it is about 2, for example, at an apartment hallway [Sohrabi et al., 1999]). And because retransmission can increase the delivery ratios of all the compared schemes, for the sake of simplicity, we do not allow retransmission in the simulations. These result in the fact that the delivery ratio is low in the simulation results. If we set less harsher environment and allow retransmission, the delivery ratio can be increased. The reason why "$Dist \times PRR$ and PRR Blacklisting" has the higher delivery ratio than "$Dist \times PRR$" is as follows:

(1) "$Dist \times PRR$ and PRR Blacklisting" avoids to choose the neighbors with bad link qualities which may have a bigger "$Dist \times PRR$".

(2) If the virtual coordinate of every neighbor is updated in every period, "$Dist \times PRR$ and PRR Blacklisting" chooses the neighbor with a higher link quality.

3.2.3 Motivation

In this section, we want to describe the motivation of our proposed update strategies. First, the method used in the existing work [Rao et al., 2003] to update the virtual coordinate will be depicted.

Equation (3.1) is used in [Rao et al., 2003] to update the virtual coordinates of every node. x_k and y_k denote the coordinate x and y of node k, respectively. And $neighborset(i)$ is defined as the neighbor set of node i. $sizeof()$ means the size of the specified objective.

$$\begin{cases} x_i = \frac{\sum_{k \in neighborset(i)} x_k}{sizeof(neighborset(i))} \\ y_i = \frac{\sum_{k \in neighborset(i)} y_k}{sizeof(neighborset(i))} \end{cases} \tag{3.1}$$

Although the strategy used in [Rao et al., 2003] has good performances under the UDG model, they do not pay enough attention to the impacts of unreliable links. In their work, they only do one simulation where they set different loss possibilities to control packets and they do not discuss about how to improve the performances of the proposed strategy under unreliable links.

After simulating the proposed scheme in [Rao et al., 2003] under a more realistic propagation model, we found a big gap between the delivery ratio of the scheme in [Rao et al., 2003] and the delivery ratio of the geographic routing. Nevertheless, the performances of the proposed scheme in [Rao et al., 2003] and the performances of the geographic routing protocol are shown to be very close in [Rao et al., 2003]. This gap motivates us to design some

new update schemes. Before introducing the gap, we want to shortly depict the compared two schemes.

(1) Average-all. This scheme represents the method proposed in [Rao et al., 2003], which means every node considers all the neighbors in the neighbor table to update the virtual coordinates no matter what the link quality is.

(2) Geography. Every node uses the location information for routing. They also exchange packets during the initialization phase. Those packets are used for measuring the approximate link quality. In fact, the more the packets are sent during the initialization phase, the more precise the virtual coordinate and the estimated link quality will be.

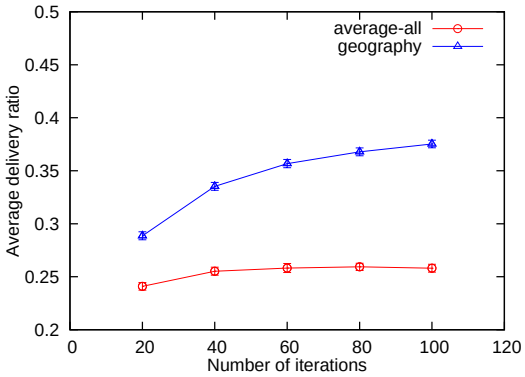


Figure 3.12: Delivery ratio difference under unreliable links

In Figure 3.12, it is shown that the delivery ratio of “average-all” is almost constant no matter how many times the virtual coordinate is updated. And the delivery ratio of “geography”, which denotes the geographic routing, is higher than that of “average-all”. Moreover, the delivery ratio of “geography” is increasing with the number of iterations. Why does the delivery ratio of geographic information based routing increases with the number of iterations? Actually, it should be almost constant if the link quality between nodes is precisely measured. In our simulations, the packets exchanged between the nodes which are used for updating the virtual coordinate are also used to calculate the link quality. As a result, the more the packets are exchanged, the more precise the link quality will be. Thus, the delivery ratio of the geographic information based routing is increasing with the number of iterations. And the difference between the two methods is great under the realistic link model although they have almost the same value under UDG model in [Rao et al., 2003]. Why does this phenomenon happen? It is because every node includes many neighbors that have low link quality into the neighbor table. Even iterating for many rounds, the virtual coordinates of all the neighboring nodes are not precise enough. Can the performances be improved by using different iteration strategies? This question will be answered in the next sections.

Table 3.3: Structure of neighbor table

ID	VC_x	VC_y	$Count$	$RSSI$	$flag$

3.2.4 Different update methods

In this section, the proposed update methods will be described in detail. Every node keeps a neighborhood table as shown in Table 3.3. If a node receives a packet from one of its neighbors, it will update the corresponding information in its neighborhood table. ID indicates the identifier of the neighboring node. VC_x and VC_y denote the virtual coordinate x and y of the neighboring node, respectively. $Count$ is used to count the number of received packets from the neighboring node indicated by ID. $RSSI$ records the estimated RSSI (Receive Signal Strength Indicator) value. $flag$ is used to mark if the packet from this neighboring node is received or not in the current round. If it is received in the current round, $flag$ is set to 1, otherwise, 0. Four update methods will be presented in the following section:

(1) Average-all. This method is used by [Rao et al., 2003]. They do not consider the link quality and average the virtual coordinates of all the nodes in the neighborhood table. The following three methods are proposed by us.

(2) Average-new. Average the virtual coordinates of the nodes from which the packets are received in the latest round.

(3) Average-new-RSSI. Average the latest received neighbors' virtual coordinate according to the RSSI.

(4) Average-new-threshold09. Average the neighbors' virtual coordinate according to the latest calculated link quality.

In the following sections, we will describe those schemes one by one in detail.

Average all

If unreliable links are considered, one node can not receive the virtual coordinate of all the neighbors at each period. In this scheme, the performances of the strategy proposed by [Rao et al., 2003] under realistic link model are evaluated. Every node updates the virtual coordinate by averaging the virtual coordinates of all the nodes in the neighborhood table even if some nodes' virtual coordinates are not updated in the current round (as shown in Equation (3.1)). This strategy does not consider any information of unreliable links. In the following sections, the proposed strategies that take link quality into account will be presented.

Average new

Due to the impacts of unreliable links, some nodes may be the neighbor of some nodes in one time and not be the neighbor in another time. So, if only the nodes from which the packets

are received in the latest round are considered (averaging the virtual coordinates of the neighbors which have $flag$ equal to 1), the performances will be better than "average-all" because those neighboring nodes have a better link quality. In Equation (3.2), $newneighborset(i)$ denotes the neighborhood set which contains all the nodes from which the packets are received in the latest round ($flag$ == 1). x_k and y_k have the same definitions as in Equation (3.1). $sizeof(newneighborset(i))$ indicates the size of the neighborhood set. This scheme takes the link quality into consideration. But it only divides the link into two types: good and bad. If the packet from one node is received in this round, the link between the two nodes is thought to be good. Otherwise, the link is bad. Although this strategy can improve the performances, the details of unreliable links are not considered. In the following section, we will introduce another scheme which consider more information of unreliable links.

$$\begin{cases} x_i = \frac{\sum_{k \in newneighborset(i)} x_k}{sizeof(newneighborset(i))} \\ y_i = \frac{\sum_{k \in newneighborset(i)} y_k}{sizeof(newneighborset(i))} \end{cases} \tag{3.2}$$

Average new RSSI

In this strategy, the RSSI is taken into consideration. The reason why the RSSI is adopted is that the nodes with better link qualities will be used to update the virtual coordinate. And due to the unreliable links, some nodes which are far away may have a higher RSSI and some other nodes that are close may have a lower RSSI. As a result, Exponentially Weighted Moving Average (EWMA) is adopted to estimate the RSSI between nodes in each round. By using the EWMA, the value can reflect the link quality between nodes more precisely. Equation (3.3) is used to calculate the new RSSI. $RSSI_n^e$ and $RSSI_{n-1}^e$ denote the latest estimated value and the previous estimated value, respectively. The latest measured value is denoted as $RSSI_n^m$. In each round, $RSSI_n^m$ is set to a new value. If one node receives a packet from a neighbor in one round, the corresponding $RSSI_n^m$ is the estimated value. If there is no packet received from one neighbor in one round, the corresponding $RSSI_n^m$ is set to 0. In our simulations, α is set to 0.5. Actually, α can be set to other different values. But the difference is very low when α is changing from 0.3 to 0.9. We will show the impacts of α in the next subsections. After getting the estimated RSSI value of every neighbor, each node updates the virtual coordinate by using weighted average. The weight is the estimated RSSI (Equation (3.4)).

$$RSSI_n^e = RSSI_{n-1}^e * \alpha + RSSI_n^m * (1 - \alpha) \tag{3.3}$$

$$\begin{cases} x_i = \frac{\sum_{k \in newneighborset(i)} x_k * RSSI_k}{\sum_{k \in newneighborset(i)} RSSI_k} \\ y_i = \frac{\sum_{k \in newneighborset(i)} y_k * RSSI_k}{\sum_{k \in newneighborset(i)} RSSI_k} \end{cases} \tag{3.4}$$

It is worth noting that we give a greater weight to the node with a better link quality. But

the value of RSSI is negative. Therefore, RSSI is transformed to a positive value. The Formula used to transform is shown in Equation (3.5). The reason why we use 105 here is because the sensitivity is set to $-105dBm$. By using this Equation, the node with a greater RSSI will have a greater weight.

$$RSSI_{transformed} = 105 + RSSI_{real} \tag{3.5}$$

Average new threshold09

In this strategy, a threshold of the link quality is set to decide if the neighbor should be considered to average. In each round, every node will look for the biggest $Count$ value in the neighborhood table which indicates how many packets should be received from their neighbors in the best case. After getting the biggest $Count$ value in the neighborhood table, every node can estimate the link qualities between themselves and all their neighbors according to Equation (3.6). Here, LQ_i denotes the link quality between the node and its neighbor i. $Count_i$ is defined as the number of packets received from node i. If LQ is bigger than 0.9, this neighbor will be added for averaging. If not, it will not be considered. Of course, we can set the threshold other values. Here, just the performances when threshold is equal to 0.9 are shown because they are much better than other cases. More information about the impacts of the threshold will be presented in the next sections.

$$LQ_i = \frac{Count_i}{max_{k \in neighborset(i)}\{Count_k\}} \tag{3.6}$$

$$\begin{cases} x_i = \frac{\sum_{(k \in newneighborset(i)) \& (LQ_k >= 0.9)} x_k}{sizeof(newneighborset(i))} \\ y_i = \frac{\sum_{(k \in newneighborset(i)) \& (LQ_k >= 0.9)} y_k}{sizeof(newneighborset(i))} \end{cases} \tag{3.7}$$

3.2.5 Performance evaluation

In this section, we will show the performances of the proposed update schemes in terms of average delivery ratio and average hop count. The two metrics are defined as follows:

(1) Average delivery ratio: the ratio of the number of packets received by the destination to the number of packets sent by the sources is defined as delivery ratio. The average value of a high number of simulations is defined as average delivery ratio.

(2) Average hop count: every packet received by the destination has a hop count. In one simulation, we can calculate the average value of those hop counts. The average hop count is defined as the average value of many times simulations.

Impacts of α in Formula (3.3)

In this subsection, we want to show the impacts of α in Formula (3.3) where we use EWMA in order to get a more precise link quality value. As a weight, α is adopted in this formula.

Different values of α may have different impacts on the performances. In this subsection, the impacts of α will be evaluated. The value of α is changing from 0.3 to 0.9. Other simulation parameters are the same as those described in previous section.

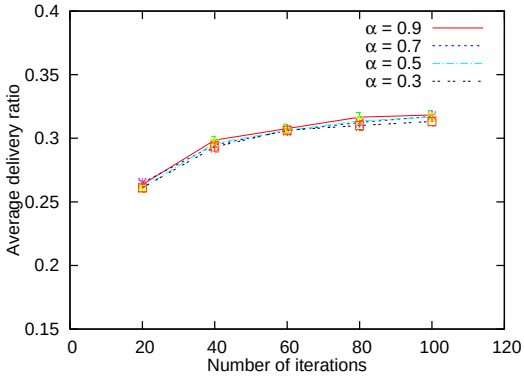


Figure 3.13: Impacts of α

We can see from Figure 3.13 that the delivery ratio is almost the same no matter how we change the value of α. It turns out that changing α has almost no impacts on the performances. Therefore, α is set to 0.5 in all our other simulations.

Impacts of the threshold in Formula (3.6)

In Section 3.2.4, we have described the proposed update schemes. For the scheme called "average-new-threshold09", a threshold is used. In this section, we want to show the impacts of the threshold.

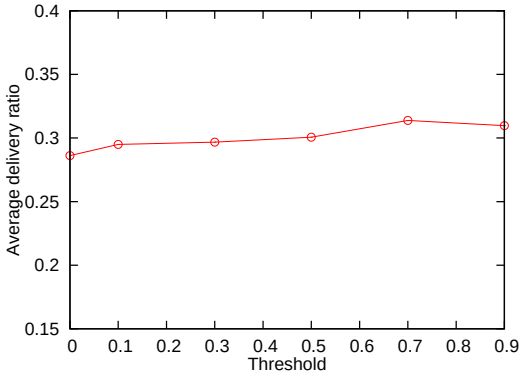


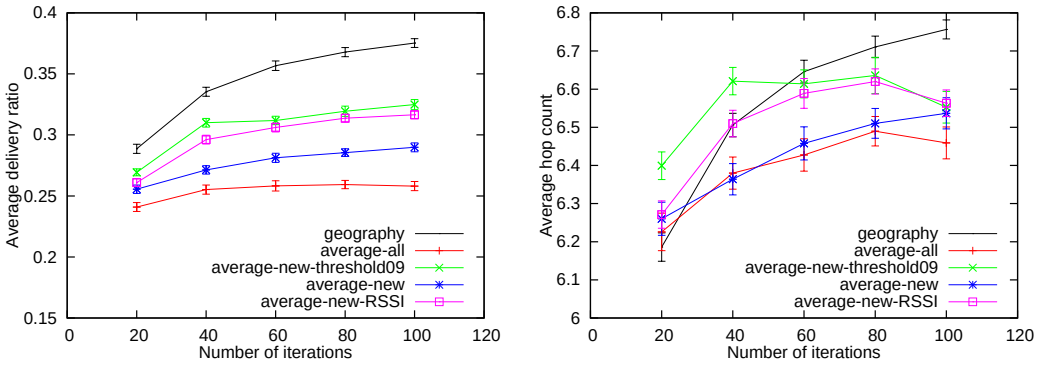
Figure 3.14: Impacts of threshold

It is shown in Figure 3.14 that the average delivery ratio is increased a little when the threshold is changing from 0 to 0.1. The average delivery ratio is almost the same when the threshold is changed from 0.1 to 0.5. It is increased again when the threshold is 0.7. But the

difference of average delivery ratio is very low as the threshold changes from 0.7 to 0.9. In our following simulations, the threshold is set to 0.9.

Impacts of the number of iterations

The process to update the virtual coordinates consists of many rounds. And the more rounds the network runs, the more precise the virtual coordinate and the link quality will be. In this subsection, we want to show the impacts of the number of iterations. The locations of the perimeter nodes are the same as described in Chapter 3.2.1. The number of iterations is changed from from 20 to 100.



(a) Average delivery ratio (b) Average hop count

Figure 3.15: The impacts of the number of iterations

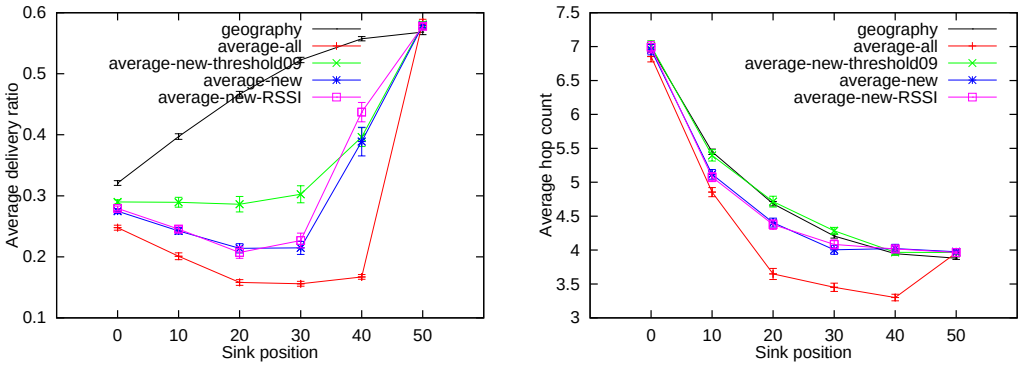
In Figure 3.15(a), we can see that the “average-all” (the protocol proposed in [Rao et al., 2003]) has the smallest delivery ratio among all strategies. The delivery ratio is about 0.24 with 20 iterations and then keeps to 0.25 no matter how many iterations we do. “average-new” has the delivery ratio a little higher than “average-all”. The reason why “average-new” has higher delivery ratio than “average-all” is because “average-new” use more nodes with higher link quality than “average-all”. The “average-new-threshold09” has the delivery ratio most closer to “geography” because the neighbors it chooses for updating are most similar with the neighbors used in unit disk model. So, it has the most precise virtual coordinate which leads to higher delivery ratio. And the “average-new-RSSI” has the moderate delivery ratio that is very close to “average-new-threshold09”. From the results in Figure 3.15(a), we can see that “average-new-threshold09” is the best one among all the strategies. And “average-new-RSSI” is also a good choice.

In Figure 3.15(b), it shows that the average hop count of these strategies is very close but with some differences. At the beginning, “geography” has the smallest hop count because all other virtual coordinate based methods do not have precise virtual coordinate at this time. After iterating several times, they have smaller hop count than “geography”. It is shown in Figure 3.15(b) that all the virtual coordinate protocols have smaller hop count than “geography” from

60 iterations. This is because all the virtual coordinate protocols have smaller delivery ratio. Among the hop counts of all the virtual coordinate protocols, “average-new-threshold09” has the biggest hop count and “average-all” has the smallest hop count. The order is the same as that of delivery ratio in Figure 3.15(a). With 100 times’ iterations, the hop counts of “average-new-threshold09” and “average-new-RSSI” are very close to the hop counts of “average-new”. From this point we can conclude that “average-new-threshold09” and “average-new-RSSI” have best performance due to the fact that they have higher delivery ratio and without increasing the hop count.

Impacts of the sink position

In the previous sections, the average delivery ratios of all the update schemes are not so great. That is because the sink node is at (0, 0) and the protocols do not use retransmission. In this section, we want to show the performances of those update schemes by changing the position of the sink node from (0, 0) to (50, 50). The location of the perimeter nodes is the same as those in the previous sections. And 40 round iterations are used in this simulation.



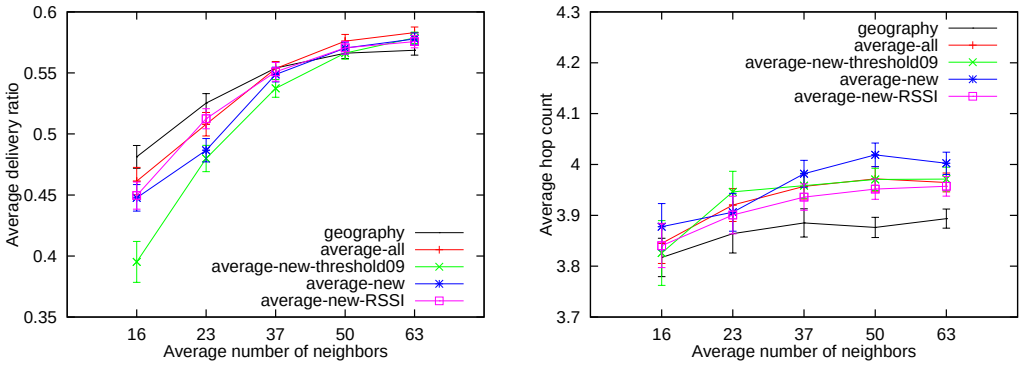
(a) Average delivery ratio (b) Average hop count

Figure 3.16: The impacts of the sink position

We can see from Figure 3.16(a) that ”geography” has the higher delivery ratio than other update schemes when the location of the sink node is not (50, 50). And the delivery ratio of “geography” is a little bit lower than that of other update schemes when the sink node is at the center. This is because the number of hop to the sink node is decreased when the sink node is located at (50, 50). And the sink node is also a landmark node when it is moving from (0, 0). The strange result is that the average delivery ratios of all the update schemes are decreasing as the sink node moves to the center. It is shown in this result that it is better to have symmetric landmark nodes (e.g. (0,0) and (50, 50) in this simulation). We can also conclude that “average-new-threshold09” has a better delivery ratio when the average hop count to the sink node is higher. And “average-all” can have a higher delivery ratio when the hop count to the sink node is very small.

Impacts of the density with sink at the center

From the previous simulation results, we can see that "average-new-threshold09" has the greatest delivery ratio among the proposed update schemes when the sink node is located at (0, 0). And the average delivery ratio of "average-new-threshold09" is lower than that of "average-all" when the sink node's location is (50, 50). In this section, we want to show the impacts of the density on the performances of those proposed update schemes. In this simulation, the sink node is located at (50, 50). And the number of deployed nodes is chosen from the following set $\{200, 400, 600, 800, 1000\}$. Because of the unreliable links, one node sometimes can receive packets from some neighbors that are far away. In this simulation, we only count the neighbors that have the link quality which is higher than 0.2. As a result, the average number of neighbors with the aforementioned number of nodes is $\{16, 23, 37, 50, 63\}$.



(a) Average delivery ratio (b) Average hop count

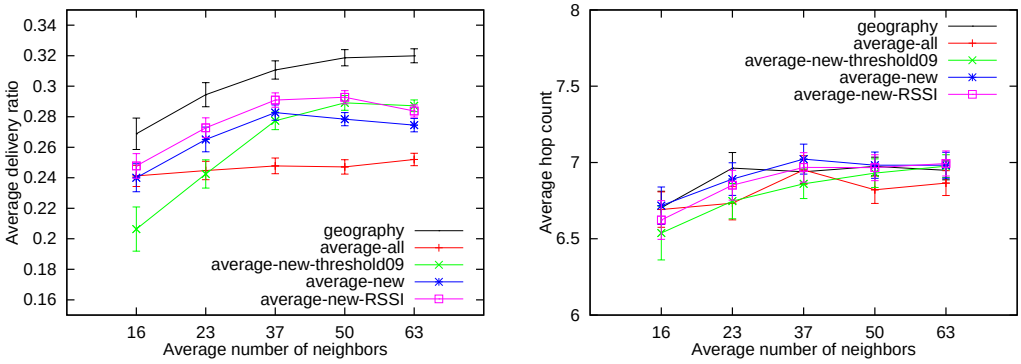
Figure 3.17: The impacts of the density with sink at the center

We can see from Figure 3.17(a) that "average-all" has the greatest delivery ratio among the proposed update schemes. A very important observation is that the average delivery ratio of "average-new-threshold09" is the lowest one when the average number of neighbors is 16, 23, 37 and 50. This is because "average-new-threshold09" excludes many neighbors from the neighbor table. And it does not have enough choices, which results in the lower delivery ratio. The average delivery ratio of "average-new-RSSI" is very close to that of "average-all" when the network density is low.

Impacts of the density with sink at the corner

In this subsection, we want to show the impacts of the density on the performances of those proposed update schemes when the sink node is located at (0, 0). The average number of neighbors is chosen from the following set 16, 23, 37, 50, 63.

It is shown very clear in Figure 3.18(a) that "average-new-threshold09" has a higher delivery ratio than other proposed update schemes when the average number of neighbors is 63.



(a) Average delivery ratio (b) Average hop count

Figure 3.18: The impacts of the density with sink at the corner

The average delivery ratio of “average-new-RSSI” is a little bit lower than that of “average-new-threshold09” when the density is high. As the density decreases, “average-new-RSSI” has the highest average delivery ratio while the delivery ratio of “average-new-threshold09” is the lowest one.

In Section 3.2, we have presented three update schemes for virtual coordinate routing. From the simulation results, we can see that the proposed schemes have their own advantages to be used in different scenarios. We can conclude as follows:

(1) “average-new” has a higher delivery ratio when the network is small-scale, as it is shown Figure 3.17(a) and Figure 3.18(a).

(2) “average-new-threshold09” has a higher delivery ratio when the network density is high (see Figure 3.15(a) and Figure 3.18(a)).

(3) “average-new-RSSI” is a good choice because it has a good performance in terms of average delivery ratio no matter what is the scenario. For example, the average delivery ratio of ”average-new-RSSI” is very close to that of ”average-new-threshold09” in Figure 3.15(a) and the average delivery ratio of ”average-new-RSSI” is higher than that of ”average-new-threshold09” in most of the cases in Figure 3.18(a).

Because there exist asymmetric wireless links in low power WSNs, we want to discuss the impacts of asymmetric links on the proposed updating methods. The important point is that our proposed updating methods can work even when there exist many asymmetric links, but the performances will be decreased. A way of enhancing the performances of our proposed methods under asymmetric links is that the round-trip link quality (the success probability that both the DATA packet is received by the receiver and the ACK packet is received by the sender) can be adopted instead of only the one-way link quality to update the virtual coordinates in our proposed updating methods.

3.3 Summary

In this chapter, we studied the relationships between the three key points which are important to have reliable and real-time constrained communication protocols, namely virtual coordinate, unreliable links and duty-cycle. We first studied the impacts of the duty-cycle MAC and unreliable links on the performances of virtual coordinate based routing protocols. We showed that it is necessary to have MAC/routing cross-layer protocols for ultra-low duty-cycle WSNs. This study result gives us the guideline to design the cross-layer protocols which will be presented in the next chapters.

Then, we discussed the impacts of unreliable links on constructing virtual coordinate and routing. And it appears that it is better for the virtual coordinate routing protocols to consider the unreliable links when constructing the virtual coordinate and routing. The virtual coordinate discussed in this chapter are two-dimensional. And two-dimensional virtual coordinate is better to be used in the case where the traffics are from any node to any other nodes. However, the traffics in some kind of WSNs are all flowed to the sink. As a result, one-dimensional virtual coordinate is suitable for this case (e.g. hop count to the sink). With this kind of one-dimensional virtual coordinate, it is better to take the unreliable links into account because many node may have the same hop count. Moreover, the one-dimension virtual coordinate and the unreliable links can be exploited by the MAC layer. Based on those observations and the study results that it is better to have cross-layer protocols for ultra-low duty-cycle WSNs, we have proposed some cross-layer forwarding protocols which will be presented in the next chapters.

Chapter 4

Reliable and time constrained forwarding

4.1 Introduction

As we have presented in Chapter 1.1.1, WSNs applications can be mainly classified into the following four types of data flows: event-driven, query-driven, continuous and hybrid. In this thesis, we focus on the event-driven WSNs applications. For many event-driven applications, once an event happens, the alarm packet has to be reported to the sink with a hard deadline. If the packet arrives after the deadline, the packet would be completely useless. For example, in a real-time target detection and tracking application, when a target moves into an area monitored by a WSN, it will be detected by some sensor nodes. A report packet that contains the target's information will be sent by a node to the sink node for tracking. As the target moves, a new report packet will be generated because the target will be detected by other sensor nodes. If the old report packet can not be received by the sink node before the target is detected by other sensor nodes and a new report packet is generated, it will be useless because the new report packet contains more recent information (in this example, the deadline depends on the the target's usual speed). The characteristics of event-driven WSN applications are that: (1) they do not have data most of the time, and (2) they have to report the alarm to the sink with real-time constraints. Therefore, the protocols designed for event-driven WSNs should not only be very energy-efficient in order to prolong the lifetime of the network, but also time-constrained.

It is shown in Chapter 2.1.1 that duty-cycle can save a large amount of energy. However, turning off the radio will negatively affect other performances, such as end-to-end delay, connectivity and so on. This raises a problem: how to schedule the wakeup time of the radios to save energy without obviously decreasing other performances. Some existing scheduling algorithms can reduce the end-to-end delay of the network with duty-cycle sensor node [Lu et al., 2004; 2005], but they do not take the unreliable links into consideration. Shorter delays and higher delivery ratios can be obtained if duty-cycling and link unreliability are well-utilized

as done in [Gu and He, 2007]. Although the existing protocol which takes duty-cycling and unreliable links into account can have a higher delivery ratio, the biggest problem is that it does not consider real-time constraint, which is necessary for real-time event-driven WSN applications. On the other hand, some real-time forwarding protocols [Lu et al., 2005; Wakamiya et al., 2009] do not consider the unreliable links.

In the following sections, we will present the proposed two novel forwarding protocols for low duty-cycle WSNs, i.e. synchronous and asynchronous forwarding. The basic metrics used by these two forwarding protocols are EDR (Expected Delivery Ratio) and hop count. The two various protocols use the similar algorithm to initialize the network. After the initialization phase, synchronous forwarding protocol has scheduled the wakeup slots for every node. They can transmit the packets at the right time when the candidate node is awake. Asynchronous forwarding protocol has to wake up the candidate node using preamble sampling technique. The details of two protocols will be described in this chapter.

4.2 Synchronous forwarding

At first, we want to present the synchronous forwarding protocol which will be described by the following subsections: (1) assumptions, (2) Motivation, (3) system model, (4) wakeup scheduling algorithm and (5) performance evaluation.

4.2.1 Assumptions

In the synchronous forwarding protocol, we make the following assumptions:

(1) All nodes are locally synchronized to their neighbors. Some existing synchronization protocols can achieve this [Ganeriwal et al., 2003; Maróti et al., 2004].

(2) Only one node sends the alarm when the event happens. This can be done by using one aggregation method [Liu et al., 2009; Ye et al., 2005].

(3) In real-time applications, the characteristics of the application give the maximum delay requirement. In this protocol, we focus on the protocol's end-to-end delay bound. Of course, the protocol's maximum end-to-end delay must satisfy the application delay requirement. The parameters of our protocol can be adjusted according to the application delay requirement. It is left for future work.

(4) After the alarm packets are aggregated, the node which is chosen to be the reporter will send the packet in its wakeup slot. In this protocol, we only consider the delay from the moment when the original sender starts sending the packet to the time when the packet is received by the sink. The delay from the moment when the event happens to the time when the sender starts sending the packet in its wakeup slot belongs to the

aggregation delay which is not considered in this protocol. By knowing the application delay requirement, we can adjust the parameters of our protocol and choose a suitable aggregation protocol to satisfy the application delay requirement.

(5) One duty-cycle period is divided into many slots which have the same duration. One slot is only long enough to be able to send a *DATA* packet and receive an *ACK* packet. We assume the *ACK* packet from the receiver can be received by the sender if the receiver receives the *DATA* packet successfully. This is reasonable because the size of the *DATA* packet is much greater than the size of the *ACK* packet and the wireless channel has correlation characteristic [Cerpa et al., 2005; Sang et al., 2007]. Although the probability that an ACK packet is successfully received by the sender after the receiver has successfully received the DATA packet is high, in reality, it is not 100 percents. If the ACK packet is lost, there will exist duplicated DATA packets in the network. However, duplicated packets may be good for some applications because redundancy increases reliability. It may be bad for some other applications because it consumes much more energy. Studying the impacts of the duplicated packets is a future work.

(6) Each node wakes up for only one slot during one period and sleeps in the remaining slots.

4.2.2 Motivation

This protocol is motivated by the interesting and basic relationship between the expected delivery ratio (*EDR*) and the wakeup time. In this section, we want to briefly introduce the relationship by a simple example. In Figure 4.1, there are four nodes, *A*, *B*, *C* and *D*, which have wakeup slots WS_A, WS_B, WS_C and WS_D respectively (the definition is listed in Table 4.1). Node *D* is the sink. The data near the lines denotes the success ratio of a round-trip transmission (*DATA* and *ACK*) between two nodes. For example, the success ratio from node *A* to *B* is 0.6. Node A can forward the *DATA* packet to node D through node B or C. In order to have a higher delivery ratio, A will try the next candidate in the forwarding set if the previous transmission fails. The interesting problem is that if we assign different wakeup slots to node *B* and *C*, the *EDR* of node A to the sink is different.

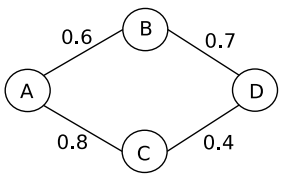


Figure 4.1: Underlying basic example

(1)$WS_A < WS_B < WS_C < WS_D$

In this case, we assume node *B* has an earlier wakeup slot than node *C*. The *EDR* of node *A* will be:

$$\begin{aligned} EDR_A^1 &= P_{AB} \times P_{BD} + (1 - P_{AB}) \times P_{AC} \times P_{CD} \\ &= 0.6 \times 0.7 + 0.4 \times 0.8 \times 0.4 = 0.548 \end{aligned} \tag{4.1}$$

Here, P_{AB} denotes the success ratio between node A and B. The others have similar definitions.

(2)$WS_A < WS_C < WS_B < WS_D$

Here, in contrast, node *C* is assigned a lower wakeup slot than node *B*. In this case, the *EDR* of node *A* is given by:

$$\begin{aligned} EDR_A^2 &= P_{AC} \times P_{CD} + (1 - P_{AC}) \times P_{AB} \times P_{BD} \\ &= 0.8 \times 0.4 + 0.2 \times 0.6 \times 0.7 = 0.404 \end{aligned} \tag{4.2}$$

Table 4.1: Notations and descriptions in WSEDR

Notation	**Description**
α	the link quality threshold for calculating new *HC*
T	the length of the duty-cycle period
SR	the selectable range of wakeup slot for each hop
P_{n_j}	the round-trip success ratio to the neighbor j
WS_i	node i's scheduled wakeup slot
HC_i	node i's hop count
LQ_{n_j}	the link quality from neighbor j
EDR_i	node i's expected delivery ratio to the sink

We can see that EDR_A^1 is larger than EDR_A^2 if we let node *B* wake up before node *C*. From this basic example, we know that there should be some relationships between the *EDR* and the wakeup slots. We will show the relationships in the following sections. We schedule the wakeup slots of all the nodes according to their EDR to the sink. As we have shown in the aforementioned simple example, if a transmission fails, the packet will be transmitted to the following wakeup neighbor. This can be viewed as a multi-parent data forwarding scheme.

Before describing our proposed scheduling algorithm in detail, we want to emphasize that our proposed algorithm is distributed. Every node schedules its own wakeup slots. It is obvious that the delivery ratio will be increased if we use a centralized algorithm to schedule the wakeup slots for the sensor nodes because our proposed distributed algorithm will assign the same slots to the nodes with the similar EDRs. We will show this in Section 4.2.4. However, centralized algorithms are very expensive for large-scale WSNs because the sink node must collect the information of all the sensor nodes, then construct a schedule, then broadcast the schedule

to all the sensor nodes. As a result, we consider it is not realistic, for energy efficiency, to implement centralized algorithms in large-scale WSNs.

4.2.3 System Model

In this forwarding protocol, when one node has packets to send, it tries to send the packet to the neighbor that wakes up first in the forwarding set. If the first transmission fails, it will try to send the packet to the candidate that wakes up next. The packet will be discarded if it is not successfully transmitted to any neighbors in the forwarding set. For example, in Figure 4.2, node *A* only includes nodes *D*, *E* and *F* into its forwarding candidate set. Node A first tries to send the packet to node D. If this transmission fails, node A will send the packet to node E. If all the transmissions to the three nodes fail, node A will give up on the packet. In our model, each node *i* has the tuple (HC_i, EDR_i, WS_i). The exact definitions of each label are given in Table 4.1.

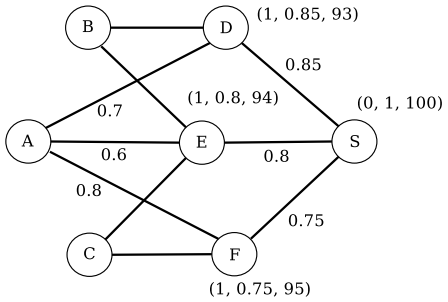


Figure 4.2: A simple example of our model

Let F_i denote the forwarding candidate set of node i. Let $\pi_n(F_i) = \langle i_{n_1}, i_{n_2}, \ldots, i_{n_r} \rangle$ be one permutation of nodes in F_i, the numbering indicates the sequence in which the nodes will wake up. For each node, we can get the *EDR* of node i with the forwarding set $\pi_n(F_i)$ with Equation (4.3).

$$EDR(\pi_n(F_i)) = \sum_{k=1}^{r} EDR_{n_k} P_{n_k} \prod_{j=0}^{k-1} (1 - P_{n_j}) \tag{4.3}$$

This mathematical model has the following two important properties:

Property 1: *$EDR(\pi_n(F_i))$ can reach the maximum if and only if the following condition is satisfied:*

$$\begin{cases} EDR_{n_1} > EDR_{n_2} \cdots > EDR_{n_r} \\ WS_{n_1} < WS_{n_2} \cdots < WS_{n_r} \end{cases} \tag{4.4}$$

Proof. First, when $r = 1$, property 1 holds. Next, we assume it holds for $r = N(N \geq 1)$. When $r = N+1$, we assume the $(N+1)th$ node has EDR_{n_b} and P_{n_b} in which EDR_{n_b} denotes

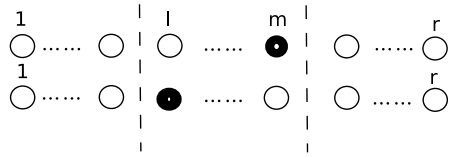


Figure 4.3: Example sequence for property 1

the node's expected delivery ratio to the sink and P_{n_b} is defined as the round trip success ratio between this node and the sender. We also assume that the right place for $(N+1)th$ node in the forwarding sequence is m according to the EDR_{n_b}. Next, we have to prove the maximum *EDR* of the sender can only be obtained if the $(N+1)th$ node is in the mth place. The *EDR* of each node in the forwarding set satisfies the following inequality.

$$\begin{cases} EDR_{n_b} < EDR_{n_k} & \text{if } k < m \\ EDR_{n_b} > EDR_{n_k} & \text{if } k > m \end{cases} \tag{4.5}$$

When the $(N+1)th$ node is placed at the mth place (as shown in the above row of Figure 4.3), the *EDR* of node i is given by:

$$EDR^{m}_{\pi_n(F_i)} = \sum_{k=1}^{r} EDR_{n_k} P_{n_k} \prod_{j=0}^{k-1} (1 - P_{n_j}) \tag{4.6}$$

In order to prove the mth place is the best one for the $(N+1)th$ node, we divide our proof process into two parts: (1) put it anywhere before m, (2) put it anywhere after m. If the *EDR* derived from both is lower than that derived from the mth place, we can conclude that the mth place is the best place for $(N+1)th$ node.

(1) Place the node anywhere before m.

If we do not put the $(N+1)th$ node at the mth place, instead of putting it anywhere before m, say l, as shown in the line below in Figure 4.3, we can calculate the *EDR* of node i by Equation (4.7).

$$\begin{aligned} EDR^{l}_{\pi_n(F_i)} \;=\; & \sum_{k=1}^{l-1} EDR_{n_k} P_{n_k} \prod_{j=0}^{k-1} (1 - P_{n_j}) \\ & + \prod_{k=0}^{l-1} (1 - P_{n_k}) [EDR_{n_b} P_{n_b} + (1 - P_{n_b}) \sum_{k=l}^{m-1} EDR_{n_k} P_{n_k} \prod_{j=l}^{k-1} (1 - P_{n_j})] \\ & + \sum_{k=m+1}^{r} EDR_{n_k} P_{n_k} \prod_{j=0}^{k-1} (1 - P_{n_j}) \end{aligned} \tag{4.7}$$

The difference between the two methods, called δ, is given by Equation (4.8).

$$\begin{aligned}\delta &= EDR^{m}_{\pi_n(F_i)} - EDR^{l}_{\pi_n(F_i)} \\ &= \prod_{k=0}^{l-1}(1-P_{n_k})P_{n_b}\sum_{j=l}^{m-1}(EDR_{n_j}-EDR_{n_b})P_{n_j}\prod_{h=l}^{j-1}(1-P_{n_h}) \end{aligned} \tag{4.8}$$

Because for all $j < m$, $EDR_{n_j} > EDR_{n_b}$, then $\delta > 0$, hence, we can get $EDR^{m}_{\pi_n(F_i)} > EDR^{l}_{\pi_n(F_i)}$.

(2) Place the node anywhere after m.

The method to prove this is the same as in the previous section.

From the results in the two cases, we can see that Property 1 holds for $r = N + 1$. Thus, we can conclude that Property 1 holds for any r ($r \geq 1$). □

This property means that the maximum *EDR* of each node can be obtained by assigning the lower wakeup slot to the candidate node in its forwarding set whose *EDR* to the sink is higher. For example, in Figure 4.2, node *D* has the smallest wakeup slot in *A*'s forwarding set because it has the biggest *EDR* to the sink (Node *D*, *E* and *F* have *EDR* 0.85, 0.8 and 0.75 respectively, and they have *WS* 93, 94 and 95 respectively).

Property 2: *$max(EDR(\pi_n(F_i)))$ is a strictly increasing function of the size of the forwarding set.*

Proof. Assume we have $r = N$ nodes in the forwarding set. According to property 1, we know that we can have the maximum *EDR* if we give the node with the greatest *EDR* to the sink the highest priority. Now we assume that the r nodes are well-arranged (as shown in the upper row of Figure 4.4). We add another $(N+1)th$ node in the forwarding set. Assume we have to put this node into the mth place according to property 1. The $(N+1)th$ node has EDR_{n_b} and P_{n_b}. Now we will prove that the maximum value of the *EDR* increases after adding this node.

The maximum *EDR* of node i with N nodes in the forwarding set is given by:

$$max(EDR^{N}_{\pi_n(F_i)}) = \sum_{k=1}^{r} EDR_{n_k}P_{n_k}\prod_{j=0}^{k-1}(1-P_{n_j}) \tag{4.9}$$

After adding the $(N+1)th$ node, the maximum *EDR* of node i is given by:

$$\begin{aligned} max(EDR^{N+1}_{\pi_n(F_i)}) &= \sum_{k=1}^{r} EDR_{n_k}P_{n_k}\prod_{j=0}^{k-1}(1-P_{n_j}) + \prod_{h=0}^{m-1}(1-P_{n_h})[EDR_{n_b}P_{n_b} \\ &\quad + \sum_{k=m}^{r} EDR_{n_k}P_{n_k}\prod_{j=m}^{k-1}(1-P_{n_j})(1-P_{n_b})] \end{aligned} \tag{4.10}$$

The difference between the two, called δ, is

$$\begin{aligned}
\delta &= max(EDR^{N+1}_{\pi_n(F_i)}) - max(EDR^{N}_{\pi_n(F_i)}) \\
&= \prod_{h=0}^{m-1}(1-P_{n_h})P_{n_b}[EDR_{n_b} - \sum_{i=m}^{r} EDR_{n_i}P_{n_i}\prod_{j=m}^{i-1}(1-P_{n_j})] \\
&\geq \prod_{h=0}^{m-1}(1-P_{n_h})P_{n_b}[EDR_{n_b} - EDR_{n_m}\sum_{i=m}^{r} P_{n_i}\prod_{j=m}^{i-1}(1-P_{n_j})] \\
&> \prod_{h=0}^{m-1}(1-P_{n_h})P_{n_b}(EDR_{n_b} - EDR_{n_m}) \\
&> 0 \qquad (4.11)
\end{aligned}$$

Thus, we can see the maximum *EDR* is increased by adding one node in the forwarding set. Obviously, adding more than one node in the forwarding set will further increase the maximum *EDR*. □

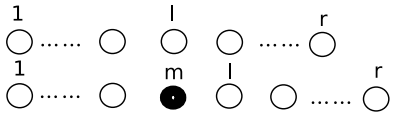


Figure 4.4: Example sequence for property 2

This property means that the maximum *EDR* of each node will increase if they have more candidates in their forwarding set. For example, in Figure 4.2, if node *A* adds *D*, *E* and *F* to its forwarding set, the maximum *EDR* is 0.811. However, if node *A* only adds *D* and *E* to its forwarding set, the maximum *EDR* is 0.739.

This model shows that each node can have the maximum *EDR* if the nodes in their forwarding set with the greater *EDR* have the earlier wakeup time. The problem lies in constructing the forwarding set for each node and scheduling each node's wakeup time. Moreover, for some time-critical applications, the delivery ratio is not the only requirement. To have a bounded delay is another important objective. In the next section, we will introduce the proposed wakeup scheduling algorithm which can have a bounded delay and higher delivery ratio by using the model describing in this section.

4.2.4 Wakeup scheduling algorithm

Proposition

Before introducing the wakeup scheduling algorithm, we want to describe the structure of our forwarding scheme. It includes two phases: initialization and run-time. In the initialization phase, each node first gets the neighborhood table and the link quality between them and their neighbors, and then executes the wakeup scheduling algorithm. After the initialization phase, we have the run-time phase where each node is awake for one slot and asleep in the remaining

slots in one period if they do not have a packet to forward. Any node that has sensed the event will forward the packet to the neighbor in the forwarding set.

$$HC_i^{n+1} = \begin{cases} HC_{n_j} + 1 & if\ HC_i^n > HC_{n_j} + 1\ \ \&\& \\ & LQ_{n_j} \geq \alpha \\ HC_i^n & otherwise \end{cases} \tag{4.12}$$

We want to describe the two Equations used to calculate the hop count and wakeup slot in the scheduling algorithm. In Equation (4.12), HC_i^n and HC_i^{n+1} denote the hop counts of node i in n^{th} step and $(n+1)^{th}$ step respectively (each node may change the hop count for many times). HC_{n_j} denotes the hop count of the j^{th} neighbor in the forwarding set. LQ_{n_j} is defined as the link quality from neighbor j. We set a threshold α to decide if the node needs to change the hop count. The reason why we add this threshold on the link quality is because it is not useful to change the hop count when the link quality is poor.

$$WS_i = (T - SR \times HC_i) + SR \times (1 - EDR_i) \tag{4.13}$$

In Equation (4.13), $SR = T/HC_{upbound}$, where $HC_{upbound}$ denotes the hop count bound of the network. The work which studies the relationship between the hop count and the distance can be found in [Dulman et al., 2006]. In our protocol, we do not need to know the exact maximum hop count, but the more precise the maximum hop count is, the higher delivery ratio the protocol will have. Further, it is difficult to get the precise maximum hop count in a distributed way in WSNs, so we use an approximate value. Other definitions can be found in Table 4.1. Each node calculates the wakeup slot according to the hop count and *EDR*. If the total number of slots is small, there will be some neighbors having the same wakeup slots. This will cause the decrease of the average delivery ratio.

In the scheduling algorithm, we set the hop count and wakeup slot of each node according to Equations (4.12) and (4.13) respectively. If a node receives a packet from one of its neighbors, it will decide if it has to change the hop count according to Equation (4.12). After the node gets the hop count, it will calculate the *WS*. The key idea is to choose a slot in the total period *T* according to the hop count and *EDR*. For example, if a node has hop count h, it will choose the *WS* from the following range, [$T - SR \times h$, $T - SR \times (h-1)$). We want to give a simple example in Figure 4.5 to show the basic idea of choosing the wakeup slot. In this example, $SR = 4slots$ and $T = 5 \times SR = 20slots$. If a node has hop count 3, it will choose the wakeup slots from [8, 12).

The distributed scheduling algorithm for each node other than the sink is described in Algorithm 1. The initialization values of the sink node is $(0, 1, T)$ (it means that the hop count of the sink is 0 and its *EDR* is 1 and it has the T^{th} slot). First, the sink broadcasts its *EDR*, *WS* and *HC* values. The nodes that receive the broadcast message from their neighbors calculate the *HC* and *EDR*. Then, they get the *WS* according to the calculated *EDR* and *HC*. If the change

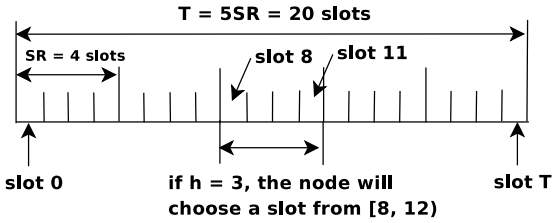


Figure 4.5: A simple example of choosing the wakeup slot

Algorithm 1: Distributed Wakeup Scheduling Algorithm

1: Sink broadcasts a packet that includes its *EDR*, *WS* and *HC*.
2: Every node except the sink runs the following algorithm.
3: **if** receives a packet from one of the neighbors
4: Calculates the new *HC* with Equation (4.12)
5: Calculates the new *EDR* with Equation (4.3)
6: **if** the change of *EDR* is higher than a threshold **or** the *HC* is changed
7: Calculates the new *WS* with Equation (4.13)
8: Broadcasts the new values
9: **endif**
10: **endif**

of the new *EDR* exceeds a certain value or the *HC* is changed, the node will rebroadcast the new values. It must be noted that each node only adds the neighbors with lower hop counts into its forwarding set during the initialization phase. This is because the initialization process will be an endless loop and can not converge if we add the neighbors with the same hop count into the forwarding set during the initialization phase.

After getting the hop count and wakeup slots, each node has to decide which node should be included in the forwarding set during the run-time phase. In our forwarding scheme, two methods can be considered as follows:

(1) add only the nodes with lower hop count to the forwarding set during the run-time phase, as we did in the initialization phase. With this method, all the nodes in the forwarding set satisfy inequality 4.14 and it is obvious that this method has the maximum delay $HC \times SR$ which is bounded by T.

$$\begin{cases} HC_i > HC_{n_j} \\ WS_i < WS_{n_j} \end{cases} \tag{4.14}$$

(2) add the nodes with the same and lower hop counts to the forwarding set during the run-time phase. With this method, the node first sends the packet to the nodes with lower hop counts. If all the nodes fail, it will try to send the packet to the nodes with the same hop count. This increases the average delivery ratio of each node if the link quality of the nodes with lower hop counts is low. In order to bound the maximum delay, in this situation, we simply let each packet be sent to the node with the same hop count once. For example, in Figure 4.6, node A, C and D have hop number i, node B has hop number

$i-1$. Node A has a packet to forward. It first sends to node B. If it fails, node A will send the packet to node C. After node C receives the packet, it will send the packet to node B. In this case, if node C fails to send the packet to node B, it will not try to send to node D because we want to bound the delay as shown in Equation (4.15). In fact, we can let node C try to send the packet to node D. The difference is that we will have much greater maximum delay and higher delivery ratio. There exists a trade-off between them.

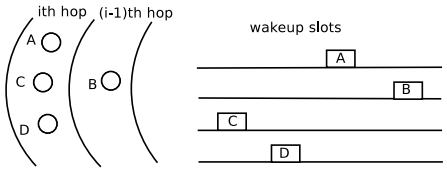


Figure 4.6: Example for explaining maximum delay

Property 3: Using method 2, we can have the maximum delay as described in Equality 4.15 (HC_{max} denotes the maximum number of hops from any node to the sink, $Delay_{max}$ is defined as the maximum delay of the network).

$$Delay_{max} = (T + 3SR) \times HC_{max} \tag{4.15}$$

Proof. When a packet is sent directly from hop count i to $i-1$ (for example, in Figure 4.6, a packet is sent from node A to B), the maximum delay is $2SR$. Similarly, when a packet is sent from hop count i to i (the packet is sent from node A to C), the maximum delay is SR. If a packet cannot be directly sent from hop count i to $i-1$, instead of being sent from hop count i to i, and then from hop count i to $i-1$ (Node A fails to send packet to B. Then, node A tries to send the packet to node C. If node C receives the packet from node A, node C transmits the packet to node B). In this case, the maximum delay is $T + 3SR$. As a result, we can get the maximum delay from hop count i to $i-1$ is $T + 3SR$, and any node with hop count i has the maximum delay $T + 3SR + Delay_{i-1}$. Thus, we can get the maximum delay of any node with hop count i as follows,

$$\begin{aligned} Delay^{i}_{max} &= T + 3SR + Delay^{i-1}_{max} \\ &= \underbrace{T + 3SR + \cdots + T + 3SR}_{i} \\ &= (T + 3SR) \times i \end{aligned} \tag{4.16}$$

□

Discussion on WSEDR

As we can see from the Equation (4.13), if two nodes have very close EDR, the calculated wakeup slots may be the same. In this subsection, we want to compare the performance of WSEDR with the performances of an ideal centralized solution (impossible to implement in real sensor networks) OPT and a solution with randomness (to avoid some nodes with very close EDR to have the same wakeup slot) WSEDR-random. In the following subsection, the details of those two solutions will be presented.

—OPT

With this solution, we want to compare WSEDR to a solution which assures each node has its own slot and all the nodes have different wakeup slots. The problem of implementing this kind of solution is that it is necessary to know the EDRs of all the nodes and schedule the wakeup slots for them. It is possible to do this in a simulator but not realistic in real sensor networks. Because we can easily know the information of all the sensor nodes in the simulator, we want to show the performances of an optimized solution which is called OPT. The details of implementing OPT is as follow: first, each hop is assigned the same number of slots. We sort the nodes with the same hop count according their EDRs. Then, the node with the greatest EDR will be assigned the lowest wakeup slot. The node with the second greatest EDR will be assigned the second lowest wakeup slot. If the total number of slots are great enough, all the nodes could be assigned different slots.

—WSEDR-random

The main idea is to add randomness to avoid some nodes with very close EDR to choose the same slot. For that, in this solution, we further divide the slots within a SR into ten groups. Each group has 10 slots. The hop count determines the chosen SR and the value of EDR determines the chosen group. Then those nodes which choose the same group can randomly choose slots inside the group. For example, the nodes with EDR between 0.9 and 1 will be allocated in the first group. Accordingly, the nodes with EDR between 0 and 0.1 will be allocated in the last group. For the nodes that have the similar EDRs, say 0.98 and 0.97, they will both be allocated in the first group and will randomly choose a slot in the first group. The formula of WSEDR-random is shown in Equation (4.17) which is used to add a random value. By doing this, the delivery ratio may be different. Will the delivery ratio be increased or decreased? This will be shown in the following simulation results.

$$WS_i = (T - SR \times HC_i) + \lfloor (1 - EDR_i) \times 10 \rfloor \times SR/10 + \lfloor random() \times SR/10 \rfloor \quad (4.17)$$

—Simulation results

We have simulated the proposed algorithm and the aforementioned two solutions using WSNet [1]. The basic simulation parameters are listed in Table 3.2 (Chapter 3.2.5). In each experiment, we let every node send a packet to the sink. Each result is the average value of 100 experiments run with different topologies. The 95% confidence intervals are within $1 \sim 10\%$ of the means.

In the simulations, we deploy 250 nodes in a $150m \times 150m$ area. We increase the total number of slots in a period, which means we increase the length of T (the greater the T is, the lower the duty-cycle will be). Because *SR* is proportional to T, *SR* is also increased. The length of T is increased from 750, 1000, 2000 to 3000 slots, which leads to the result that the length of *SR* is increased from 75, 100, 200 to 300. The simulation results are shown in Table 4.2. *WSEDR-random* represents the method in Equation (4.17) while *WSEDR* indicates the scheme in Equation (4.13).

Table 4.2: Average delivery ratios of *WSEDR*, *WSEDR-random* and OPT

Number of slots	750	1000	2000	3000
OPT	0.655	0.66	0.652	0.659
WSEDR	0.523	0.526	0.533	0.548
WSEDR-random	0.517	0.525	0.524	0.533

It is shown in Table 4.2 that the average delivery ratios of WSEDR and WSEDR-random are increased when we increase the total number of slots. This is because the number of nodes that have the same wakeup slot decreases as the number of total slots is increasing. The average delivery ratio of WSEDR is higher than that of WSEDR-random because WSEDR gives the node with a greater EDR higher priority. It is also shown in Table 4.2 that the average delivery ratio of OPT is higher than both two other solutions because OPT has different slots for all the neighboring nodes. The improvement of OPT is about 0.12. However, compared to the fact that it is not realistic to implement OPT in large-scale WSNs, the increment is marginal. In summary, OPT has greatest delivery ratio but it is not realistic. WSEDR-random does not have better delivery ratio although it is also a distributed solution. WSEDR is a good trade-off between the two. So we will study the performances in detail in the following section.

4.2.5 Performance evaluation

In the following sections, we will compare the performances of the following two algorithms in terms of average delivery ratio, average end-to-end delay, and average energy consumption. Delivery ratio corresponds to the number of packets received by the sink divided by the number of packets sent by the senders. End-to-end delay corresponds to the difference between the time when the packet is received by the sink and the time when the original sender transmits

[1] http://wsnet.gforge.inria.fr/

the packet. Here, energy consumption is measured by the average number of times a packet is sent until it is received by the sink. The key point of our algorithm is to schedule the wakeup time according to the hop count and *EDR* to the sink. To the best of our knowledge, there are no other algorithms which consider ultra-low duty-cycle, unreliable links and bounded delay simultaneously. We choose to compare our algorithm with the one which randomly schedules the wakeup time to show the improvements.

(1) **WSEDR**: the proposed distributed **W**akeup **S**cheduling algorithm according to the **E**xpected **D**elivery **R**atio to the sink.

(2) **Random**: every node randomly gets the wakeup slot according to the hop count, as shown in Equation (4.18). *random()* denotes a random value between 0 and 1.

$$WS_i = (T - SR \times HC_i) + SR \times random(). \tag{4.18}$$

In our simulations, we set the threshold described in Algorithm 1 to a very small value (10^{-6}). We also let all the nodes broadcast at least ten times their parameters in the wakeup scheduling algorithm in order to make the calculated *HC* and *WS* more accurate.

We have conducted the following two kinds of experiments:

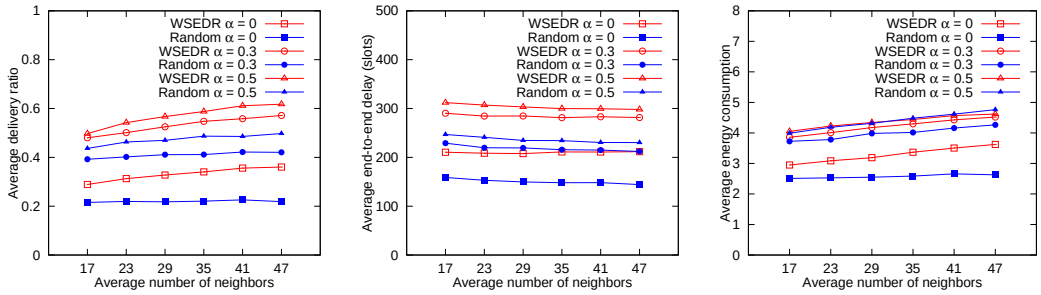
(1) Experiment 1: In this experiment, we have implemented the method 1 described in Section 4.2.4. Each node just considers the neighboring nodes with the lower hop count.

(2) Experiment 2: This experiment implements the method 2 depicted in Section 4.2.4. Every node considers the neighboring nodes with not only the lower hop count but also the same hop count.

Simulation results of experiment 1 (only lower hop count)

Impacts of density and α

First, we evaluate the performances of *WSEDR* under different node density and α (defined in Table 4.1). We set the duty-cycle to 0.1% (the total number of slots is 1000. If there are no packets to forward, each node is awake for 1 slot and asleep in the remaining slots. If one node has a packet to forward, it will be awake for more than 1 slot). The size of the area is $150m \times 150m$. The sink is located at (75, 75). We vary the number of nodes to get different node densities. The number of nodes is changed from 150, 200, 250, 300, 350 to 400. Because of the unreliable links, one node sometimes can receive packets from some neighbors that are far away. In this simulation, we only count the neighbors that have the link quality which is higher than 0.2. As a result, the average number of neighbors with the aforementioned number of nodes is from 17, 23, 29, 35, 41 to 47. The value of α is changed from 0 to 0.5.



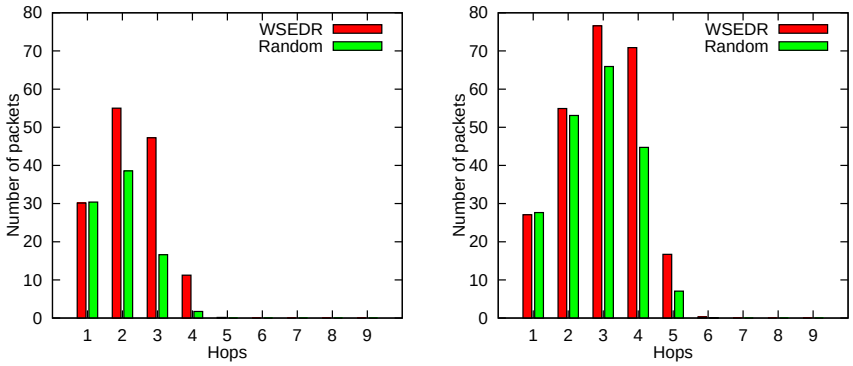
(a) Expected delivery ratio (b) Expected end-to-end delay (c) Expected energy consumption

Figure 4.7: Impacts of density and α in experiment 1

In Figure 4.7(a), we can see that the average delivery ratio of *WSEDR* increases with both the density and α, but the average delivery ratio of *Random*, which has a small variance when the number of nodes is changed, increases only with α. With the same α, the average delivery ratio of *WSEDR* is higher than that of *Random*. The reason why the average delivery ratio of *WSEDR* increases with the density is that each node will have more candidates if the density is higher. The more candidates a node has, the higher delivery ratio it will have. This simulation result is in line with Property 2. The reason why the average delivery ratio of *Random* has a very small variance and is always smaller than the average delivery ratio of *WSEDR* is because *Random* does not take link qualities into consideration. Although one node has more candidates in *Random*, the delivery ratio can not be increased a lot due to the randomness.

In Figure 4.7(b), the average end-to-end delay (counted in slots) of both two schemes is almost not changed no matter how the density changes, but increases with α. Additionally, the average end-to-end delay of *WSEDR* is a little greater than that of *Random*. The average end-to-end delay of both two schemes is bounded by *T*. In Figure 4.7(c), we can see that the average energy consumption of both two algorithms is increasing with the density and α, but the increments are small when α is changed from 0.3 to 0.5. The average energy consumption of *WSEDR* is higher than that of *Random* when α is set to 0 and 0.3, but it is almost the same as that of *Random* when α is set to 0.5.

We want to explain Figure 4.7(b) and Figure 4.7(c) using Figure 4.8 where the hop distributions of received packets are plotted. We can see from Figure 4.8 that both schemes transmit more packets from the higher hop count to the sink when we change α from 0 to 0.5. This is the reason why the average end-to-end delay and energy consumption of both schemes are increasing with α. The reason why *WSEDR* has a higher end-to-end delay and energy consumption than *Random* is because *WSEDR* transmits more packets with a higher hop count to the sink than *Random*. For example, 30.1, 54.9, 47.2 and 11.2 packets are received with *WSEDR* from hop 1, 2, 3 and 4 respectively while 30.3, 38.5, 16.6 and 1.71 packets are received with *Random* when α is set to 0.

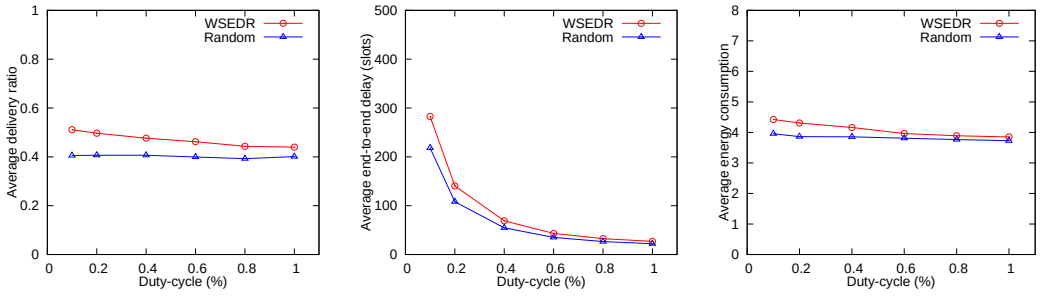


(a) 400 nodes, α = 0 (b) 400 nodes, α = 0.5

Figure 4.8: Hop distribution of the received packets in experiment 1 with different α

Impacts of duty-cycle

In this section, we evaluate the performances of *WSEDR* by changing the duty-cycle. We set α to 0.3 and deploy 250 nodes in a $150m \times 150m$ area, and change the duty-cycle from 1% to 0.1% (for all these settings, every node is awake for 1 slot and asleep in the remaining slots. We change the total number of slots to get the different duty-cycle. In order to increase the duty-cycle, we decrease the length of the period. For example, for a given slot duration, if the duty-cycle is 0.1%, every node will wake up for one slot and sleep for 999 slots. If we want to increase the duty-cycle to 1%, every node will wake up for one slot and sleep for 99 slots. The length of the period is decreased from 1000 slots to 100 slots). The sink node is located at (75, 75). As described in the mathematical model (Property 2), if we have more nodes in the candidate set, the *EDR* will be increased. If we keep the density constant, then the lower the duty-cycle is, the lower the probability that two nodes will have the same wakeup slot. And if the number of neighbors that have the same wakeup slot is small, each node will have higher delivery ratio.



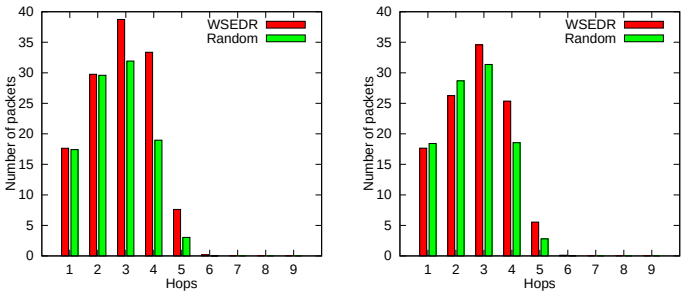
(a) Expected delivery ratio (b) Expected end-to-end delay (c) Expected energy consumption

Figure 4.9: Impacts of duty-cycle in experiment 1

In Figure 4.9(a), we can see that the average delivery ratio of *WSEDR* is higher than that of

Random whatever the duty-cycle is. This simulation result is in line with Property 1. The average delivery ratio of *WSEDR* increases with the decrease of the duty-cycle while the average delivery ratio of *Random* is almost the same. The reason is that every node has more candidates if the duty-cycle is lower, and more candidates lead to a higher delivery ratio for *WSEDR*. This is in line with Property 2.

Figure 4.9(b) shows that the average end-to-end delay of *WSEDR* is a little higher than that of *Random* when the duty-cycle is set to 0.1%. As the increase of the duty-cycle, the average end-to-end delay of both two is becoming closer and closer. The maximum delay of all our simulations is bounded as we described in Section 4.2.4. In Figure 4.9(c), it appears that the average energy consumption of *WSEDR* is a little higher than that of $Random$ when the duty-cycle is 0.1% while they have almost the same energy consumption when the duty-cycle is set to 1%.



(a) duty-cycle 0.1% (b) duty-cycle 1%

Figure 4.10: Hop distribution of the received packets in experiment 1 with different duty-cycle

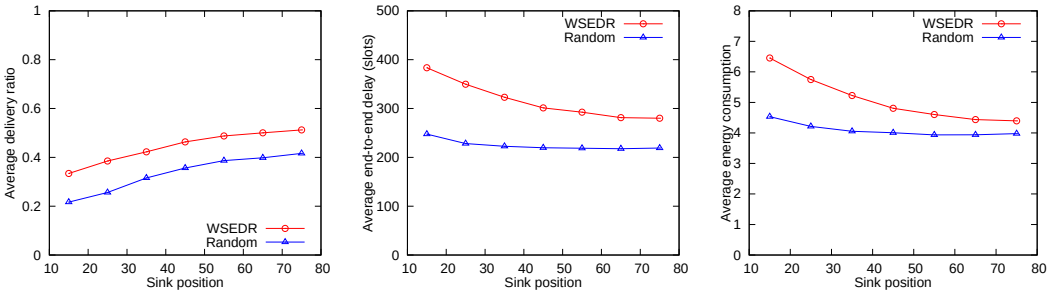
As shown in Figure 4.10, *WSEDR* transmits much more packets from the nodes with the higher hop count to the sink when the duty-cycle is set to 0.1%. This is the reason why the average end-to-end delay and energy consumption of *WSEDR* are much higher than that of *Random*.

Impacts of sink position

In this section, we evaluate the impacts of sink position. We keep the number of nodes at 250, duty-cycle of 0.1%, set the value of α to 0.3 and change the sink position from $(15, 15)$ to $(75, 75)$.

In Figure 4.11(a), we can see that the average delivery ratio of *WSEDR* is higher than that of *Random*. The trends of the average delivery ratio of the two schemes are the same, increasing as the sink moves toward the center. The reason is that the number of hops is decreasing if the sink is moving toward the center. The greater the hop count is, the lower the average delivery ratio will be.

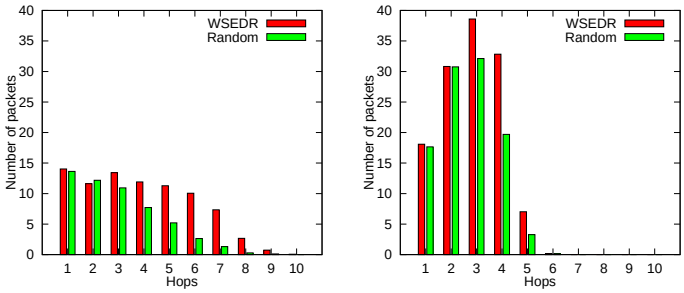
Figure 4.11(b) and 4.11(c) show that the average end-to-end delay and energy consumption



(a) Expected delivery ratio (b) Expected end-to-end delay (c) Expected energy consumption

Figure 4.11: Impacts of sink position in experiment 1

of *WSEDR* are higher than that of *Random*. The differences are greater if the sink is near the corner.



(a) Sink position (15, 15) (b) Sink position (75, 75)

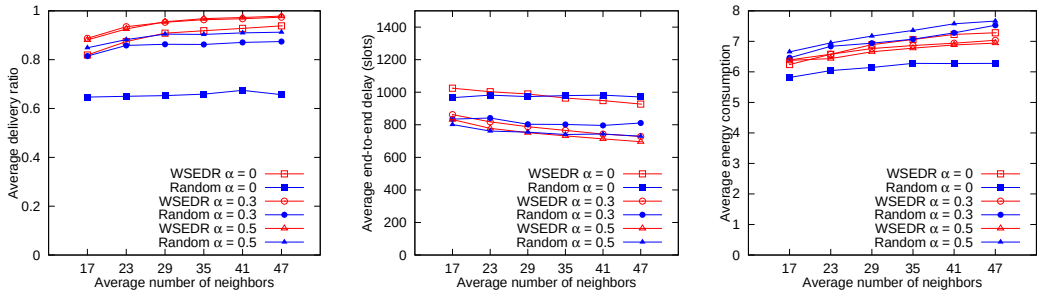
Figure 4.12: Hop distribution of the received packets in experiment 1 with different sink position

We want to explain the reasons of the differences in Figure 4.11 by Figure 4.12 where the hop distribution of the received packets are plotted. We can see that *WSEDR* transmits much more packets from the nodes with the higher hop count to the sink when the sink node is located at (15, 15). This is the reason why the average end-to-end delay and energy consumption of *WSEDR* is much higher than that of *Random* when the sink is near the corner.

From the simulation results of experiment 1, we can see that the average delivery ratio is not very high. In the next section, another method to increase the delivery ratio by using nodes with the same hop count will be presented.

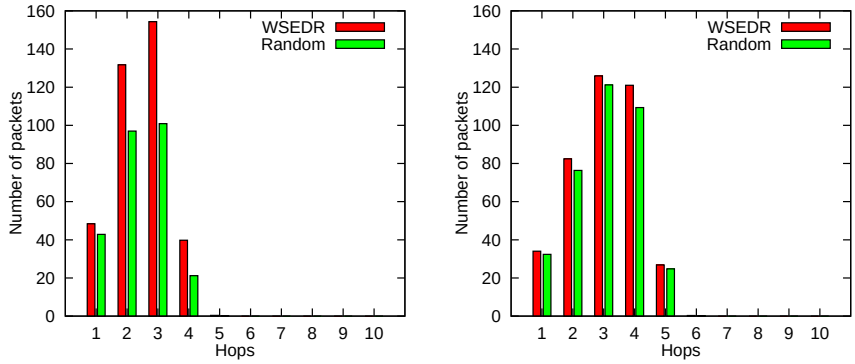
Simulation results of experiment 2 (both the lower and same hop counts)

In the previous sections, it is shown that *WSEDR* has a higher average delivery ratio than *Random*. But the average delivery ratio of both schemes is not very high (about 60% when the density is high). In this section, we will show that the average delivery ratio of our algorithm can be further increased, but the maximum delay is increased as well. The simulation parameters are set as the same as those used in the simulations of experiment 1. So we just present the



(a) Expected delivery ratio (b) Expected end-to-end delay (c) Expected energy consumption

Figure 4.13: Impacts of density and α in experiment 2



(a) 400 nodes, α = 0 (b) 400 nodes, α = 0.5

Figure 4.14: Hop distribution of the received packets in experiment 2 with different α

simulation results of experiment 2 without describing the detailed simulation parameters again.

Impacts of density and α

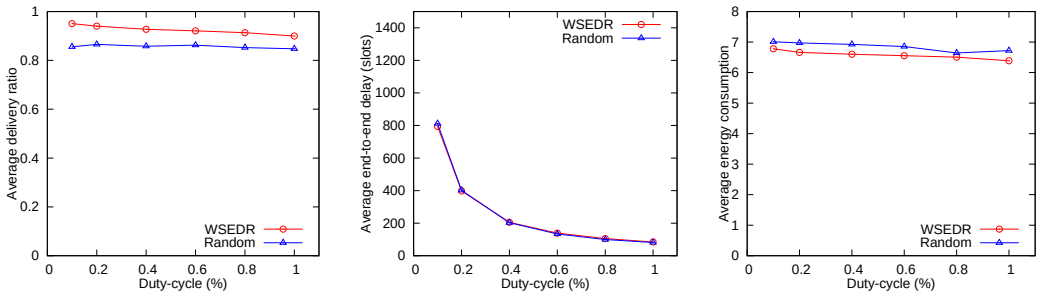
In Figure 4.13(a), we can see that the average delivery ratio of *WSEDR* is higher than that of *Random* no matter what the α and density are. The average delivery ratio of both two increases with the increase of α, but the increments when α is changed from 0.3 to 0.5 are less than the increments when α increases from 0 to 0.3.

It is shown in Figure 4.13(b) that the average end-to-end delay of both two decreases with the increase of α. The average end-to-end delay of *WSEDR* is higher than that of *Random* when the number of nodes is 150. As the increase of the number of nodes, the average end-to-end delay of *WSEDR* is decreasing. It is smaller than that of *Random* when the number of nodes is 400. In Figure 4.13(c), we can see the average energy consumption of *WSEDR* is greater than that of *Random* when the value of α is 0, but it is lower than that of *Random* when the value of α is 0.3 and 0.5.

It is shown in Figure 4.14(a) that *WSEDR* transmits much more packets from the nodes with the greater hop count to the sink. This is the reason why the average energy consumption

of *WSEDR* is greater than that of *Random* when α is set to 0. In Figure 4.14(b), we can see that *WSEDR* just transmits a little bit more packets from the nodes with the greater hop count, which leads to the results that both two schemes have the higher delivery ratio and the delivery ratio difference between the two schemes is not as greater as the difference in the case where α is set to 0.

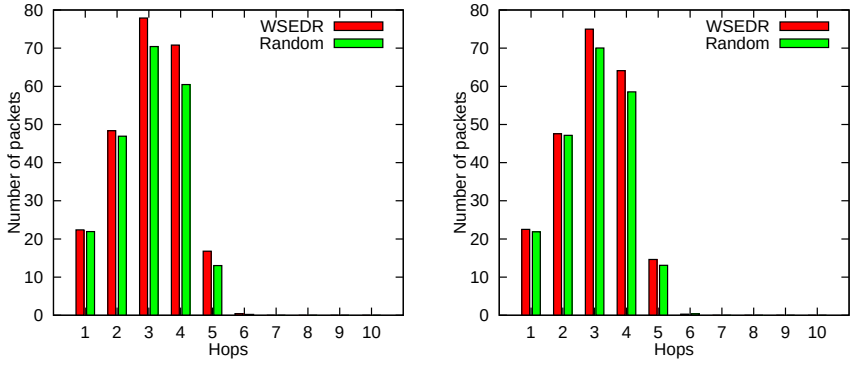
Impacts of duty-cycle



(a) Expected delivery ratio (b) Expected end-to-end delay (c) Expected energy consumption

Figure 4.15: Impacts of duty-cycle in experiment 2

In Figure 4.15(a), we can see that the average delivery ratio of *WSEDR* is higher than that of *Random* in all cases. The difference is greater when the duty-cycle is lower. It is shown in Figure 4.15(b) that the average end-to-end delay of both two is very close. As shown in Figure 4.15(c), the average energy consumption of *WSEDR* is a little lower than that of *Random*. From these results we can see the advantages of *WSEDR*, having the higher delivery ratio as well as the lower energy consumption.



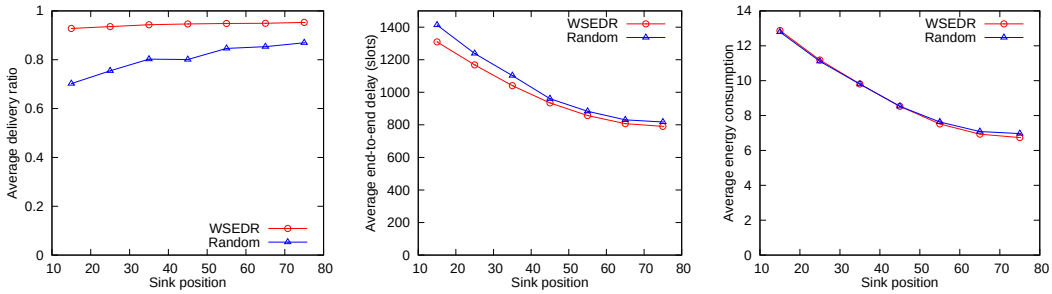
(a) duty-cycle 0.1% (b) duty-cycle 1%

Figure 4.16: Hop distribution of the received packets in experiment 2 with different duty-cycle

In Figure 4.16, we show the hop distribution of the packets received by the sink node.

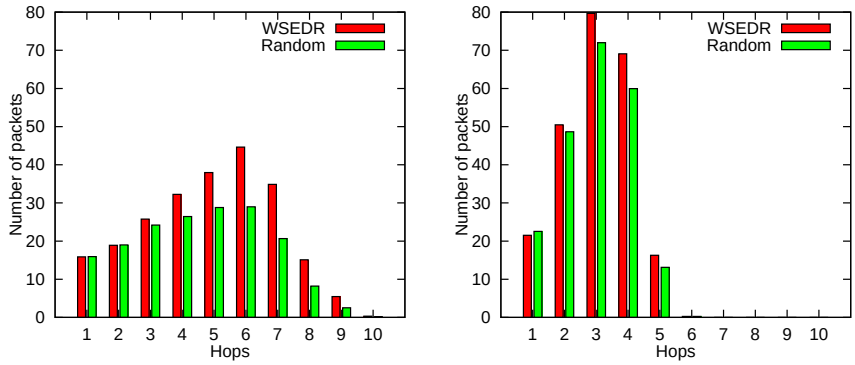
The number of the transmitted packets of both two schemes are very close. This is the reason why the average delivery ratio, end-to-end-delay and energy consumption are very close. The differences of the transmitted packets between *WSEDR* and *Random* when the duty-cycle is 0.1% are greater than the differences when the duty-cycle is set to 1%. This leads to the result that the difference of the average delivery ratio between *WSEDR* and *Random* is a little greater when the duty-cycle is set to 0.1%.

Impacts of sink position



(a) Expected delivery ratio (b) Expected end-to-end delay (c) Expected energy consumption

Figure 4.17: Impacts of the sink position in experiment 2



(a) Sink position (15, 15) (b) Sink position (75, 75)

Figure 4.18: Hop distribution of the received packets in experiment 2 with different sink position

In Figure 4.17(a), it is shown that the average delivery ratio of *WSEDR* is much higher than that of *Random*. As the sink approaches the corner, the average delivery ratio difference between the two increases. From this figure, we can see that *WSEDR* will have much better performance if the size of the network is large.

In Figure 4.17(b), we can see that the average end-to-end delay of *Random* is a little higher than that of *WSEDR*. The difference is greater when the sink is near the corner. Figure 4.17(c)

shows that *WSEDR* consumes a little more energy than *Random* when the sink is deployed near the corner, and consumes less energy when the sink node is near the center.

In Figure 4.18, we show the hop distributions of two simulations which have the sink nodes located at (15, 15) and (75, 75) respectively. *WSEDR* transmits much more packets from the higher hop count to the sink when the sink is located at (15, 15). This is the reason why the delivery ratio and delay differences are greater.

Table 4.3: Maximum delay and Maximum hop count

Sink position		(15, 15)	(25, 25)	(35, 35)	(45, 45)	(55, 55)	(65, 65)	(75, 75)
WSEDR	Maximum hop count	10	9	9	8	8	6	6
	Maximum delay	5720	4796	5507	4556	4525	4486	3496
	Theoretical delay bound	13000	11700	11700	10400	10400	7800	7800
Random	Maximum hop count	10	9	9	8	7	7	6
	Maximum delay	6659	6683	6646	5695	4595	4481	4489
	Theoretical delay bound	13000	11700	11700	10400	9100	9100	7800

The maximum hop count and delay of both the two algorithms are listed in Table 4.3. We can see that the maximum delay is bounded by Equation (4.15). The simulation results are in line with Property 3. For example, when the sink's position is (15, 15), the simulated maximum delay of *WSEDR* and *Random* is 5720 and 6659 slots, respectively. The delay of both the two schemes is shorter than the theoretical delay.

4.2.6 Discussion about the two experiments

In the two experiments, different schemes are adopted. In the second scheme, we use both the nodes with the same hop count and the nodes with the lower hop count as the candidate nodes. Because the second scheme has much more candidate nodes than the first scheme, the delivery ratio of the second scheme is much higher than the of the first scheme. And because of the low duty-cycle, the second scheme also introduces higher end-to-end delay.

If we allow the first scheme to retransmit the packets, the end-to-end delay will be similar to that of the second scheme. On the other hand, because the second scheme still has more candidate nodes, the delivery ratio will be higher too. As a result, we did not simulate the case where we allow the node to retransmit the packets in the first scheme.

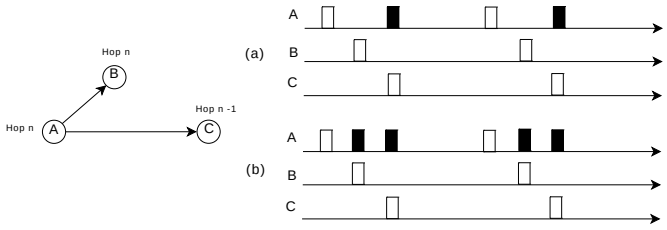


Figure 4.19: Discussion about the two experiments

We want to explain this by a simple example shown in Figure 4.19. In this example, we have three nodes, A,B and C. The hop counts of node A and B are n. The hop count of node C

is n-1. The basic ideas of scheme 1 and scheme 2 are shown in Figure 4.19 (a) and Figure 4.19 (b) respectively. As we can see from the two figures, the maximum delays of the two schemes are the same in this example. And because scheme 2 has much more candidate nodes than scheme 1, it will have a higher delivery ratio.

4.2.7 Conclusion

In this section, we have presented the synchronous forwarding protocol for low duty-cycle WSNs which schedules the wakeup slot of each node according to their hop count and *EDR* to the sink. The proposed algorithm maximizes the delivery ratio under real-time constraints. Through the mathematical model we proposed, we proved that the proposed algorithm has the best performance in terms of average delivery ratio. Simulation results are closely in line with the mathematical model.

Although the proposed synchronous forwarding protocol has good performances, to synchronize the sensor nodes in large-scale WSNs is expensive. As a result, asynchronous forwarding protocol is more energy efficient and easier to implement. In the next section, we will present the proposed asynchronous forwarding protocol.

4.3 Asynchronous forwarding

In this section,the proposed asynchronous forwarding protocol which is called Real-Time Constrained Forwarding (RTCF) will be presented. It mainly includes the following three parts: (1) calculate the Expected Delivery Ratio (EDR) and hop count, (2) select potential candidates, (3) calculate the backoff sleep time. We will depict them in detail in the following subsections.

4.3.1 Assumptions

In this protocol, less assumptions are used compared with the assumptions used in the synchronous forwarding protocol (see Section 4.2.1). We assume that only one node sends the alarm when the event happens. This can be done by using one aggregation method [Liu et al., 2009; Ye et al., 2005]. It is also assumed that the *ACK* packet from the receiver can be received by the sender if the receiver receives the *DATA* packet successfully. This is reasonable because the size of the *DATA* packet is much greater than the size of the *ACK* packet and the wireless channel has correlation characteristic [Cerpa et al., 2005; Sang et al., 2007].

4.3.2 overview

RTCF includes two phases: initialization and run-time phases. During the initialization phase, each node measures the link qualities between themselves and their neighbors (in this protocol, we do not talk about how to achieve this. Further information about how to get the link quality can refer to [Woo et al., 2003; Zhang et al., 2009]). Each node also calculates the EDR and

hop count to the sink node during the initialization phase. We will introduce the details in the following subsection. Because the topology of the network may be changed after running for a long time, e.g. node dies or link qualities change, the network should be reinitialized in order to maintain the performance. This can be done with a very lower frequency.

During the run-time phase, every node will be in the sleep state most of the time to save energy and prolong the lifetime. Due to the fact that every node has a low duty-cycle, preamble sampling technique is adopted in our protocol. If there is no event, every node will duty-cycle repeatedly, waking up for a short time to sense the channel and sleeping for a long time. The wakeup time of each node is chosen randomly within a cycle. And all the nodes have the same length of wakeup and sleep period.

When an event happens, a node is chosen to be the reporter and has a packet to send. It will transmit many short preambles separated by short listening periods. All the short preambles and listening periods constitute the preamble period (Figure 4.20). A node that does not receive any preamble packet when it wakes up will go to sleep and wake up later after the predefined sleep period. The candidate node that receives one of the short preamble packets will immediately reply with an ACK packet and wake up again to receive the DATA packet after a calculated backoff sleep period which is chosen according to the remaining number of preambles to be sent by the sender, the hop count and the EDR of itself. For the nodes with the same hop count, the greater the EDR value, the earlier wakeup time during the DATA period. At the end of the preamble period, the sender may have received some replies and starts the DATA period. During the DATA period, the sender will try to send the DATA packet at all the calculated wakeup times until receiving one ACK packet. Once the sender receives an ACK packet, it will immediately go to sleep. If the sender does not receive any ACK packet in the DATA period, it will decide whether to resend the preamble packets or to drop the packet by considering the remaining time before the deadline.

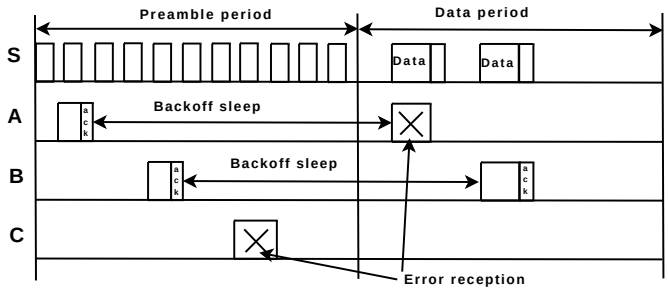


Figure 4.20: Real-time constrained forwarding

We want to explain RTCF by a simple example shown in Figure 4.20. We have four nodes in this example, namely S, A, B and C. Node S has a packet to send. It first sends a long preamble which is composed of some short preambles. After sending one short preamble, node S listens to the channel for a short time. Due to unreliable links, node C does not receive

the short preamble. It goes to sleep and will wake up until the next duty-cycle wakeup period. Node A and B receive the preamble packet. They reply with ACK packets immediately after receiving the short preamble packets and go to sleep. Then, they will wake up until the backoff sleep timer expires. Node A wakes up earlier than node B during the DATA period due to the lower hop count and greater EDR. Node S first tries to send the DATA packet to node A. Node A does not receive the DATA packet. Then, node S tries to send the DATA packet to node B. Node B receives the DATA packet and replies with an ACK. Node S receives the ACK packet. It will go to sleep and wake up until next duty-cycle wakeup period. This process will continue until the DATA packet reaches the destination.

4.3.3 Calculate the EDR and hop count

In WSEDR, the *EDR* and the hop count are used for forwarding. WSEDR has an initialization phase to calculate these two parameters. As RTCF is an asynchronous version of WSEDR, the *EDR* and the hop count in RTCF are calculated in the same way. Therefore, we can use the same algorithm proposed in WSEDR to initialize the network and calculate the two parameters. The details of the algorithm can be found in Section 4.2.4.

4.3.4 Select potential candidates

In WSN applications, each node may have a few neighbors, but it is not necessary to let all the neighboring nodes be the potential candidates. It is obvious that the delivery ratio will be increased if more nodes are chosen to be the next hop candidates, but this also increases the energy consumption as well as end-to-end delay because the packet may experience more hops (packet may be sent backward). How to select the potential candidates according to the real-time requirements?

In RTCF, the sender uses the information embedded in the preamble packets to tell the neighboring nodes which ones can be the next hop candidates. When the sending node finds the remaining time before the deadline is not enough to experience h hops (h is the hop count of the current sending node), it will drop the packet. And if it finds the remaining time before the deadline is longer enough to experience h hops and not enough to transmit h + 1 hops, it will add the neighboring nodes with hop count h - 1 into the candidate node set. If the sending node finds the remaining time is longer enough to transmit h + 1 hops and not enough to transmit h + 2 hops, it will add the neighboring nodes with hop count h and h - 1 into the candidate node set. Otherwise, if the remaining time is longer enough to transmit h + 2 hops, it will add the neighboring nodes with hop count h - 1, h and h + 1 into the candidate node set.

In order to realize the aforementioned scheme, we only need to add a flag in the preamble packet. In RTCF, the preamble packet includes the following information: (1) node id of the sender, (2) preamble's sequence number, (3) hop count of the sender, (4) flag. Preamble's sequence number is used to calculate the backoff sleep time which will be described in the

following subsection. When the flag embedded in the preamble packet is set to 1, the candidates which are h -1 hop will reply. If the flag is set to 2, the candidates which are h or h - 1 hop will take part in the forwarding process. If the flag is set to 3, the neighboring nodes with hop h - 1, h and h + 1 will reply.

In Equation (4.19), CS denotes the **C**andidate **S**et. NS^{-} represents the **N**eighbor **S**et where all the neighbors are $h-1$ hop away from the sink. $NS^{=}$ is defined as the **N**eighbor **S**et where all the nodes are h hop away from the sink. NS^{+} represents the **n**eighbor **s**et where all the nodes have hop count h + 1. $T_{remaining}$ represents the remaining time to reach the deadline.

$$CS = \begin{cases} Drop\ \ the\ \ packet & T_{remaining} < T_h \\ NS^{-} & T_h \leq T_{remaining} < T_{h+1} \\ NS^{-} \cup NS^{=} & T_{h+1} \leq T_{remaining} < T_{h+2} \\ NS^{-} \cup NS^{=} \cup NS^{+} & T_{remaining} \geq T_{h+2} \end{cases} \tag{4.19}$$

In our protocol, the remaining time is calculated by the elapsed time information embedded in the DATA packet. After a node receives the DATA packet, it will calculate the remaining time and set the flag before sending the preamble packet. T_{h+1} denotes the estimated time for the packet to be transmitted from h+1 hop to the sink. T_h can be calculated by Equation (4.20). T_{1hop}^{max}, which is the sum of the length of the preamble period and DATA period (see Figure 4.20), denotes the maximum delay that the packet experiences 1 hop.

$$T_h = h \times T_{1hop}^{max} \tag{4.20}$$

When the remaining time before the deadline is not enough to experience the rest of the hops, our proposed protocol drops the packets. This is the reason why RTCF has the miss ratio 0% (if a node finds that a packet can not reach the sink before the deadline, the packet will be dropped). We can see from Section 4.3.6 that the delivery ratio of RTCF is still higher than another one although RTCF drops the packets which may miss the deadline.

4.3.5 Calculate the backoff sleep time

As shown in Figure 4.20, a node will go to backoff sleep after it receives one of the short preambles successfully. In this subsection, we will introduce how we calculate the backoff sleep time. It is the summation of the following two sections: (1) the remaining time for the rest of the short preambles, (2) the time period that is related to the EDR and hop count. We calculate the backoff sleep time (T_{bs}) by Equation (4.21) and (4.22).

$$T_{bs} = N_{remaining} \times T_{pl} + \beta \times T_{dp} \tag{4.21}$$

$$\beta = 1 - ((h_s - h_r) + 2) \times \frac{1}{3} + (1 - EDR) \times \frac{1}{3} \tag{4.22}$$

$N_{remaining}$ denotes the remaining number of preamble packets that the sending node will

send. It can be calculated according to the sequence number which is embedded in the received preamble packet. After receiving the preamble packet, a candidate node can calculate the backoff sleep time using Equation (4.21) and (4.22). The backoff parameter, β, will be sent to the sender by embedding in the preamble ACK packet. After the sender receives the ACK packet, it can know the wakeup time of this receiver. The detailed definitions of the symbols in Equation (4.21) and (4.22) are listed in Table 4.4.

Table 4.4: Notations and descriptions in RTCF

Notation	Description
T_{dp}	*the time length of the DATA period*
T_{pp}	*the time length of the preamble period*
T_{bs}	*the time length of backoff sleep*
h_s	*the hop count of the sender*
T_{pl}	*the time length of one preamble and listening period*
$T_{DATA+ACK}$	*the time length of one DATA and one ACK packet*
h_r	*the hop count of the receiver*
EDR	*the expected delivery ratio of the receiver*

We want to explain why we use the Equation (4.22) to calculate the backoff sleep time. As described before, when a node is sending the preamble packets, the neighboring nodes that have hop count h - 1, h, and h + 1 (h is the hop count of the sender) may be chosen as the candidate nodes depending on the remaining time before the deadline. Among those nodes, the nodes with hop count h - 1 will be given the highest priority. And the nodes with hop count h + 1 will be given the lowest priority. So, we divide the data period (T_{dp}) into three parts. The first part is for the nodes with hop count h - 1, the last part for the nodes with hop count h + 1.

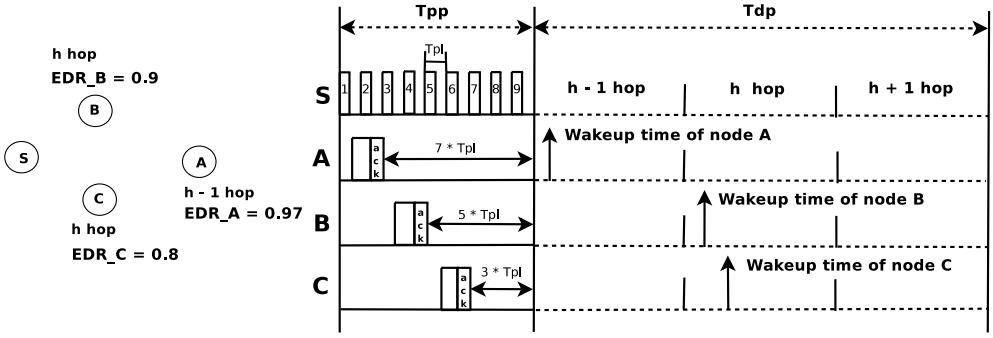


Figure 4.21: Example of calculating the backoff time

We explain the method of calculating the wakeup time by a simple example shown in Figure 4.21. Four nodes, S, A, B and C, are used in this example. The EDRs of node A, B and C are 0.97, 0.9 and 0.8 respectively. And the hop count of node A is h - 1 (h is the hop count of the sender). The hop counts of node B and C are h. Every node will calculate the wakeup time after they successfully receive one of the short preamble packets.

We can see from the Figure 4.21 that the backoff sleep time of each node consists of two sections. The first section is the duration that every node waits for the finish of the preamble period. The total number of preamble packets used in this example is 9. Because node A receives the second preamble packet, the first section of the backoff sleep time of node A is $7 \times T_{pl}$. Similarly, the first sections of the backoff sleep times of node B and C are $5 \times T_{pl}$ and $3 \times T_{pl}$ respectively. The second section of the backoff sleep time of each node is related to the EDR and hop count of themselves. We assume T_{dp} is the period for transmitting DATA packets. The whole DATA period is divided into three parts. The first part is reserved for the nodes with hop h - 1. And the last part is for the nodes with hop h + 1. Because node A has hop count h - 1, it wakes up during the first part. Both node A and B have hop count h, so they wake up in the second part. And because node B has a greater EDR than node C, node B wakes up before node C.

4.3.6 Performance evaluation

In this section, we will show the performances of RTCF by simulations. The basic simulation parameters are the same as those listed in Table 3.2 except the data rate here is set to 38.4kbps. Each point in the figures is the average result of 100 times experiments with different topologies. In each experiment with the same topology, the source node sends 50 packets to the sink. The 95% confidence intervals are within $1 \sim 10\%$ of the means.

Compared protocol

In this subsection, we want to show the protocol to which RTCF is compared. The existing real-time MAC protocols are either synchronization-based or unit disk graph model-based. For example, I-EDF [Caccamo et al., 2002] is a hard real-time protocol based on synchronization and unit disk graph model. Some real-time routing protocols, for example, RPAR [Chipara et al., 2006] and SPEED [He et al., 2003], can not be used directly in low duty-cycle WSNs because they do not consider the duty-cycle problem. When the sensor nodes are not synchronized to their neighboring nodes, it is very difficult for RPAR and SPEED to precisely predict the delay between a node and its neighboring nodes. As a result, in order to show the performances of our proposed protocol, we choose to compare it with a combined protocol which we call OP + XMAC, in which OP means the Optimized Path routing protocol. In the routing layer, each node chooses the neighboring node which has the biggest $EDR \times link\ quality$ metric to be the next hop candidate node (this method is chosen because of the same idea proposed in [Seada et al., 2004b]). XMAC [Buettner et al., 2006] is chosen as the MAC layer protocol. For OP + XMAC, the larger the retransmission times are, the higher the delivery ratio will be. In the following subsections, we will show the simulation results in which the maximum retransmission time is set to 5, 7 and 9 times.

Compared metrics

The performances of RTCF are compared with the performances of OP + XMAC in terms of delivery ratio, end-to-end delay and miss ratio.

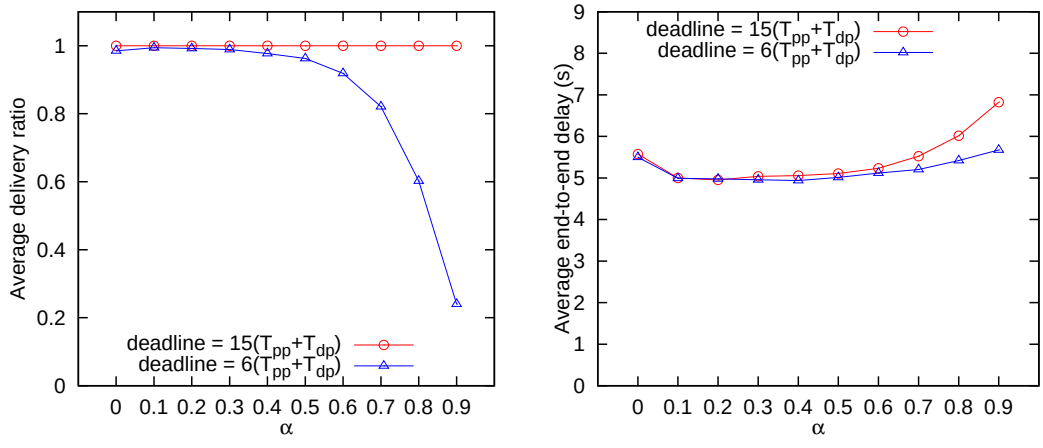
(1) Delivery ratio. The ratio of the number of packets received by the destination node to the number of the packets sent by the source node.

(2) End-to-end delay. The time experienced by the packet from the original sender to the sink node.

(3) Miss ratio. The ratio of the number of packets that exceed the deadline to the number of packets received by the sink node.

Impact of the value of α

We can see from previous sections that the value of α has impacts on the hop count of every node. The greater the value of α is, the greater the maximum hop count of the network will be. However, the hop count has significant impacts on the delivery ratio and the end-to-end delay. In this section, we will show the impacts of the value of α. In this simulation, 150 nodes are deployed in a $100m \times 100m$ area (the average number of neighbors is 35). The coordinates of the sink node and source node are (15, 15) and (85, 85) respectively. The rest of the nodes are randomly deployed. The duty-cycle is set to 1/175. The value of α is changed from 0 to 0.9. The length of the DATA period is set to $250 \times T_{DATA+ACK}$. The deadline in this simulation is set to two different cases: $15 \times (T_{pp} + T_{dp})$ and $6 \times (T_{pp} + T_{dp})$.



(a) Average delivery ratio (b) Average end-to-end delay

Figure 4.22: Impacts of α on the performances of RTCF

In Figure 4.22(a), it is shown that the average delivery ratio is almost the same no matter what the value of α is when the deadline is $15 \times (T_{pp} + T_{dp})$. And the average delivery ratio is decreasing as the increase of the value of α when the deadline is set to $6 \times (T_{pp} + T_{dp})$. This

is because the increase of the value of α will result in the increase of the number of hops from the source node to the sink node. When the deadline is set to $15 \times (T_{pp} + T_{dp})$ that is long enough, although the number of hops is increased, the source node still has more candidates to forward the packet, which results in the fact that the average delivery ratio is not decreased. But when the deadline is set to $6 \times (T_{pp} + T_{dp})$, the number of dropped packets is increased as the number of hops increases. So the average delivery ratio is decreased.

And we can see from Figure 4.22(b) that the average end-to-end delays of both two cases are changed a lot as we change the value of α. The average end-to-end delays are decreasing at the beginning, then increasing as the value of α increases. When the value of α is set to 0, although the maximum hop count of the network is lower, the link qualities between the neighboring hops are also lower. This results in the greater end-to-end delay at the beginning. As the value of α increases, although the maximum hop count of the network is increasing, the link qualities are also increased. This is the reason why the average end-to-end delays decrease and then increase. And the average end-to-end delay of the case where the deadline is set to $15 \times (T_{pp} + T_{dp})$ is always higher than that of the case where the deadline is $6 \times (T_{pp} + T_{dp})$.

The exact value of α could be set according to the applications' deadline and reliability requirements. In our following simulations, α is set to 0.3.

Impact of the duty-cycle

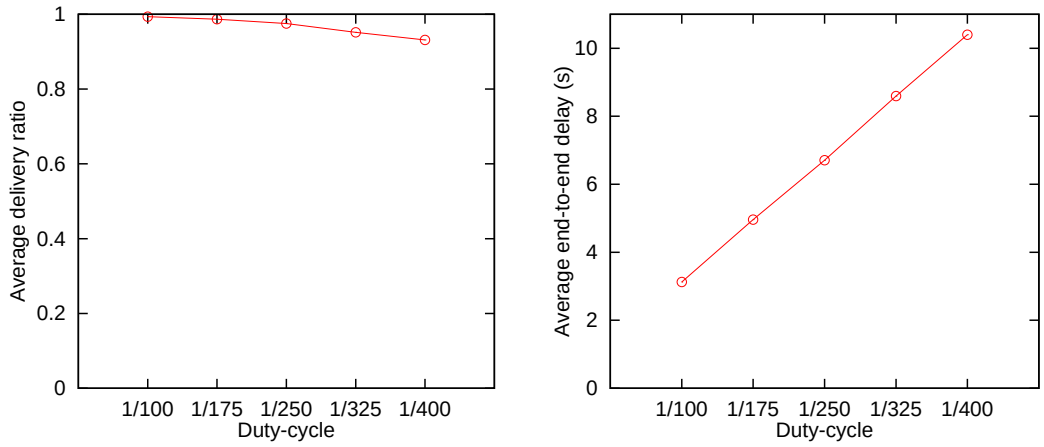
Given a constant wakeup period, if the duty-cycle is increased, the length of whole cycle will be decreased. As a result, the probability that some nodes have collisions during the preamble period will be increased. This will impact the delivery ratio because less nodes are woke up. In this simulation, we keep the number of nodes at 150. The positions of the source and the sink are (85, 85) and (15, 15) respectively. The duty-cycle is chosen from the following set: $\{$1/100, 1/175, 1/250, 1/325 and 1/400$\}$. We do not change the length of wakeup period and change the duty-cycle by adjusting the length of sleeping period in a cycle. The deadline is set to $6 \times (T_{pp} + T_{dp})$. And the length of T_{dp} is set to $250 \times T_{DATA+ACK}$.

From Figure 4.23(a) we can see that the average delivery ratio is decreasing as the duty-cycle decreases. But the differences are very small. The reason why the average delivery ratio is decreased is because the time saving at the previous hops is not long enough to experience the remaining hops as the length of the preamble period increases, and those packets are dropped.

In Figure 4.23(b), it is shown that the average end-to-end delay is increasing when we decrease the duty-cycle. This is because the length of the preamble period is increased when we decrease the duty-cycle.

Impact of the length of the DATA period (T_{dp})

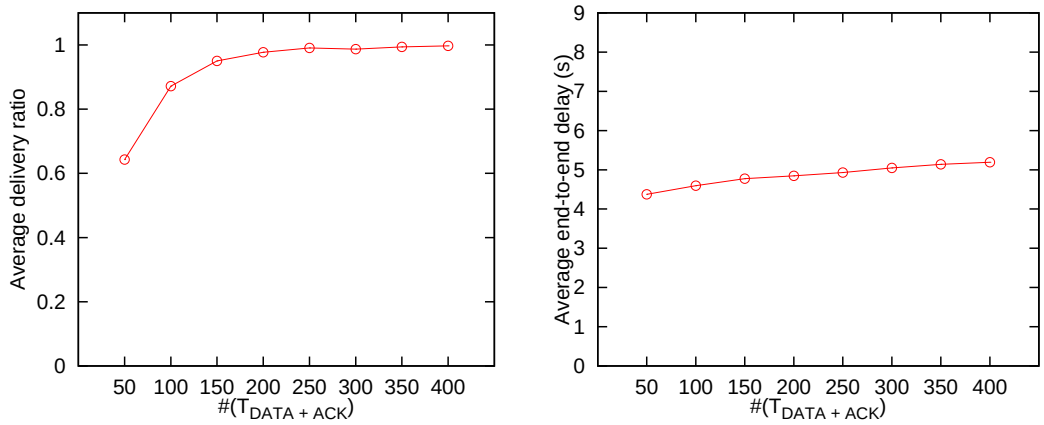
Another parameter, the length of the DATA period (T_{dp}), also has impacts on the delivery ratio. If we set a longer DATA period, the collision probability of the DATA packets will be



(a) Average delivery ratio (b) Average end-to-end delay

Figure 4.23: Impacts of duty-cycle on the performances of RTCF

lowered. In this subsection, we want to show the impacts of the length of the DATA period. In the simulations, the sink and the source nodes' positions are the same as that in previous simulations. Other nodes are randomly deployed. The duty-cycle in this simulation is set to 1/175. And we set the deadline to $6 \times (T_{pp} + T_{dp})$. The length of the DATA period is changed from $50 \times T_{DATA+ACK}$ to $400 \times T_{DATA+ACK}$.



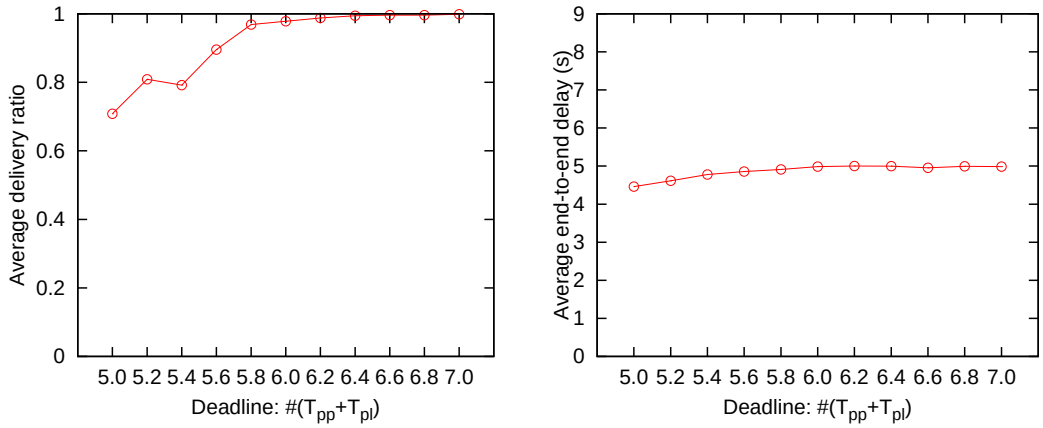
(a) Average delivery ratio (b) Average end-to-end delay

Figure 4.24: Impacts of T_{dp} on the performances of RTCF

The impact of the length of the DATA period on the average delivery ratio is shown in Figure 4.24(a). We can see that the average delivery ratio is increasing as T_{dp} increases. This is because the collision probability of the DATA packets is decreased when we increase the length of the DATA period. On the other hand, the average end-to-end delay is increased a little as the length of the DATA period increases, which is shown in Figure 4.24(b).

Impact of the deadline

Because RTCF can dynamically adjust the number of candidate nodes according to the remaining time before the deadline, the performances of RTCF will be changed when we change the deadline. In this subsection, we want to show the impact of the deadline. In the simulation, 150 nodes are deployed in the $100m \times 100m$ area. The source node and sink node are at (85, 85) and (15, 15) respectively. Other nodes are randomly deployed. The duty-cycle of every node is 1/175. The DATA period (T_{dp}) is set to $250 \times T_{DATA+ACK}$. We change the deadline from $5 \times (T_{pp} + T_{pl})$ to $7 \times (T_{pp} + T_{pl})$.



(a) Average delivery ratio (b) Average end-to-end delay

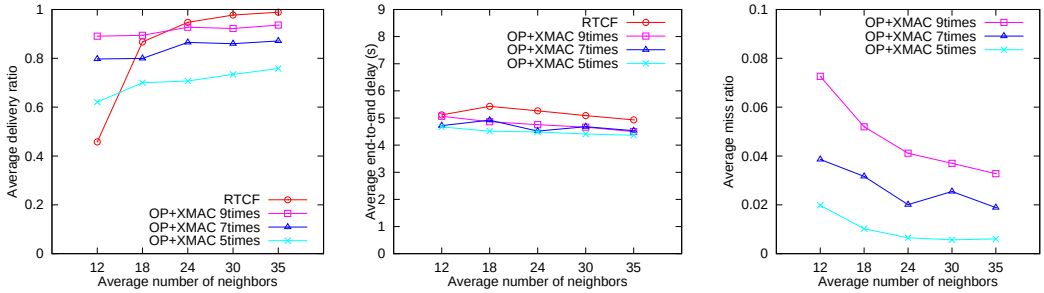
Figure 4.25: Impacts of deadline on the performances of RTCF

We can see from Figure 4.25(a) that the average delivery ratio of RTCF is increasing with the increase of the deadline. This is because more potential candidates are chosen as the deadline is increasing. When the deadline is greater than $6 \times (T_{pp} + T_{pl})$, the variances are very small. It is shown in Figure 4.25(b) that the average end-to-end delay increases a little when the deadline is increased.

Impact of the density

For RTCF, the greater the density is, the greater the size of potential candidate node set may be. This may increase the delivery ratio of RTCF. In this subsection, we want to show the impacts of the density. We keep the area size constant and change the number of nodes in this area from 50 to 150 (because of the unreliable links, one node sometimes can receive packets from some neighbors that are far away. In this simulation, we only count the neighbors that have the link quality which is higher than 0.2. As a result, the average number of neighbors with the aforementioned number of nodes is from 12 to 35). The positions of the source and the sink are the same as described in previous sections. The duty-cycle in this simulation is set to 1/175. And T_{dp} is set to $250 \times T_{DATA+ACK}$. The deadline in this simulation is $6 \times (T_{pp} + T_{pl})$. For

OP + XMAC, the source and the sink nodes' positions are the same as that in RTCF. And the duty-cycle is also the same. The retransmission times of OP + XMAC is changing from 5 to 9.



(a) Average delivery ratio (b) Average end-to-end delay (c) Average miss ratio

Figure 4.26: Impacts of density on the performances of RTCF

We can see from Figure 4.26(a) that the average delivery ratios of both the two protocols are increasing with the increase of the average number of neighbors. The average delivery ratio of RTCF is lower than the average delivery ratio of OP + XMAC when the density of the network is very low (e.g. the average number of neighbors is 12). This is because RTCF has less potential candidates when the density is low while one candidate node is enough for OP + XMAC because it can retransmit to this candidate node for many times. And we can see that the average delivery ratio of OP + XMAC is increasing when we increase the number of retransmission times. For example, the average delivery ratio of OP + XMAC with 7 retransmission times is greater than that of OP + XMAC with 5 retransmission times. When the average number of neighbors changes from 18 to 35, the average delivery ratio of RTCF is always higher than the delivery ratio of OP + XMAC with 5 and 7 retransmission times. The reason why RTCF can have a higher average delivery ratio when the number of nodes is greater is because RTCF has more candidate nodes.

In Figure 4.26(b), it is shown that the average end-to-end delay of RTCF is a little higher than the average end-to-end delay of OP + XMAC whatever the retransmission times is. The average end-to-end delay of OP + XMAC with 9 retransmission times is greater than that with 7 retransmission times. Figure 4.26(c) shows that the average miss ratio of OP + XMAC is decreasing when we increase the number of nodes. And the average miss ratio of OP + XMAC is increasing as the number of retransmission times increases. However, the miss ratio of RTCF is zero which is not shown in this figure.

4.3.7 Conclusion

In this subsection, we have proposed a Real-Time Constrained Forwarding (RTCF) protocol for low duty-cycle event-driven WSNs. RTCF is an asynchronous forwarding protocol that adopts preamble sampling technique. RTCF dynamically adjusts the number of potential relay

nodes by considering the remaining time before the required deadline. After comparing RTCF with OP + XMAC, we showed that RTCF has better performances than OP + XMAC in terms of average delivery ratio and average miss ratio.

4.4 Summary

In this chapter, two cross-layer forwarding protocols (i.e. WSEDR and RTCF) have been proposed for ultra-low duty-cycle, event-driven WSNs with real-time constraints. Both the two protocols use the EDR and hop count information to forward the packets. WSEDR is a synchronous forwarding protocol which assumes all the sensor nodes are locally synchronized. It schedules the wakeup slots of every node according to their own hop count and EDR locally. This protocol can have a bounded delay on the packets that are received by the sink. The performances of WSEDR were shown by mathematical model and simulation. The disadvantage of this protocol is that it is based on synchronization which is difficult or expensive to implement in large-scale WSNs.

RTCF is an asynchronous forwarding protocol which is much easier or cheaper to implement compared to WSEDR. Preamble sampling technique is adopted by RTCF because a sender does not know the exact wakeup time of its neighbors. RTCF is an asynchronous version of WSEDR, so it also uses the hop count and EDR to forward the packets. RTCF can dynamically adjust the number of potential relay nodes by considering the remaining time before the deadline. The performances of RTCF were shown by simulation results.

Chapter 5

Robust against topology changes

5.1 Introduction

As we have described in Chapter 1.1.3, the topology of WSNs is sometimes dynamic, for example, due to unreliable links, dead nodes or new added nodes. Because of the dynamic topology, it is necessary to have robust protocols for WSNs. Generally speaking, the impacts of the topology change can be alleviated by periodically sending packets to reinitialize the network, but it will consume a large amount of energy.

In Chapter 4, we have proposed two forwarding protocols (i.e. WSEDR and RTCF) which aim to increase the reliability under unreliable links and real-time constraints. Both of them have good performances in terms of delivery ratio and energy consumption. However, less attention is paid to the topology change problem. Both the two forwarding protocols adopt EDR (Expected Delivery Ratio) and hop count as the main metrics for forwarding. EDR is related to more than one neighbors. As a result, it is difficult to update EDR by sending one reinitialization packet because every node needs to know the EDRs of some neighbors. Although we can send more than one reinitialization packets in order to get more precise updated EDR, they consume much more energy. So there exists a trade-off between the accuracy and the energy consumption. Moreover, periodically sending the reinitialization packets can not completely solve the problem in event-driven WSNs. For example, some nodes die between two reinitialization times, and an event happens just after the death of those nodes. In this case, the reinitialization packets can not help. Thus it is better to design a robust protocol which can dynamically update the network no matter when an event happens and when some nodes die.

In this chapter, we will present a new proposed Robust Forwarding (RF) protocol. Different from WSEDR and RTCF, RF uses RSSI/LQI as the metric for forwarding. By using this metric, it is easy for the protocol to update the network when the topology changes because this metric is only related to one neighbor. Compared with RTCF proposed in Chapter 4, RF can have a higher delivery ratio when the topology changes. The detailed simulation results will be shown at the end of this chapter.

5.2 Robust Forwarding

5.2.1 Overview

Similar to RTCF in Chapter 4, RF is an asynchronous forwarding protocol and includes two phases: initialization and run-time phases. During the initialization phase, each node gets the COST information. The details will be described in the following subsection.

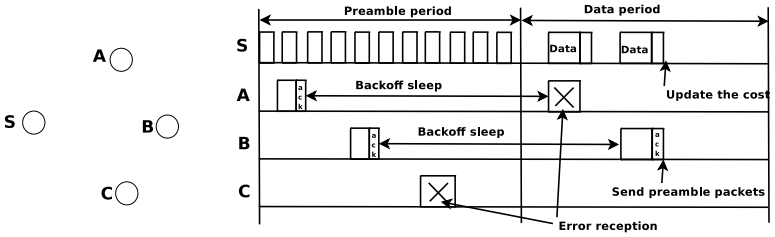


Figure 5.1: A simple example to show the basic idea of RF

During the run-time phase, every node will be in the sleep state most of the time to save energy and prolong the lifetime of WSNs. Due to the fact that every node has a low duty-cycle, preamble sampling technique is adopted in RF. A node that does not receive any preamble packet during the preamble period will go to sleep and wake up until the next wakeup time. When a node has a packet to send, it will send many short preambles separated by short listening periods. All the short preambles and listening periods constitute the preamble period (Figure 5.1). The basic forwarding process is similar to the process in RTCF presented in Chapter 4. The differences lie in the following points: (1) this protocol uses RSSI/LQI as the metric while the protocols in Chapter 4 adopt EDR, (2) the algorithms used to calculate the backoff sleep time are different and (3) this protocol has updating schemes. In the next sections, we will depict them one by one.

5.2.2 Get the COST information

In RTCF, the metrics used for forwarding are EDR and hop count. Because EDR is related to a few neighbors, it is not easy to update. One node needs to know the EDRs of all those neighbors. Although this can be done by sending more than one reinitialization packets, it consumes more energy. As a result, a more robust metric, RSSI (Received Signal Strength Indication)/LQI (Link Quality Indicator), is used in RF. RSSI and LQI are two metrics that can indicate the link quality. After receiving a packet, a node can know the two link quality values stored in their respective registers [CC2420, 2007]. Because RSSI and LQI can be got after a node receives a packet from another node, the protocol can easily update when the topology changes if it adopts those metrics. In a word, the key difference between EDR and LQI is that EDR is a statistical value while LQI is an instantaneous value ([Becker et al., 2009] presents some interesting experimentation results on the time-varying characteristics of LQI and shows

the relationships between LQI and transmission power level and distance).

In the proposed RF protocol, the COST value of every node indicates the minimized energy consumption from itself to the sink node. In order to get the COST information, every node just has to calculate the summation of the RSSI/LQI along the optimal path to the sink node. The details of this process will be described in the next subsection.

During the initialization phase, the sink node broadcasts an INIT packet which contains the COST and hop count information. The initialization phase is presented in Algorithm 1. The sink's COST and hop count are initialized to 0. The initialized COSTs and hop counts of other nodes are infinity. The node received an INIT packet will consider to update its information if the received signal strength is greater than a predefined threshold. If its old hop count is greater than the sum of the hop count of the sender from which the INIT packet is send and one, the node will update its hop count which is shown on line 4, 5 and 6 in Algorithm 1. Similarly, if the COST of the receiver is greater than the sum of the COST of the sender and the COST of the link between the sender and the receiver, the receiver will update its COST using the method shown on line 7, 8 and 9 in Algorithm 1. After updating, the node will rebroadcast an INIT packet that contains the new information after a backoff time. The algorithm here is similar to the one proposed in [Ye et al., 2001].

Algorithm 1: Initialize the network

1: Sink broadcasts the INIT packet that contains the *COST* and hop count information.
2: Every node except the sink runs the following algorithm.
3: **if** $RSSI_{received} > RSSI_{threshold}$
4: **if** $HC_{old} > HC_{sender} + 1$
5: $HC_{new} = HC_{sender} + 1$
6: **endif**
7: **if** $COST_{old} > COST_{sender} + COST_{link}$
8: $COST_{new} = COST_{sender} + COST_{link}$
9: **endif**
10: Broadcasts the INIT packet with new information after a backoff time
11: which is proportion to the link cost. Only broadcasts at the time when the
12: smallest backoff time expires if a node has more than one backoff times.
13: **endif**

In Algorithm 1, $RSSI_{received}$ and $RSSI_{threshold}$ are defined as the received signal strength and the signal strength threshold respectively. HC_{old} and HC_{new} indicate the hop count before updating and the hop count after updating respectively. HC_{sender} is the hop count of the sender which sends the INIT packet. $COST$ means the estimated energy consumption from a node to the destination. $COST_{old}$ and $COST_{new}$ represent the $COST$ before updating and the $COST$ after updating respectively. $COST_{sender}$ is defined as the $COST$ of the sender which sends the INIT packet. $COST_{link}$ is the $COST$ between the sender and the receiver.

5.2.3 Calculate the backoff sleep time

After the initialization phase, all the nodes get the COST information. Then, the network will enter into the run-time phase. As we described in Chapter 5.2.1, a node will send out preamble packets when an event is detected. There would be some neighboring nodes that receive one of the preamble packets. When a node receives a preamble packet, it might be selected as the next hop, so it has to wait for the decision that whether it is chosen as the next hop. In order to save energy, we let the node that receives the preamble packet sleep before waking up for receiving the DATA packet. Due to the fact that there are more than one candidates, there will have collisions if all those candidates wake up at the same time. Moreover, we want to choose the node with the least COST as the next hop. As a result, Algorithm 2 is proposed to calculate the backoff time. In RTCF, there is an algorithm to calculate the backoff sleep time. Because the metric used in RTCF is EDR which is a value between 0 and 1 and the COST in RF is an integer, the algorithm to calculate the backoff sleep time can not be directly used for RF. The key idea of Algorithm 2 is to let the node with a lower $COST$ wake up much earlier.

Algorithm 2: Calculate the backoff sleep time

1: **if** $COST == 0$ //Sink node
2: $T_{backoff} = 0$
3: **else**
4: $temp = \lfloor (log10(COST) \times 100) \rfloor \%100/100.0$
5: $temp_{pre} = \lfloor (log10(COST_{pre}) \times 100) \rfloor \%100/100.0$
6: **if** $COST_{pre} > COST$ and $temp_{pre} < temp$
7: $T_{backoff} = (1 - temp) \times CWF \times T_{DATA+ACK}$
8: **else if** $COST_{pre} < COST$ and $temp_{pre} > temp$
9: $T_{backoff} = (1 + temp) \times CWF \times T_{DATA+ACK}$
10: **else**
11: $T_{backoff} = temp \times CWF \times T_{DATA+ACK}$
12: **endif**
13: **endif**

By using Algorithms 2, every node can calculate the backoff sleep time after receiving one of the preamble packets. $COST$ indicates the estimated energy consumption of a node to the sink. $COST_{pre}$ represents the $COST$ of the node which has sent the preamble packets. $COST_{pre}$ is embedded in the short preamble packets, so every node can calculate the backoff sleep time after they receive one of the preamble packets. CWF is defined as a contention window factor. $T_{DATA+ACK}$ indicates the time for transmitting a DATA and an ACK packet.

The $COST$ of a node may be very great which depends on the number of hops, for example, the COST may be around 1000 if the hop count is 10. Because we want to only use COST to calculate the backoff sleep time, and the delay will be great if we directly use COST for the backoff sleep time, so logarithmic function is used to try to make every node have different wakeup time and lower sleep delay, which is shown in line 4 to line 12. We want to explain

the backoff sleep algorithm by the simple examples shown in Figure 5.2. Line 6 and 7 in Algorithm 2 are shown in the case 1 of Figure 5.2. The case 2 in Figure 5.2 shows the method in line 8 and 9. The case 3 describes the method in line 11.

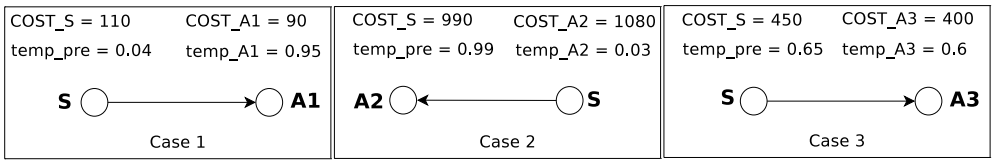


Figure 5.2: Simple examples to explain the calculation of the backoff sleep algorithm

A more detailed example is shown in Figure 5.3. In this example, node S is the sender and its COST is 990. There are other two nodes, A and B. Their COSTs are 1080 and 900 respectively. And we define T as the product of CWF and $T_{DATA+ACK}$. Because node B has a lower COST, it should have a lower backoff sleep time when forwarding. After using the proposed calculation algorithm, node B gets a shorter backoff sleep time than node A.

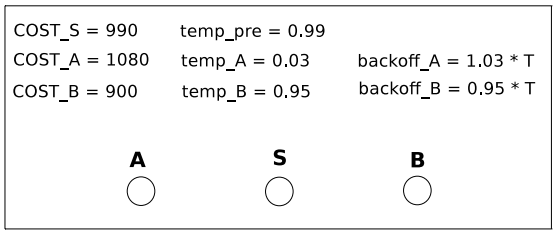


Figure 5.3: An example to show backoff sleep algorithm

5.2.4 Update the COST

As we discussed in the previous sections, WSNs sometime have dynamic topology. Some nodes may run out of energy or some new nodes may be added. So the protocol designed for WSNs should be robust to cater the dynamicity. In this section, we will show the update scheme of RF.

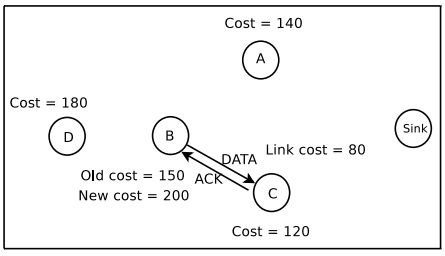


Figure 5.4: General update example of RF

In RF, after the sender transmits the DATA packet, it will update its cost if it successfully

receives the ACK packet from the receiver. Similar to the method used in the initialization phase, the sender updates its cost according to the Equation (5.1). The definitions of the parameters in Equation (5.1) are the same as those in Algorithm 1. The basic update scheme is shown in Figure 5.4. In this simple example, node B first sends the DATA packet to node C because it has the lowest $COST$ (120). After node C receives the DATA packet from node B, an ACK packet is sent. And node B receives the ACK packet from node C. Then, node B updates its COST to 200 because the COST of node C is 120 and the link cost between node B and node C is 80.

$$COST_{new} = COST_{sender} + COST_{link} \tag{5.1}$$

With the update scheme presented in Figure 5.4, RF can easily handle the dead node and new added node cases. In the following two subsections, we will introduce the details about how the update scheme performs when the network has dead nodes and new added nodes.

Deal with the dead node

We want to show how RF performs when the network has dead nodes by a simple example shown in Figure 5.5. In this example, 7 nodes are used. And the $COST$ of node A is 280 after the initialization phase (other nodes' COSTs are shown in Figure 5.5). After running for a certain time, node E and F die. When node A has a packet to send, it will send the preamble packets. And node B receives one preamble packet. Then, node A sends a DATA packet to node B. Node A will update its COST to 370 after receiving the ACK packet from node B.

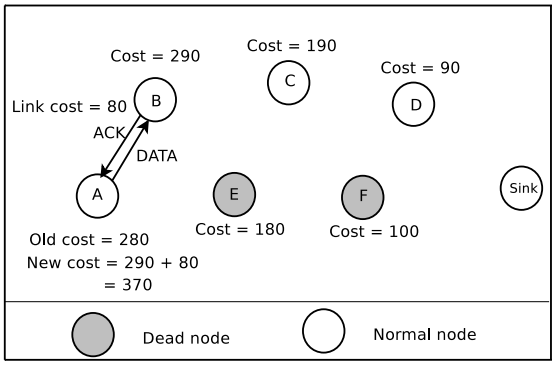


Figure 5.5: Update example when the network has dead nodes

Deal with the new added node

In order to deal with the new added nodes, we propose two schemes, i.e. proactive update and on-demand update. Proactive update means the new added node sends packets to get its COST after it is deployed. On-demand update indicates the new added node does not do anything until it receives preamble packets. In the following subsections, we will describe the aforementioned two schemes in detail.

——Proactive update

When a new node is added into the network, it will try to send preambles to wake up the neighboring nodes. After a node is woke up, it will send an ACK packet containing the $COST$ information of itself. The new node can update its $COST$ information after receiving some ACK packets from its neighboring nodes. This update scheme is shown in Figure 5.6. In this example, the COSTs of node B, C and D are 100, 190 and 90 respectively. Node A is a new node deployed in the network. After node A sends the preamble packets, node B, C and D all receive one of the preamble packets. The link costs between node A and node B, node C and node D are 90, 100 and 105. After node A receives those three ACKs, it has three choices to update the COST, i.e. 100 + 90 = 190, 190 + 100 = 290 or 90 + 105 = 195. Because the lowest value is 190, it is chosen as the COST of node A.

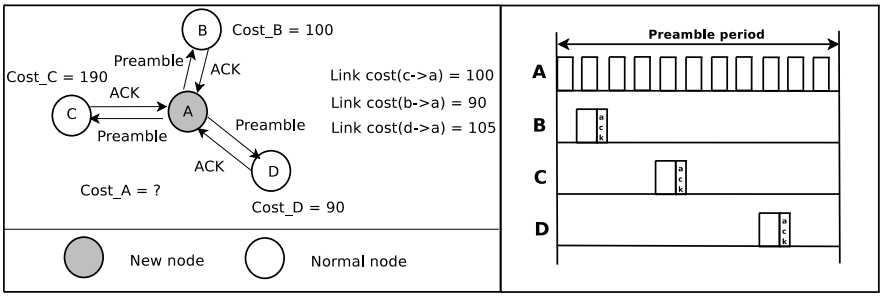


Figure 5.6: Proactive update example

——On-demand update

In on-demand update scheme, a new added node does not need to do anything until it receives a preamble packet. When the new node receives the preamble packet which includes the sender's COST information, it will run the on-demand update algorithm shown in Algorithm 3. $COST_{sender}$, $COST_{new-node}$ and $COST_{link}$ represent the COST of the sender, the new added node and the link between the sender and the new added node, respectively.

Algorithm 3: On-demand update algorithm

1: **if** $COST_{sender} - COST_{link} > 0$
2: $\quad COST_{new-node} = COST_{sender} - COST_{link}$
3: **else**
4: $\quad COST_{new-node} = COST_{sender} + COST_{link}$
5: **endif**

The update scheme is shown by a simple example in Figure 5.7. In this example, there are four nodes, A, B, C and D. Node A is deployed after the network runs for a long time. After node A is deployed, it does not need to do anything. When node C has a packet to send, it will send the preamble packets to wake up the potential candidates. Node A will reply with an

ACK packet after receiving one of the preamble packet. At the same time, node A updates its cost to 90 which is the difference between the cost of node C and the cost of the link between node C and node A.

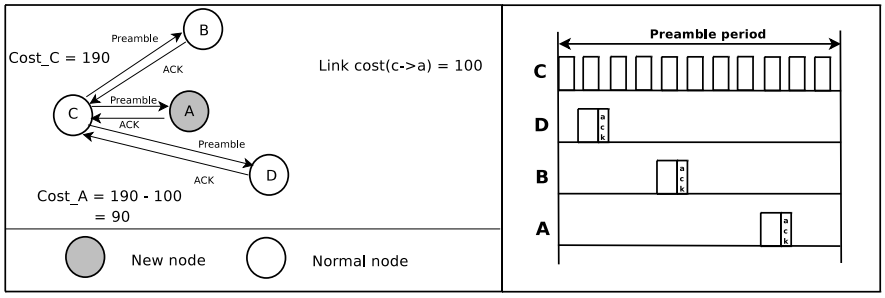


Figure 5.7: On-demand update example

5.2.5 Property of RF

For routing or forwarding protocols, it is important to know if the protocol is loop-free or not. In this section, we want to show the loop-free property of RF.

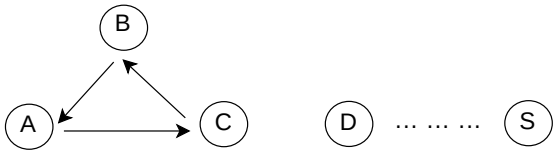


Figure 5.8: Simple example to prove loop-free

Property: *RF is loop-free if the network is connected.*

Proof. We want to prove the property by the simple example shown in Figure 5.8. In this example, five nodes, A, B, C, D and S are used. Node S is the destination node.

If the links are stable, the relationship of the COSTs of those nodes are: $COST_A > COST_B > COST_C > COST_D > COST_S$. We assume there is a loop among node A, B and C. Because RF chooses the node with a lower COST when forwarding, if there is a loop, this means the COST of node D is greater than the COST of node A, B and C. This is a contradiction. Therefore, RF is loop-free.

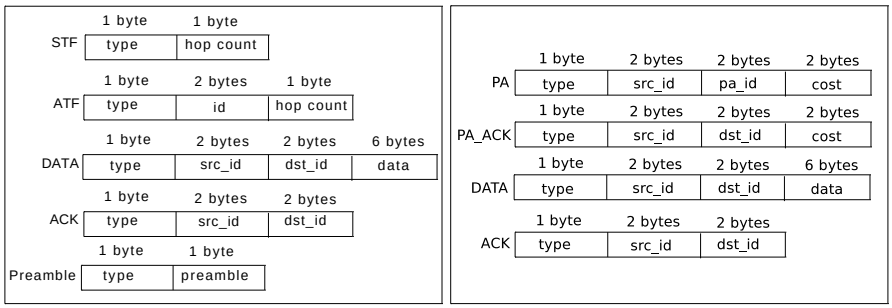
If the links are unstable, the relationship of the COSTs of those nodes are not known. We assume there is a loop among node A, B and C, which means $COST_D$ is greater than $COST_A$, $COST_B$ and $COST_C$. Because RF chooses the node with a lower COST and update the COST after successfully sending the DATA packet, $COST_A$, $COST_B$ or $COST_C$ will be greater than $COST_D$ after some times update. This is a contradiction. Therefore, RF is loop-free. □

5.3 Performance evaluation

In this section, we will show the simulation results of RF. The basic simulation parameters are the same as that used in RTCF (Chapter 4.3.6). Each point in the figures is the average result of 100 times experiments with different topologies. In each experiment with the same topology, the source node sends 50 packets to the sink. The 95% confidence intervals are within $1 \sim 10\%$ of the means. We set the simulation area to $100m \times 100m$. And the locations of the source and the sink nodes are (85, 85) and (15, 15) respectively.

5.3.1 Compared protocol

We want to compare RF to SOFA [Lee et al., 2006] because it is, to the best of our knowledge, the only existing asynchronous and robust forwarding protocol which considers ultra-low duty-cycle in WSNs. In Chapter 2.6, we have given the general idea of this protocol. In SOFA, the authors propose some schemes to update the hop count information when the network has new added nodes or disappeared nodes. This is the key idea of this protocol. Based on the proposed updating schemes, they adopt BMAC [Polastre et al., 2004] as the MAC layer protocol. But the disadvantage of SOFA is that they do not take the link quality into consideration when they forward the packet toward the sink node. In the following subsections, we will compare the performances of SOFA and RF.



(a) SOFA (b) RF

Figure 5.9: The messages used in SOFA and RF

Before showing the simulation results, we want to introduce the message types used in both protocols. The messages used in SOFA and RF are shown in Figure 5.9. Five message types are used in SOFA, i.e. Preamble, STF (Solicit To Forward), ATF (Accept To Forward), DATA and ACK. Four message types are adopted by RF, i.e. Preamble, ACK to preamble, DATA and ACK to DATA.

Because BMAC is used by SOFA, it can have a very low duty-cycle. This is because the receiver in BMAC will be ready to receive the DATA packet if it hears the transmission of other nodes. Compared with SOFA, RF has a greater duty-cycle because the receiver in RF needs to receive the preamble packet and get some information from the preamble packet. To be fair and

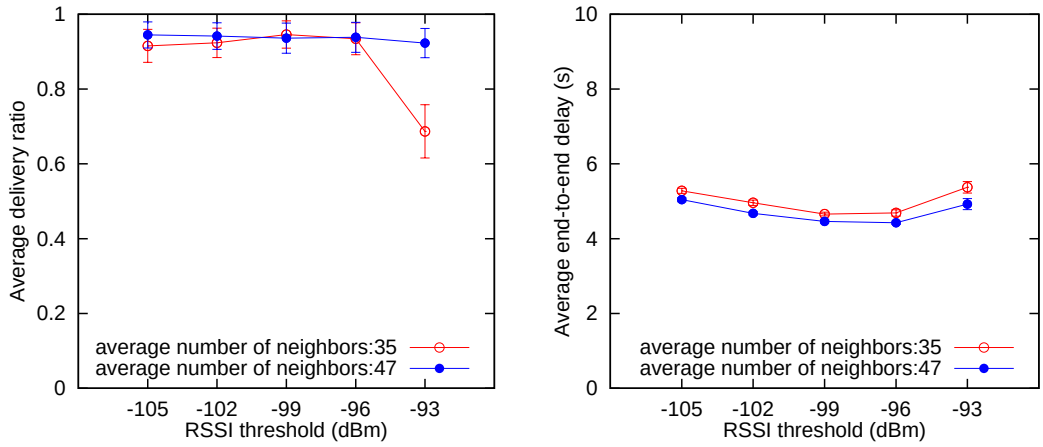
in order to make the two protocols have the same lifetime, the duty-cycles of the two protocols are set different. We want to explain this by a simple example. There are two nodes, A and B. A uses SOFA and B adopts RF. The total energy of each node is E. The duty-cycles of node A and B are D_A and D_B. And the lengths of wakeup period of node A and B are W_A and W_B. So the lifetimes of node A and B are $\frac{E}{W_A \times D_A}$ and $\frac{E}{W_B \times D_B}$. If we want the two nodes have the same lifetime, ie. $\frac{E}{W_A \times D_A} = \frac{E}{W_B \times D_B}$, the duty-cycles of the two nodes should be different, ie. $D_A \neq D_B$, because the nodes have different lengths of wakeup period ($W_A \neq W_B$).

In our simulations, the duty-cycles of sensor nodes in SOFA are set to 1/100 while RF sets the duty-cycles of all the nodes to 1/175. This is because the length of wakeup period of SOFA is set to 2.55ms (ie. $W_{SOFA} = 2.55ms$) according to [Polastre et al., 2004]. In RF, as we can see from Figure 5.9, the lengths of Preamble and ACK to preamble packets are both 7 bytes. Moreover, in order to make sure that a node could receive one of the preamble packets, the length of the wakeup period must be equal or greater than the sum of the lengths of Preamble packet and ACK to preamble packet. As a result, the length of wakeup period of RF is set to $2 \times T_{preamble} + T_{preamble-ack}$, where $T_{preamble}$ indicates the time used to send one preamble packet and $T_{preamble-ack}$ represents the time used to transmit one preamble ack packet. Because it costs 26042ns to send one bit in our simulation, the length of wakeup period of RF is $W_{RF} = (2 \times 7 + 7) \times 8 \times 26042 = 4.375ms$. So if we want the two protocol can result in the same lifetime and we set the duty-cycle of SOFA to 1/100, the duty-cycle of RF will be $D_{RF} = \frac{D_{SOFA} \times W_{SOFA}}{W_{RF}} \approx 1/175$.

5.3.2 Impacts of the RSSI threshold during the initialization phase

As we described in previous sections, each node gets its cost information during the initialization phase. If we set a RSSI threshold in this phase, what will be the performance of RF?. In this section, we will show the impacts of the RSSI threshold which is changed from -105dBm to -93dBm in the simulations. We have run two cases with different average number of neighbors, i.e. 35 and 47 (the number of nodes is 150 and 200 respectively). And the sink node only sends one INIT packet during the initialization phase. The contention window factor is set to 70 (The reason why we choose this parameter will be shown in Section 5.3.4).

In Figure 5.10(a), we can see that the average delivery ratio of the case with 47 neighbors is almost the same no matter what we change the RSSI threshold. And the average delivery ratio of the case with 35 neighbors is very close to the average delivery ratio of the case with 47 neighbors when the RSSI threshold is lower than -96. The average delivery ratio of the case with 35 neighbors decreases a lot when the RSSI threshold is set to -93. This shows that the RSSI threshold has negligible impacts on the average delivery ratio when the density is higher. This is because the network can be well initialized when the network density is higher. In Section 5.3.5, we will show the impacts of the density and it will be shown in that section that the performances are very close when the density is higher.



(a) Average delivery ratio (b) Average end-to-end delay

Figure 5.10: Impacts of the RSSI threshold on the performances of RF

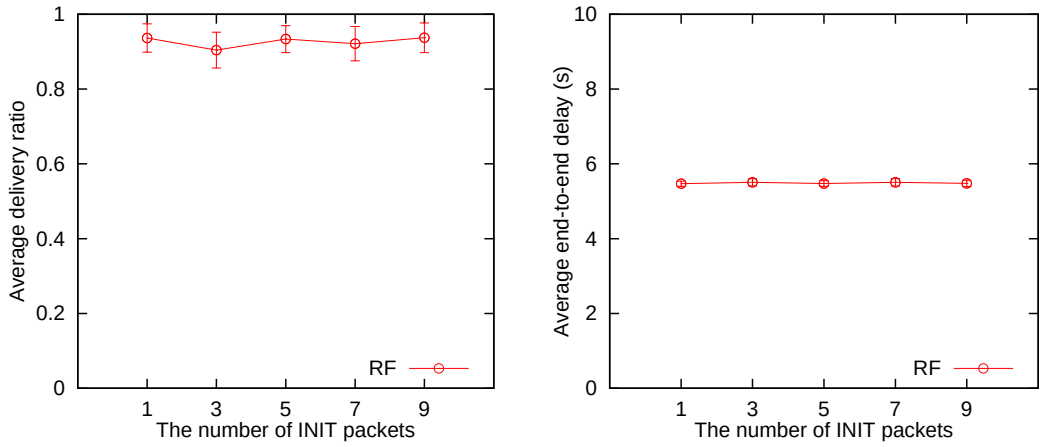
It is shown in Figure 5.10(b) that the trends of the two cases that have 35 and 47 neighbors are the same, decreasing a little then increasing when we increase the RSSI threshold. The average end-to-end delay of the case with 35 neighbors is higher than that of the case with 47 neighbors. This is because the network can not be well initialized when the network density is lower.

From those simulation results, we can see that RF can have a better performance if the network is well initialized. In this subsection, we let the sink node send only one INIT packet and increase the network density in order to well initialize the network. It appears that the network will have better performances if the density is increased. In fact, we can let the sink node send more than one INIT packet, which could also well initialize the network. In the following subsection, we will show the impacts of the number of INTI packets sending by the sink node.

5.3.3 Impacts of the number of INIT packets

During the initialization phase, the sink node sends the INIT packet to initialize the COST information of the network. Due to unreliable links, one INIT packet may be not enough to initialize all the nodes if the RSSI threshold is high, which is shown in the previous subsection. And the average delivery ratio is higher in the case where the RSSI threshold is lower, which is shown in Figure 5.10(a). In this section, we want to show the impacts of the number of the INIT packets on the performances of RF. We set the RSSI threshold to -105dBm because RF has better performances in this case. In the simulations, 150 nodes are randomly deployed, which results in the fact that the average number of neighbors is 35. The contention window factor is set to 70.

We can see from Figure 5.11(a) that the average delivery ratio varies a little when we change the number of INIT packets. And it is shown in Figure 5.11(b) that the average end-to-end



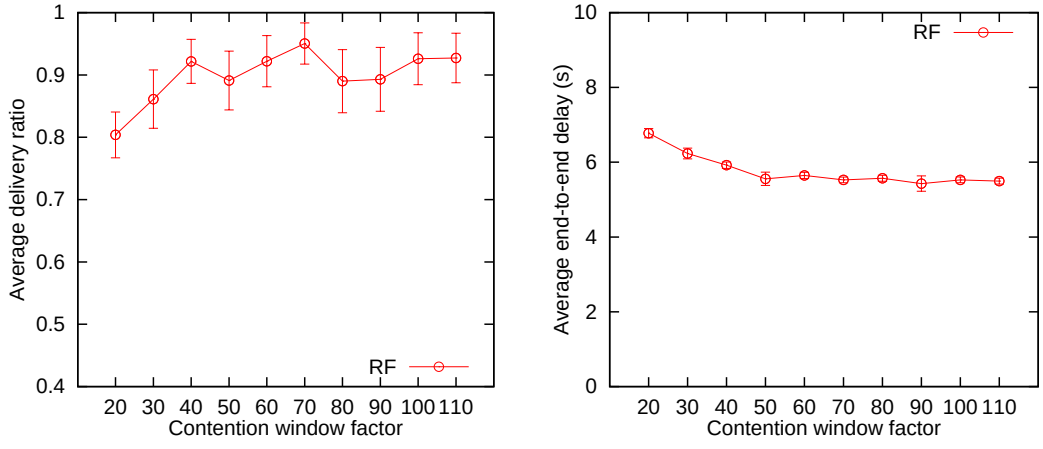
(a) Average delivery ratio (b) Average end-to-end delay

Figure 5.11: Impacts of the number of INIT packets

delay is almost constant as the number of INIT packets increases. Those results indicate that the number of INIT packet has negligible impacts on the performances of RF. In the following simulations, we do not set RSSI threshold and let the sink node send 3 INIT packets during the initialization phase.

5.3.4 Impacts of contention window factor

During the contention phase, every node will calculate the backoff sleep time according to Algorithm 2. In order to reduce the collision probability, a contention window factor is used in Algorithm 2. If we change the value of CWF, the collision probability will be changed too. Then, the performances (e.g. delivery ratio and end-to-end delay) of RF will be affected. In this section, the impacts of the contention window factor will be shown. In the simulations, 150 nodes are randomly deployed. The contention window factor is changed from 20 to 110.



(a) Average delivery ratio (b) Average end-to-end delay

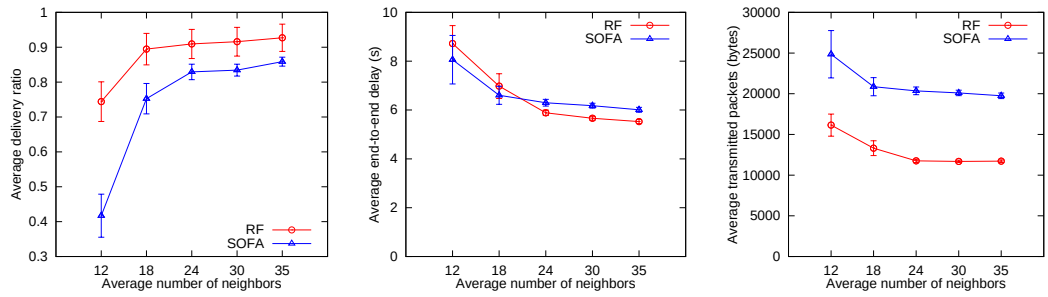
Figure 5.12: Impacts of the contention window factor

We can see from Figure 5.12(a) that the average delivery ratio is generally increasing with the increase of the contention window factor. The greater the contention window factor is, the lower the probability that nodes wake up with collision will be. As a result, the sender can have a higher probability to send the packet to the candidate with a better path quality to the sink node. This results in the higher delivery ratio. The interesting point is that when CWF is set to the value that is greater than 40, the average delivery ratio becomes variance stable.

It is shown in Figure 5.12(b) that the average end-to-end delay of RF is decreasing with the increase of the contention window factor. This is because the sender can send the packet to a better candidate with a higher probability as the contention window factor is increasing. Similar to the case in Figure 5.12(a), the average end-to-end delay is stable when the CWF is set to a value which is greater than 50. Based on the fact that the performances become stable when the CWF is greater, the contention window factor is set to 70 in the following simulations.

5.3.5 Impacts of density

The more nodes the network has, the higher the probability for a packet to reach the sink will be. In this section, we will show the impact of the density. All nodes except the source and the sink are randomly deployed. We change the number of nodes from the following set $\{50, 75, 100, 125, 150\}$. Because of the unreliable links, one node sometimes can receive packets from some neighbors that are far away. In this simulation, we only count the neighbors that have the link quality which is higher than 0.2. As a result, the average number of neighbors with the aforementioned number of nodes is 12, 18, 24, 30 and 35, respectively.



(a) Average delivery ratio (b) Average end-to-end delay (c) Average transmitted packet

Figure 5.13: Impacts of the number of nodes

We can see from Figure 5.13(a) that the average delivery ratios of both the two protocols are increasing as the average number of neighbors increases. The average delivery ratio of RF is always higher than that of SOFA. When the average number of neighbors is 12, the average delivery ratio of RF is much higher than that of SOFA (the average delivery ratio of RF is about 0.78 while the average delivery ratio of SOFA is about 0.41). The differences of average delivery ratios between the two protocols are decreasing as the average number of neighbors increases. This result shows that RF is much more robust than SOFA.

It is shown in Figure 5.13(b) that the average end-to-end delays of both the two protocols are decreasing as the density is increasing. This is because a packet can reach the sink node with less hop counts when the network density is higher. And the average end-to-end delay of RF is higher than that of SOFA when the average number of neighbors is 12 and 18. The reason is that RF can transmit more packets that experience long paths to the sink node. When the average number of neighbors is greater than 18, the average end-to-end delay of RF is lower than that of SOFA. When the network density is higher, both the two protocols have higher average delivery ratios. However, RF can transmit a packet to a better candidate (i.e. has a lower COST). This is the reason why RF has lower end-to-end delay when the network density is higher.

Figure 5.13(c) shows that the average transmitted packets of SOFA is much higher than that of RF. The average transmitted packets of both the two protocols are decreasing as the density is increasing. From those simulation results we can conclude that RF can have a higher delivery ratio with less energy consumption.

5.3.6 Impacts of dead nodes

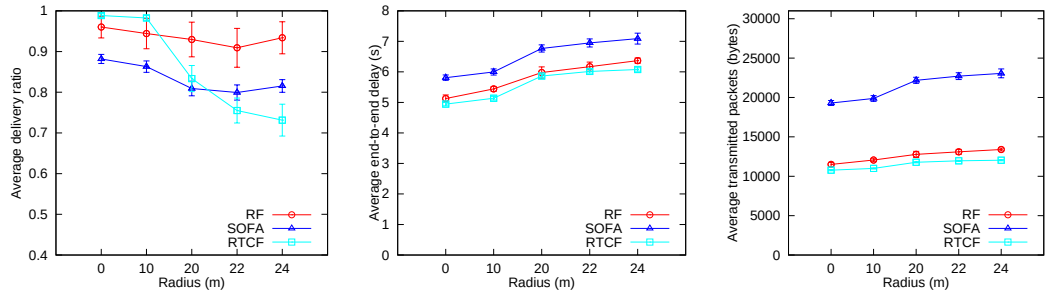
After initializing the network, every node will get a COST. During the run-time phase, if some nodes are dead, the performance of RF will be affected, for example, the average delivery ratio might decrease and the average end-to-end delay might increase. In this section, we will show the impacts of the number of dead nodes.

In the previous sections, we have compared the performances of RF to SOFA. In this section, in addition to SOFA, we also want to compare RF with RTCF because RTCF has a higher average delivery ratio when the network does not have dead nodes and it does not have update scheme. The performances of RTCF will be significantly affected when the network has some dead nodes. The configurations of RTCF are the same as that of RF. In order to show the impact of dead nodes, we want to adopt two methods to kill some nodes from the network: (1) kill the nodes in the center of the network, (2) kill the nodes randomly. In both the two cases, 200 nodes are deployed. After the initialization phase, some nodes (except the source and the sink nodes) are removed from the network.

In the center of the network

In this case, the way of removing some nodes from the network is as follow: the nodes located in the circle that is centered at (50, 50) and with a certain radius will be removed. We choose the radius from the following set $\{0, 10, 20, 22, 24\}$.

From Figure 5.14(a) we can see that the average delivery ratios of all the three protocols are decreasing as the radius is increasing. The average delivery ratio of RF is always higher than that of SOFA no matter what the radius is. And the average delivery ratio of RTCF is higher than that of RF and SOFA when the radius is 0 and 10. When the radius is greater than



(a) Average delivery ratio (b) Average end-to-end delay (c) Average transmitted packet

Figure 5.14: Impacts of the number of dead nodes (network center)

20, the average delivery ratio of RTCF is lower than that of both RF and SOFA. This is because RTCF uses the EDR as the metric that is more accurate. When the radius of the dead nodes is little, RTCF can tolerate this impact, so the average delivery ratio of RTCF is higher than that of both the other two protocols. As the radius is increasing, some nodes may lose most of or all the next-hop candidates. And RTCF does not have update scheme, so the average delivery ratio is lower than both the other protocols when the radius is greater. From this figure, we can see that RTCF has better performances when the network does not have too many dead nodes. And RTCF can not be used in the network where there are many dead nodes because the performances are significantly decreased.

In Figure 5.14(b), it is shown that the average end-to-end delays of all the three protocols are increasing with the increase of the radius. This is because the length of the path to the sink node is longer when the network has dead nodes. SOFA has the highest average end-to-end delay and the average end-to-end delay of RF is higher than that of RTCF.

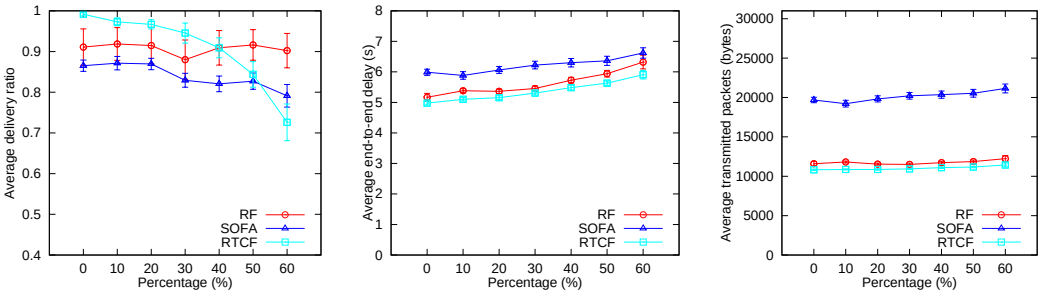
Figure 5.14(c) shows that the number of transmitted packets of SOFA is much higher than that of RF and RTCF. RTCF has the lowest average transmitted packets. From the three figures discussed above, we can conclude that RF can have a higher delivery ratio with less energy consumption when the network has dead nodes.

In Chapter 4, the main metric is EDR which is based on the packet reception success rate. In this chapter, RF uses RSSI as the main metric. Although using EDR has better performances, it is difficult to update. We have shown the performances in Figure 5.14. So that is the reason why we did not analyze the impacts of RSSI in Chapter 4. However, we did a simulation to show the impacts of the threshold of the packet reception success rate on the performances of RTCF.

Randomly

Different from the previous case, in this case, we randomly kill a certain percentage of nodes (except the source and the sink nodes) from the network. In the simulations, the percentage is changed from 0%, 10%, 20%, 30% and 40%.

In Figure 5.15(a), we can see that the average delivery ratio of RF is always higher than that



(a) Average delivery ratio (b) Average end-to-end delay (c) Average transmitted packet

Figure 5.15: Impacts of the number of dead nodes (randomly)

of SOFA. The average delivery ratio of RTCF is higher than that of RF when the percentage of dead nodes is lower than 40% while the average delivery ratio of RF is becoming higher than that of RTCF when the percentage of dead nodes is higher than 40%. This is because RTCF can not tolerate too many dead nodes while RF has update scheme. And the average delivery ratio of RTCF is lower than that of SOFA when we randomly kill 60% nodes from the network.

It is shown in Figure 5.15(b) that the average end-to-end delay of SOFA is higher than the average end-to-end delays of both the other two protocols, i.e. RTCF and RF. And the average end-to-end delay of RF is very close to the average end-to-end delay of RTCF.

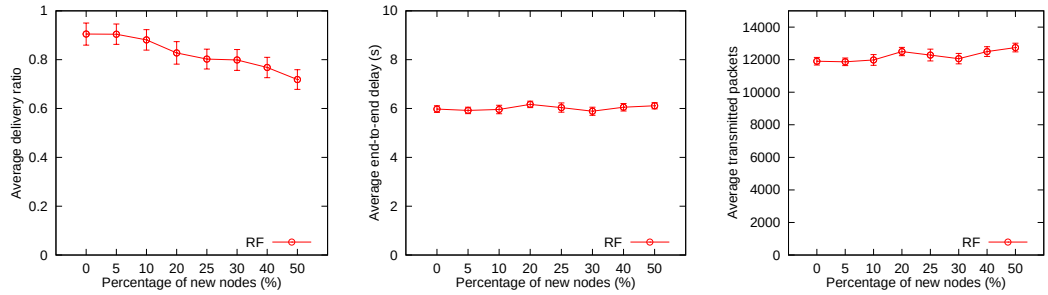
We can see from the Figure 5.15(c) that SOFA has the highest average transmitted packets among the three protocols. The average transmitted packets of SOFA is much higher than that of the other two protocols (nearly 2 times). And the average transmitted packets of RF is very close to that of RTCF.

Taking the simulation results of the previous two cases into consideration, we can conclude that RF is the most robust protocol among the three protocols. And RTCF can be used in the scenario where there do not have too many dead nodes.

5.3.7 Impacts of new added nodes

As we described in Section 5.2.4, two update schemes, proactive and on-demand, are proposed for the case where new nodes are deployed. Compared with on-demand update scheme, proactive update scheme consumes much more energy because the new deployed nodes actively send the preamble packets. As a result, we only use the on-demand update scheme in this simulation. 100 nodes are deployed and we deploy a certain percentage of new nodes when the network is in run-time phase. The percentage is changed from 5% to 50%.

It is shown in Figure 5.16(a) that the average delivery ratio is decreasing as the percentage of new nodes is increasing. The average delivery ratio is about 0.9 when there are no new nodes while the average delivery ratio is about 0.7 when there are 50% of new node (i.e. 50 new deployed nodes). And we can see from Figure 5.16(b) and Figure 5.16(c) that the average end-to-end delay and the average transmitted packets are not changed so much when we increase the percentage of new deployed nodes. Those simulation results show that RF is a highly robust



(a) Average delivery ratio (b) Average end-to-end delay (c) Average transmitted packet

Figure 5.16: Impacts of the number of new added nodes

protocol.

5.4 Summary

In this chapter, we have presented a robust forwarding protocol (RF) for ultra-low duty-cycle WSNs. Because the EDR metric used in WSEDR is not easy to update, RF adopts RSSI as the key metric for forwarding. The main advantages of RF are that the sender gives a higher priority to the candidate that has a lower COST and every node can dynamically update their COST information. We simulated RF with different settings and compared RF with SOFA. From the simulation results, we showed that RF has a higher average delivery ratio than SOFA, and the average end-to-end delay is a little higher, but the average number of transmitted packets are lower. We also compared RF with RTCF. The simulation results show that the performance of RTCF is better than that of RF when the network has no or less dead nodes. The performance of RF is better than RTCF when the network has many dead nodes.

Chapter 6

Conclusion and future work

6.1 Conclusion

Sensor nodes in WSNs are battery-powered, thus communication protocols have to be energy efficient in order to prolong the lifetime. Duty-cycle is one of the most energy efficient solutions for WSNs, but it induces other problems, in particular, longer delay. Some applications have real-time constraints, which means a packet must be sent to the sink node before a deadline. Moreover, due to the low-power transmission, wireless links among sensor nodes are highly unreliable. Furthermore, the topology in WSNs may be changed because of the dead sensor nodes or new added sensor nodes. This characteristic requires that the protocols for WSNs should be robust enough. In this thesis, considering the aforementioned problems, we have proposed some reliable and real-time constrained protocols for WSNs.

First, we reviewed the main existing protocols in WSNs in Chapter 2, including MAC, routing and cross-layer forwarding protocols. As shown in some experimental results, wireless links among low-power sensor nodes are highly unreliable, which has significant negative effects on the performances of routing protocols. Besides the impacts of unreliable links, the impacts of duty-cycle on the performances of routing protocols can not be neglected. Thus cross-layer protocols can have better performances in those cases. In this chapter, we underlined that routing over virtual coordinates can have very good performances.

The goal of Chapter 3 was to know the relationships between the three key points in order to design reliable and real-time constrained protocols, namely virtual coordinate, unreliable links and duty-cycle, as well as to get some guidelines. In Chapter 3, we first analyzed the impacts of duty-cycle MAC protocols on the performances of routing protocols. From the analyzed results, we showed that cross-layer based duty-cycle MAC protocols can significantly reduce the average end-to-end delay. In addition to the impacts of duty-cycle, we also studied the impacts of unreliable links on virtual coordinate based routing protocols. It was shown that it is better to take unreliable links into consideration when constructing the virtual coordinate. Then, we proposed three simple and efficient update schemes for constructing two dimensional

virtual coordinates. The performances of the routing protocol based on the virtual coordinate constructed by the proposed schemes were shown by large-scale simulations.

Based on the study results in Chapter 3, we proposed two versions of a cross-layer forwarding protocols in Chapter 4, i.e. synchronous version and asynchronous version. We considered the impacts of duty-cycle and unreliable links simultaneously. In synchronous forwarding protocol, we showed that different wakeup slots have significant impacts on the expected delivery ratio, and proposed a wakeup scheduling algorithm. The scheduling algorithm simply schedules the wakeup slots of every node according to their own hop count and expected delivery ratio. The performances of the proposed synchronous forwarding protocol were evaluated by large-scale simulations. The simulation results are in line with the mathematical ones. We then proposed an asynchronous version protocol. In asynchronous forwarding protocol, because every node is not synchronized to each other, preamble sampling technique was adopted to wake up the potential candidates. And the asynchronous forwarding protocol can dynamically adjust the number of potential candidates to increase the delivery ratio by considering the remaining time before the deadline. The performances of the proposed asynchronous forwarding protocol were shown by a large number of simulations.

The two cross-layer forwarding protocols can have good performances in the scenario where the topology is relatively stable. However, dynamicity is an important characteristic in some types of WSNs, for example, link quality is changing very fast, some nodes may die due to energy exhaustion and some new nodes may be added to the network. Because of the topology changes, it is necessary to design robust protocols for those WSNs. In Chapter 5, a robust forwarding (RF) protocol was proposed. RF can automatically update the network when the network topology is changing. The forwarding metric of RF is based on RSSI/LQI. The performances of RF were compared to SOFA and RTCF because, to the best of our knowledge, SOFA is the only forwarding protocol which takes the low duty-cycle into consideration. Simulation results showed that RF can have much better performances when the network topology changes.

6.2 Future work

In both WSEDR and RTCF, EDR is adopted as the main metric for forwarding. EDR can precisely indicate which node is better to be chosen as the next-hop node, but it is not so easy and expensive to calculate EDR. In order to calculate the EDR, a node needs to gather the information of some neighbors which have the lower hop count. Because of the unreliable links, it is not enough for the sink node to only send one initialization packet. In order to have a preciser EDR, the sink node has to send more than one initialization packets. The more the initialization packets are sent, the preciser the EDR will be. However, the energy consumption will be higher as the number of the initialization packets increases. As a result, there exists a trade-off between the accuracy and the energy consumption. It is interesting to know the

optimal number of initialization packets and to find other metrics that are easier to calculate and have the comparable performances.

The performances of all our proposed protocols are verified by simulations and mathematical analysis. Although the simulation results can well show the performances of the proposed protocols, it is still necessary to verify our proposed protocols on a realistic testbed because the simulation models (e.g. physical layer models) used in the simulator can only reflect the realistic scenarios to a certain extent even if the used simulator, WSNET, can modelize a lot of physical phenomena (shadowing, fading, etc). To the best of our knowledge, WSNET is one of the simulators which have the most precise models of the physical layer. As we all know, it is always difficult to do experimentations. To implement the protocols on testbeds needs a lot of time (specially for debugging). It is also difficult to collect the running parameters of the sensor nodes in large-scale testbeds. In our case, reliability and delay are specially difficult to measure.

In this thesis, we propose protocols for real-time data gathering in WSNs. Other phases in a real-time application also have to be real-time, e.g. broadcast and aggregation. There are some cases where the sink node needs to broadcast some packets to the whole network. For example, the sink node wants to update the program or parameters of all the sensor nodes. Although broadcasting packets to the whole network is not a new research subject, to broadcast packets under ultra-low duty-cycle is a new research subject. There exist some broadcast protocols designed for ultra-low duty-cycle WSNs [Guo et al., 2009; Lai and Ravindran, 2010; Sun et al., 2009; Wang and Liu, 2009]. Nevertheless, they focus on the reliability and do not consider the real-time constraints. As a result, to broadcast messages to all the sensor nodes within a deadline for ultra-low duty-cycle WSNs is an interesting work. For example, how to design a broadcast protocol which can bound the broadcast delay, maximize the delivery ratio and minimize the number of broadcast packets?

After an event happens, many nodes may detect it and want to send packets. The generated packets are aggregated into an alarm packet and only one node sends it. For real-time applications, it is necessary to have real-time aggregation algorithms because the interval from the event happens to the time when the packet is received by the sink must be bounded. So to design a real-time aggregation protocol for WSNs is also an interesting work. For example, how to design an aggregation protocol which can bound the delay from the moment when the event happens to the moment when the packets from the nodes that detected the event are aggregated into one alarm?

According to the deadline of the network, we can divide the deadline into two sections, one for aggregation process and another for transmission process. Then the parameters of the protocols (e.g. WSEDR, RTCF or aggregation protocols) can be adjusted in order to satisfy the deadline and lifetime requirements. If both the aggregation and the transmission processes satisfy their own deadline, the deadline of the network can be satisfied. In this case, the parameters depend on the requirements of the applications. Another point of view of this problem is

to consider the requirements of the applications and then give some guidelines for the deployment.

Both the simulation results on a simulator and the experimental results on a real testbed can show the performances of a protocol, but they can not guarantee that the protocol can work as good as they want in all the different cases. By using the mathematic method, we have proven that the scheduling algorithm can optimize the delivery ratio. However, modeling our protocols with formal methods can validate the implementations of our algorithm. Moreover, in mathematical mode, we only consider the average link quality. With formal model, we can have preciser link quality models and then show precisely the performances of our protocols. So it is also very interesting to verify our protocol with formal validation method. For our protocols, the time constraints and probability should be modeled. Some formal verification tools can realize this, for example, PRISM [Hinton et al., 2006] which is a probabilistic model checking tool. The main difficulty is to take into account of the unreliable links. As the topology is dynamic, the number of states is increasing a lot. Therefore, it is impossible, in practice, to adopt this kind of tools directly to verify wireless protocols. Further works are needed to study how to use this kind of tools to verify wireless protocols.

Appendix A

List of publications

A.1 International Journal:

(1) Fei Yang and Isabelle Augé-Blum. Delivery ratio-maximized wakeup scheduling for ultra-low duty-cycled WSNs under real-time constraints, Computer Networks, 2010. (to appear)

A.2 Book Chapter:

(1) Isabelle Augé-Blum, Fei Yang and Thomas Watteyne. Real-time communications in Wireless Sensor Networks, Handbook of Research on Next Generation Networks and Ubiquitous Computing, IGI Global, 2009.

A.3 International Conference:

(1) Fei Yang and Isabelle Augé-Blum. Constructing Virtual Coordinate for Routing in Wireless Sensor Networks under Unreliable Links, in proceedings of the 5th International Wireless Communications and Mobile Computing Conference (IWCMC'09).

(2) Fei Yang and Isabelle Augé-Blum. On maximizing the delivery ratio of ultra-low duty-cycle WSNs under real-time constraints, in proceedings of the 17th Annual Meeting of the IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'09).

Bibliography

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, 38:393–422.

Akyildiz, I., Wang, X., and Wang, W. (2005). Wireless mesh networks: a survey. *Computer Networks*, 47(4):445–487.

Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., et al. (2004). A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634.

Baronti, P., Pillai, P., Chook, V., Chessa, S., Gotta, A., and Hu, Y. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards. *Computer Communications*, 30(7):1655–1695.

Becker, M., Beylot, A., Dhaou, R., Gupta, A., Kacimi, R., and Marot, M. (2009). Experimental study: Link quality and deployment issues in wireless sensor networks. In *Proceedings of the 8th IFIP International Conferences on Networking (Networking'09)*.

Bose, P., Morin, P., Stojmenović, I., and Urrutia, J. (2001). Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 6(6):609–616.

Boughanmi, N. and Song, Y. (2008). A new routing metric for satisfying both energy and delay constraints in wireless sensor networks. *Journal of Signal Processing Systems*, 51(2):137–143.

Buettner, M., Yee, G., Anderson, E., and Han, R. (2006). X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys'06)*.

Caccamo, M., Zhang, L., Sha, L., and Buttazzo, G. (2002). An implicit prioritized access protocol for wireless sensor networks. In *Proceedings of the 23rd IEEE International Real-Time Systems Symposium (RTSS'02)*.

Cao, Q. and Abdelzaher, T. (2004). A scalable logical coordinates framework for routing in wireless sensor networks. In *Proceedings of the 25rd IEEE International Real-Time Systems Symposium (RTSS'04)*.

Cao, Q. and Abdelzaher, T. (2006). Scalable logical coordinates framework for routing in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):557–593.

Cao, Q., Abdelzaher, T., He, T., and Stankovic, J. (2005). Towards optimal sleep scheduling in sensor networks for rare-event detection. In *Proceedings of the fourth IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN'05)*.

Caruso, A., Chessa, S., De, S., and Urpi, A. (2005). Gps free coordinate assignment and routing in wireless sensor networks. In *Proceedings of the 24th IEEE International Conference on Computer Communications (INFOCOM'05)*.

CC2420 (2007). Cc2420 datasheet. `http://www.ti.com/lit/gpn/cc2420`.

Ceriotti, M., Mottola, L., Picco, G., Murphy, A., Guna, S., Corra, M., Pozzi, M., Zonta, D., and Zanon, P. (2009). Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *Proceedings of the 8th IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN'09)*.

Cerpa, A., Wong, J., Potkonjak, M., and Estrin, D. (2005). Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *Proceedings of the 6th ACM international symposium on mobile ad hoc networking and computing (MobiHoc'05)*.

Chebrolu, K., Raman, B., Mishra, N., Valiveti, P. K., and Kumar, R. (2008). Brimon: a sensor network system for railway bridge monitoring. In *Proceedings of the sixth International Conference on Mobile System, Applications, and Services (MobiSys'08)*.

Chen, J., Lin, R., Li, Y., and Sun, Y. (2008). Lqer: A link quality estimation based routing for wireless sensor networks. *Sensors*, 8(2):1025–1038.

Chen, J., Zhu, P., and Qi, Z. (2007). Pr-mac: Path-oriented real-time mac protocol for wireless sensor network. In *Proceedings of the third International conference on embedded software and systems (ICESS'07)*.

Chen, Y. and Fleury, E. (2007). A distributed policy scheduling for wireless sensor networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*.

Chipara, O., He, Z., Xing, G., Chen, Q., Wang, X., Lu, C., Stankovic, J., and Abdelzaher, T. (2006). Real-time power-aware routing in sensor networks. In *Proceedings of the Fourteenth IEEE International Workshop on Quality of Service (IWQoS'06)*.

Chong, C. and Kumar, S. (2003). Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256.

Dhanaraj, M., Manoj, B., and Murthy, C. (2005). A new energy efficient protocol for minimizing multi-hop latency in wireless sensor networks. In *Proceedings of the 3rd Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'05)*.

Doolin, D. M. and Sitar, N. (2005). Wireless sensors for wildfire monitoring. In *Proceedings of the SPIE symposium on smart structures and materials*.

Dulman, S., Rossi, M., Havinga, P., and Zorzi, M. (2006). On the hop count statistics for randomly deployed wireless sensor networks. *International Journal of Sensor Networks*, 1(1):89–102.

El-Hoiydi, A. (2002). Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC'02)*.

El-Hoiydi, A. and Decotignie., J.-D. (2004). Wisemac: An ultra low power mac protocol for multi-hop wireless sensor networks. In *Proceedings of the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS'04)*.

Ergen, S. and Varaiya, P. (2006). Pedamacs: Power efficient and delay aware medium access protocol for sensor networks. *IEEE Transactions on Mobile Computing*, 5(7):920–930.

Felemban, E., Lee, C., and Ekici, E. (2006). Mmspeed: Multipath multi-speed protocol for qos guarantee of reliability and timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754.

Fonseca, R., Ratnasamy, S., Zhao, J., Ee, C., Culler, D., Shenker, S., and Stoica, I. (2005). Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05)*.

Gad-el Hak, M. (2002). *The MEMS handbook*. CRC, New York.

Ganeriwal, S., Kumar, R., and Srivastava, M. (2003). Timing-sync protocol for sensor networks. In *Proceedings of the The First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*.

GloMoSim (2000). Global mobile information system simulation library. `http://pcl.cs.ucla.edu/projects/glomosim/`.

GTNetS (2008). The georgia tech network simulator. `http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/index.html`.

Gu, L. and Stankovic, J. (2005). Radio-triggered wake-up for wireless sensor networks. *Real-Time Systems*, 29(2):157–182.

Gu, Y. and He, T. (2007). Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links. In *Proceedings of The 5th ACM Conference on Embedded Networked Sensor Systems (SenSys'07)*.

Gu, Y., He, T., Lin, M., and Xu, J. (2009). Spatiotemporal delay control for low-duty-cycle sensor networks. In *Proceedings of The 30th IEEE International Real-Time Systems Symposium (RTSS'09)*.

Guo, S., Gu, Y., Jiang, B., and He, T. (2009). Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. In *Proceedings of The 15th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'09)*.

Han, K., Ko, Y., and Kim, J. (2004). A novel gradient approach for efficient data dissemination in wireless sensor networks. In *Proceedings of the 60th IEEE Vehicular Technology Conference (VTC'04-Fall)*.

Hartung, C., Han, R., Seielstad, C., and Holbrook, S. (2006). Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proceedings of the 4th International Conference on Mobile System, Applications, and Services (MobiSys'06)*.

He, T., Blum, B., Cao, Q., Stankovic, J., Son, S., and Abdelzaher, T. (2007). Robust and timely communication over highly dynamic sensor networks. *Real-Time Systems*, 37(3):261–289.

He, T., Krishnamurthy, S., Luo, L., Yan, T., Gu, L., Stoleru, R., Zhou, G., Cao, Q., Vicaire, P., and Stankovic, J. (2006a). Vigilnet: An integrated sensor network system for energy-efficient surveillance. *ACM Transactions on Sensor Networks*, 2(1):1–38.

He, T., Stankovic, J., Lu, C., and Abdelzaher, T. (2003). Speed: a stateless protocol for real-time communication in sensor networks. In *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03)*.

He, T., Vicaire, P., Yan, T., Luo, L., Gu, L., Zhou, G., Stoleru, R., Cao, Q., Stankovic, J. A., and Abdelzaher, T. (2006b). Achieving real-time target tracking using wireless sensor networks. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*.

Hefeeda, M. and Bagheri, M. (2008). Forest fire modeling and early detection using wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 7(1):169–224.

Heinzelman, W., Chandrakasan, A., Balakrishnan, H., and MIT, C. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670.

Hightower, J. and Borriello, G. (2001). Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66.

Hinton, A., Kwiatkowska, M., Norman, G., and Parker, D. (2006). Prism: A tool for automatic verification of probabilistic systems. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*.

Hong, Y. and Scaglione, A. (2005). A scalable synchronization protocol for large scale sensor networks and its applications. *IEEE Journal on Selected Areas in Communications*, 23(5):1085–1099.

Huang, P., Chen, H., Xing, G., and Tan, Y. (2009). Sgf: A state-free gradient-based forwarding protocol for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 5(2):1–25.

IEC-Standard (2010). Wirelesshart specification. `http://www.hartcomm.org/protocol/wihart/wireless_technology.html`.

IEEE-Standard (1999). Wireless LAN medium access control (MAC) and physical layer (PHY) specification.

IEEE-Standard (2003). Wireless medium access control(MAC) and physical layer(PHY) specifications for low-rate wireless personal area networks(LR-WPANs).

IETF (2010a). Ipv6 over low power wpan. `http://datatracker.ietf.org/wg/6lowpan/charter/`.

IETF (2010b). Routing over low power and lossy networks. `https://datatracker.ietf.org/wg/roll/charter/`.

Imote2 (2007). Imote2 datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf`.

Jamieson, K., Balakrishnan, H., and Tay, Y. (2006). Sift: A mac protocol for event-driven wireless sensor networks. In *Proceedings of the Third European Workshop on Wireless Sensor Networks (EWSN'06)*, pages 260–275.

Johnson, D. B. and Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. In Imielinski, T. and Korth, H., editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers.

Karl, H. and Willig, A. (2005). *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, Ltd.

Karp, B. and Kung, H. (2000). Gpsr: Greedy perimeter stateless routing for wireless sensor networks. In *Proceedings of The 6th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'00)*.

Keshavarzian, A., Lee, H., and Venkatraman, L. (2006). Wakeup scheduling in wireless sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'06)*.

Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., and Turon, M. (2007). Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of The 6th IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN'07)*.

Ko, Y. and Vaidya, N. (2000). Location-aided routing (lar) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321.

Kosucu, B., Irgan, K., Kucuk, G., and Baydere, S. (2009). Firesensetb: a wireless sensor networks testbed for forest fire detection. In *Proceedings of The 5th International Wireless Communications and Mobile Computing Conference (IWCMC'09)*.

Kulkarni, S., Iyer, A., and Rosenberg, C. (2006). An address-light, integrated mac and routing protocol for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 14(4):793–806.

Kumar, S., Raghavan, V., and Deng, J. (2006). Medium access control protocols for ad hoc wireless networks: A survey. *Ad Hoc Networks*, 4(3):326–358.

Kuorilehto, M., Hannikainen, M., and Hamalainen, T. (2005). A survey of application distribution in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2005(5):774–788.

Lai, S. and Ravindran, B. (2010). On multihop broadcast over adaptively duty-cycled wireless sensor networks. In *Proceedings of The 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'11)*.

Lee, S., Kwak, K., and Campbell, A. (2006). Solicitation-based forwarding for sensor networks. In *Proceedings of Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'06)*.

Lee, W., Datta, A., and Cardell-Oliver, R. (2008). Flexitp: A flexible-schedule-based tdma protocol for fault-tolerant and energy-efficient wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(6):851–864.

Leong, B., Liskov, B., and Morris, R. (2007). Greedy virtual coordinates for geographic routing. In *Proceedings of the 15th IEEE International Conference on Network Protocols (ICNP'07)*.

Li, M. and Liu, Y. (2009). Underground coal mine monitoring with wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(2):1–29.

Li, Y., Chen, C., Song, Y., and Wang, Z. (2007). Real-time qos support in wireless sensor networks: a survey. In *Proceedings of the 7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems*.

Li, Y., Chen, C., Song, Y., Wang, Z., and Sun, Y. (2009). Enhancing real-time delivery in wireless sensor networks with two-hop information. *IEEE Transactions on Industrial Informatics*, 5(2):113–122.

Lin, E., Rabaey, J., and Wolisz, A. (2004). Power-efficient rendez-vous schemes for dense wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC'04)*.

Lin, S., Zhang, J., Zhou, G., Gu, L., Stankovic, J., and He, T. (2006). Atpc: adaptive transmission power control for wireless sensor networks. In *Proceedings of The 4th ACM Conference on Embedded Networked Sensor Systems (SenSys'06)*.

Liu, R., Fan, K., and Sinha, P. (2009). Locally scheduled packet bursting for data collection in wireless sensor networks. *Ad Hoc Networks*, 7(5):904–917.

Lorincz, K., Chen, B., Challen, G., Chowdhury, A., Patel, S., Bonato, P., and Welsh, M. (2009). Mercury: A wearable sensor network platform for high-fidelity motion analysis. In *Proceedings of The 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*.

Lu, C., Blum, B., Abdelzaher, T., Stankovic, J., and He, T. (2002). Rap: a real-time communication architecture for large-scale wireless sensor networks. In *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*.

Lu, G., Krishnamachari, B., and Raghavendra, C. (2004). An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *Proceedings of The 18th IEEE International Parallel & Distributed Processing Symposium (IPDPS'04)*.

Lu, G., Sadagopan, N., Krishnamachari, B., and Goel, A. (2005). Delay efficient sleep scheduling in wireless sensor networks. In *Proceedings of The 24th IEEE International Conference on Computer Communications (INFOCOM'05)*.

Macedo, M., Grilo, A., and Nunes, M. (2009). Distributed latency-energy minimization and interference avoidance in tdma wireless sensor networks. *Computer Networks*, 53(5):569–582.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*.

Mangharam, R., Rowe, A., and Rajkumar, R. (2007). Firefly: a cross-layer platform for real-time embedded wireless networks. *Real-Time Systems*, 37(3):183–231.

Maróti, M., Kusy, B., Simon, G., and Lédeczi, Á. (2004). The flooding time synchronization protocol. In *Proceedings of The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*.

Mauve, M., Widmer, A., and Hartenstein, H. (2002). A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, 15(6):30–39.

Melodia, T., Vuran, M., and Pompili, D. (2006). The state of the art in cross-layer design for wireless sensor networks. In *Proceedings of the Second International Workshop of the EURO-NGI Network of Excellence on Wireless Systems and Network Architectures in Next Generation Internet*.

Mica2 (2002). Mica2 datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`.

MicaZ (2005). Micaz datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

MultihopLQI (2004). Multihoplqi. `http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI`.

Na, J., Lim, S., and Kim, C. (2008). Dual wake-up low power listening for duty cycled wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2008:11.

Nguyen, A., Milosavljevic, N., Fang, Q., Gao, J., and Guibas, L. (2007). Landmark selection and greedy landmark-descent routing for sensor networks. In *Proceedings of The 26th IEEE International Conference on Computer Communications (INFOCOM'07)*.

NS2 (2008). The network simulator-ns2. `http://www.isi.edu/nsnam/ns/`.

Patwari, N., Hero III, A., Perkins, M., Correal, N., and O'dea, R. (2003). Relative location estimation in wireless sensor networks. *IEEE Transactions on Signal Processing*, 51(8):2137–2148.

Perkins, C. and Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM'94)*.

Perkins, C. and Royer, E. (1999). Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*.

Pister, K. and Doherty, L. (2008). Tsmp: Time synchronized mesh protocol. In *Proceedings of the IASTED International Symposium on Distributed Sensor Networks*.

Polastre, J., Hill, J., and Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *Proceedings of The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*.

Polastre, J., Szewczyk, R., and Culler, D. (2005). Telos: Enabling ultra-low power wireless research. In *The fourth IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN'05)*.

Rajendran, V., Obraczka, K., and Garcia-Luna-Aceves, J. J. (2003). Energy-efficient collision-free medium access control for wireless sensor networks. In *Proceedings of The First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*.

Rao, A., Papadimitriou, C., Shenker, S., and Stoica, I. (2003). Geographic routing without location information. In *Proceedings of The 9th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'03)*.

Rappapport, T. S. (1996). *Wireless communications: principles and practice*. Prentice Hall.

Roedig, U., Barroso, A., and Sreenan, C. (2006). f-mac: a deterministic media access control protocol without time synchronization. In *Proceedings of the Third European Workshop on Wireless Sensor Networks (EWSN'06)*.

Ross, S. M. (2004). *Introduction to probability and statistics for engineers and scientists*. Academic Press.

Sanchez, J., Marin-Perez, R., and Ruiz, P. (2007). Boss: Beacon-less on demand strategy for geographic routing in wireless sensor networks. In *Proceedings of The Fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'07)*.

Sang, L., Arora, A., and Zhang, H. (2007). On exploiting asymmetric wireless links via one-way estimation. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'07)*.

Schurgers, C., Tsiatsis, V., Ganeriwal, S., and Srivastava, M. (2002). Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 1(1):70–80.

Seada, K., Helmy, A., and Govindan, R. (2004a). On the effect of localization errors on geographic face routing in sensor networks. In *Proceedings of The 3rd IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN'04)*.

Seada, K., Zuniga, M., Helmy, A., and Krishnamachari, B. (2004b). Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *Proceedings of The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*.

Shah, R., Wietholter, S., Wolisz, A., and Rabaey, J. (2005). When does opportunistic routing make sense? In *Proceedings of the 3rd Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'05)*.

Sohrabi, K., Manriquez, B., and Pottie, G. (1999). Near ground wideband channel measurement in 800-1000 mhz. In *Proceedings of the 55th IEEE Vehicular Technology Conference (VTC'99)*.

Sommer, P. and Wattenhofer, R. (2009). Gradient clock synchronization in wireless sensor networks. In *Proceedings of The 8th IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN'09)*.

Son, B., Her, Y., and Kim, J. (2006). A design and implementation of forest-fires surveillance system based on wireless sensor networks for south korea mountains. *International journal of computer science and network security*, 6(9b):124–130.

Song, W., LaHusen, R., Huang, R., Ma, A., Xu, M., Van der Haegen, M., and Shirazi, B. (2009). Air-dropped sensor network for real-time high-fidelity volcano monitoring. In *Proceedings of the 7th International Conference on Mobile System, Applications, and Services (MobiSys'09)*.

Srinivasan, K., Dutta, P., Tavakoli, A., and Levis, P. (2008). An empirical study of low power wireless. Technical report, Stanford University.

Stankovic, J., Abdelzaher, T., Lu, C., Sha, L., and Hou, J. (2003). Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91:1002–1022.

Stojmenovic, I. (2002). Position-based routing in ad hoc networks. *IEEE Communications Magazine*, 40(7):128–134.

Stojmenovic, I., Nayak, A., and Kuruvila, J. (2005). Design guidelines for routing protocols in ad hoc and sensor networks with a realistic physical layer. *IEEE Communications Magazine*, 43(3):101–106.

Strasser, M., Meier, A. F., Langendoen, K., and Blum, P. (2007). Dwarf: Delay-aware robust forwarding for energy-constrained wireless sensor networks. In *Proceedings of The 3rd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'07)*.

Su, H. and Zhang, X. (2009). Battery-dynamics driven tdma mac protocols for wireless body-area monitoring networks in healthcare applications. *IEEE Journal on Selected Aeras in Communications*, 27(4):424–434.

Sun, Y., Gurewitz, O., Du, S., Tang, L., and Johnson, D. (2009). Adb: An efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks. In *Proceedings of The 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*.

Sun, Y., Gurewitz, O., and Johnson, D. B. (2008). Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of The 6th ACM Conference on Embedded Networked Sensor Systems (SenSys'08)*.

Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., and Culler, D. (2004). An analysis of a large scale habitat monitoring application. In *Proceedings of The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*.

Tanenbaum, A. S. (2003). *Computer Networks (Fourth Edition)*. Prentice Hall PTR.

Tay, Y., Jamieson, K., and Balakrishnan, H. (2004). Collision-minimizing csma and its applications to wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 22(6):1048–1057.

Tilak, S., Abu-Ghazaleh, N., and Heinzelman, W. (2002). A taxonomy of wireless microsensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(2):28–36.

Tmote-sky (2006). Tmote sky datasheet. `http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf`.

Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D., and Hong, W. (2005). A macroscope in the redwoods. In *Proceedings of The 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys'05)*.

Van Dam, T. and Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of The First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*.

Vuran, M. and Akyildiz, I. (2006). Spatial correlation-based collaborative medium access control in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 14(2):316–329.

Wakamiya, N., Kawai, T., Murata, M., Yanagihara, K., Nozaki, M., and Fukunaga, S. (2009). A sensor network protocol for automatic meter reading in an apartment building. *Ad Hoc & Sensor Wireless Networks*, 7(1-2):115–137.

Wang, C., Zeng, G., and Xiao, L. (2007). Optimizing end to end routing performance in wireless sensor networks. In *Proceedings of The 3rd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'07)*.

Wang, F. and Liu, J. (2009). Duty-cycle-aware broadcast in wireless sensor networks. In *Proceedings of The 28th IEEE International Conference on Computer Communications (INFOCOM'09)*.

Wang, X., Wang, X., Fu, X., Xing, G., and Jha, N. (2009). Flow-based real-time communication in multi-channel wireless sensor networks. In *Proceedings of the 6th European Workshop on Wireless Sensor Networks (EWSN'09)*.

Wang, X., Wang, X., Xing, G., and Yao, Y. (2010). Dynamic duty cycle control for end-to-end delay guarantees in wireless sensor networks. In *Proceedings of the 18th IEEE International Workshop on Quality of Service (IWQoS'10)*.

Wang, Y., Attebury, G., and Ramamurthy, B. (2006). A survey of security issues in wireless sensor networks. *IEEE Communications Surveys and Tutorials*, 8(2):2–23.

Watteyne, T. and Augé-Blum, I. (2005). Proposition of a hard real-time mac protocol for wireless sensor networks. In *Proceedings of The 13th Annual Meeting of the IEEE International Symposium on Modeling,Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'05)*.

Watteyne, T., Simplot-Ryl, D., Augé-Blum, I., and Dohler, M. (2007). On using virtual coordinates for routing in the context of wireless sensor networks. In *Proceedings of The 18nd IEEE Symposium on Personal, Indoor, Mobile and Radio Communications (PIMRC'07)*.

Woo, A., Tong, T., and Culler, D. (2003). Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of The First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*.

Wood, A., Virone, G., Doan, T., Cao, Q., Selavo, L., Wu, Y., Fang, L., He, Z., Lin, S., and Stankovic, J. (2006). Alarm-net: Wireless sensor networks for assisted-living and residential monitoring. Technical report, University of Virginia Computer Science Department.

Xiao, S., Dhamdhere, A., Sivaraman, V., and Burdett, A. (2009). Transmission power control in body area sensor networks for healthcare monitoring. *IEEE Journal on Selected Areas in Communications*, 27(1):37–48.

Xu, N. (2002). A survey of sensor network applications. *IEEE Communications Magazine*, 40(8):102–114.

Yang, X. and Vaidya, N. (2004). A wakeup scheme for sensor networks: Achieving balance between energy saving and end-to-end delay. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04)*.

Ye, F., Chen, A., Lu, S., and Zhang, L. (2001). A scalable solution to minimum cost forwarding in large sensor networks. In *Proceedings of the 10th IEEE International Conference on Computer Communication Networks (ICCCN'01)*, pages 304–309.

Ye, F., Zhong, G., Cheng, J., Lu, S., and Zhang, L. (2003). Peas: A robust energy conserving protocol for long-lived sensor networks. In *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03)*.

Ye, F., Zhong, G., Lu, S., and Zhang, L. (2005). Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *Wireless Networks*, 11(3):285–298.

Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of The 21st IEEE International Conference on Computer Communications (INFOCOM'02)*.

Ye, W., Heidemann, J., and Estrin, D. (2004). Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking(TON)*, 12(3):493–506.

Younis, O. and Fahmy, S. (2004). Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379.

Zeng, K., Lou, W., Yang, J., and Brown, D. (2007). On geographic collaborative forwarding in wireless ad hoc and sensor networks. In *Proceedings of the 6th ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'07)*.

Zhang, H., Arora, A., and Sinha, P. (2009). Link estimation and routing in sensor network backbones: Beacon-based or data-driven? *IEEE Transactions on Mobile Computing*, 8(5):653–667.

Zhang, W. and Cao, G. (2004). Dctc: dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transactions on Wireless Communications*, 3(5):1689–1701.

Zhao, J. and Govindan, R. (2003). Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of The First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*.

Zhou, G., He, T., Krishnamurthy, S., and Stankovic, J. (2004). Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd International Conference on Mobile System, Applications, and Services (MobiSys'04)*.

Zhou, Y. and Medidi, M. (2007). Sleep-based topology control for wakeup scheduling in wireless sensor networks. In *Proceedings of 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07)*.

ZigBee-Alliance (2008). Zigbee specification. `http://www.zigbee.org/`.

Zorzi, M. and Rao, R. (2003). Geographic random forwarding (geraf) for ad hoc and sensor networks: Energy and latency performance. *IEEE Transactions on Mobile Computing*, 2(4):349–365.

Zuniga, M. and Krishnamachari, B. (2004). Analyzing the transitional region in low power wireless links. In *Proceedings of first Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'04)*.